Techniques for ubiquitous reliable data storage

Volume 2 (of 2)

Michael Gilroy

A themed portfolio submitted to

the Universities of

Edinburgh

Glasgow

Heriot-Watt and

Strathclyde

For the Degree of

Doctor of Engineering in System Level Integration

Table Of Contents

Appendix I:

Published Papers

IEE/ACM Postgraduate Seminar on SoC Design, Test and Technology, 15th September 2004, Loughbourgh University

SERIAL ATA FOR FPGA TARGETS

Michael Gilroy[1], James Irvine[2]

[1] Institute of System Level Integration and A2E, UK
[2] University of Strathclyde, UK

ABSTRACT

Serial ATA is the latest in a line of high-speed serial communication standards introduced as a high bandwidth replacement for older parallel devices. Serial ATA 1.0 has been released as a direct replacement for the current Advanced Technology Attachment (ATA) communication standard in personal computer systems.

This paper details the stages involved in the development of a generic first generation Serial ATA controller IP block for an FPGA target from initial design through to simulation and test. This paper details the results of the investigation into a design for physical implementation using an FPGA target to provide the means to test in hardware the proposed design.

INTRODUCTION

Serial ATA is the latest in a number of high-speed serial communication standards introduced as a high bandwidth replacement for older parallel devices. With the introduction of Serial ATA the demise of the longstanding Parallel ATA standard seems immanent. This paper introduces the Serial ATA standard and presents the design for a generic implementation of a Serial ATA controller IP block for an FPGA target.

Unlike other high-speed serial communication standard such as USB 2.0 and IEEE 1394 FireWire, which allow high-speed connectivity for external devices, Serial ATA has been developed as a replacement for internal connections between the computer motherboard and hard disk, CD and DVD disk drives. Whilst the Serial Attached SCSI (SAS) standard has also been developed for similar purposes, that standard had not been completed by the time of this project was underway.

Serial ATA is currently in the process of replacing its parallel equivalent in both home and enterprise computer systems and continues to offer a cheaper alternative to SCSI devices. This paper discusses the design considerations for the implementation of a generic Serial ATA controller IP block.

FPGAs have developed to the stage where they can readily be considered for implementation in high-speed communication systems with both Altera and Xilinx providing FPGAs with multi Giga bit transceivers, allowing the versatility and ease of development of FPGAs to be applied to consumer devices in this field. At the outset of this development work there were as yet no SATA controllers available for FPGA targets. This has since changed with cores available for both protocol testing and hardware implementation.

The aims of this development were to rapidly develop a synthesisable Serial ATA controller for implementation on FPGA targets. In addition it was desirable for the generated IP core to be sufficiently compact to allow additional components and IP blocks to be implemented on the same FPGA.

SERIAL ATA

The storage capacity available from computer hard disk drives has increased rapidly in the last 30 years. However the speed of transfer mechanisms have not increased at a similar rate. Parallel ATA and SCSI systems speeds were increased sporadically to keep pace with the increases in hard disk capacity, with the former going from an initial 3 MB/second to today's 133MB/second. These speed increases have been made in such a way as to ensure continued backward compatibility but not for the future needs for data transfer.

The Serial ATA standard addresses these issues from the outset by looking to the future and providing a simple roadmap for the introduction of next generation devices and doubling the transfer speed for each generation. First generation Serial ATA transfers data at up to 1.5Gbit/second (150MB/second). Whilst this in itself is not a huge improvement over the existing Parallel ATA standard, second generation devices will double transfer speeds to 3Gbit/second.

Serial ATA therefore provides a scalable interface which can be supported by next generation devices for a number of years. In addition, Serial ATA devices are compatible with Parallel ATA devices through the use of appropriate cables which convert between them, allowing Serial ATA hosts to communicate with Parallel ATA devices and vice versa. This guarantee of continued support for legacy devices is desirable to assist migration to this new technology.

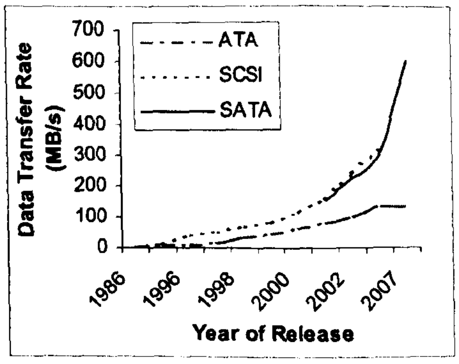


Figure 1: Transfer speed of Parallel ATA, Serial ATA and SCSI systems from the 1986 until 2007

SATA Physical Characteristics

Serial ATA uses point-to-point communications allowing the entire channel bandwidth to be dedicated to the link. This has the added advantage of simplifying the software required to control devices as there is no need to handle the various possible master/slave connections which exist in Parallel ATA. The Serial ATA connectors and cabling are far simpler, thinner and longer than their Parallel ATA predecessors.

There are only 4 signal pins used in Serial ATA connections and a voltage of only 250 milli volts is required to transmit data between Serial ATA devices. Such voltage levels are more suited to integration into a SoC design than the 3.3-5V supplies required for Parallel ATA.

Serial ATA offers a cheaper method of implementing enterprise systems and offers a viable alternative to SCSI in the enterprise market place (1). Although this advantage will be more explicit in second generation Serial ATA devices the large difference in prices of Serial ATA devices when compared with SCSI devices should aid Serial ATA uptake. Serial ATA also supports hot-plugging (installation of a new drive whilst the computer system is running) which is a major advantage in enterprise systems. This functionality would allow the capacity of a RAID system to be upgraded with no down time and allow system administrator to replace faulty drives more easily.

To date Serial ATA controllers have been added to hard disk drives only, as the transfer rates that it offers are not required for CD or floppy disk drives (2). Although Serial ATA is now being supported by major motherboard manufacturers, until devices other than hard disk drives are developed which support Serial ATA, both Parallel and Serial ATA interfaces will be needed within computer systems.

Table 1 summarises the differences between Serial ATA 1.0 and Parallel ATA.

Frame Information Structures

Within the Serial ATA standard, a frame is defined as a group of Dwords (32-bits) that convey information between the host and device. The frame is bounded by primitives, which may be inserted to control the rate of information flow. The content of a frame is known as the payload. The information field's contents are divided into three categories: register type; setup type; and data type. The type and layout of the payload is indicated by the contents of the first Dword of information located in byte 0 of the payload.

The various types of frame payload contents are concerned with the transmission of the contents of the Shadow Register Block Registers, Direct Memory Access (DMA) registers and PIO Registers. The Shadow Register Block Registers are used to ensure continued support for legacy systems whilst DMA Registers which are not used by legacy systems are required for continued development of the Serial ATA standard in the future. These registers contents are not as yet fully characterised with many reserved for future generations of Serial ATA.

TABLE 1 - Comparison of Serial ATA 1.0 and Parallel ATA devices (3)

	Serial ATA Gen 1	**Parallel ATA**
Theoretical Transfer rate	150 Mbytes/sec	133 Mbytes/sec
Transfer mode	Serial	PIO
Connector type	7-pin SATA	40 pin, 80 conductor IDE
Maximum cable length	1 meter	0.45 meter
Power Cable	SATA Cable	IDE Power cable
Signal Voltage	250 mV	3.3 -5 V

0	Features	Command	C R R Reserved (0)	FIS Type (27h)
1	Dev/Head	Cyl High	Cyl Low	Sector Number
2	Features (exp)	Cyl High (exp)	Cyl Low (exp)	Sector Num [exp]
3	Control	Reserved (0)	Sector Count [exp]	Sector Count
4	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)

Figure 2: Register Host to device Frame Information Structure (FIS) layout (3)

PROPOSED DESIGN

Design Overview

Our proposed design as shown in Figure 3 comprises of the CRC, DMA controller, scrambler, 8b/10b encoder and decoder. The physical serial transceivers which are required have been omitted from the design as it is proposed that these be implemented using external SERDES controllers. This was partly influenced by the lack of availability of the appropriate FPGAs at the time of development, and to permit the design to be truly generic, allowing it to be implemented in multiple FPGA devices and not just those with suitable Serial Phys.

SATA control

The Serial ATA controller may be controlled by one of two transfer mechanisms, Parallel Input/Output (PIO) or through a DMA controller. The use of the PIO register block ensures that the Serial ATA controller is backward compatible with Parallel ATA devices. This remains the oldest method to communicate with hard disk drives to be implemented, although it is still used as the default access mechanism when booting a computer for the first time. Whilst the use of the PIO registers is permitted they severely limit the data transfer rate which may be achieved.

DMA and the faster Ultra-DMA communication standards have been the de facto means of communication between disks and memory. It permits far higher data transfer rates to be supported. The Serial ATA standard supports DMA access through direct inputs from a DMA controller to the Transport layer controller. No DMA controller is defined by the Serial ATA standard, only the expected inputs and outputs are defined. It is therefore up to the vendor to select or design a suitable DMA controller to provide these inputs and respond to the outputs. The Transport layer FSM is controlled directly by the DMA controller for

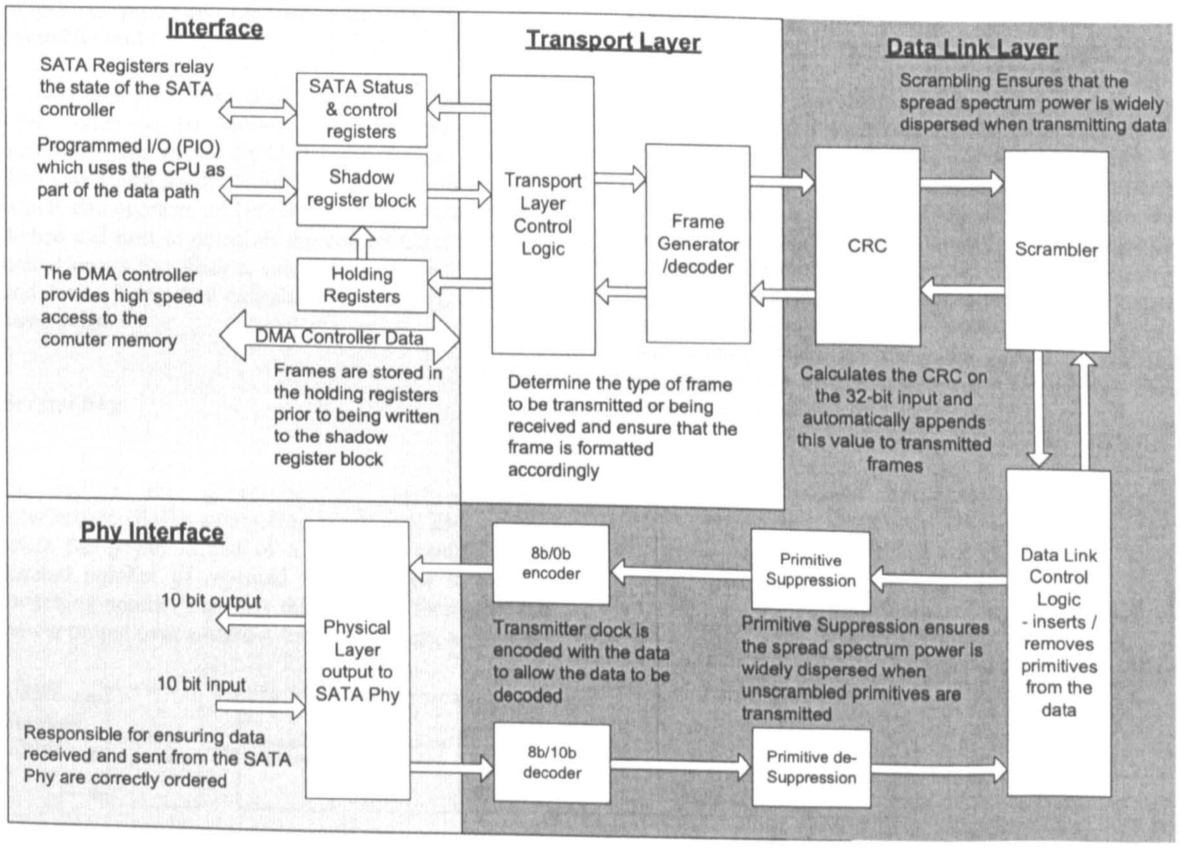

Interface
SATA Registers relay the state of the SATA controller
Programmed I/O (PIO) which uses the CPU as part of the data path
The DMA controller provides high speed access to the comuter memory
SATA Status & control registers
Shadow register block
Holding Registers
DMA Controller Data
Frames are stored in the holding registers prior to being written to the shadow register block
Transport Layer
Transport Layer Control Logic
Frame Generator /decoder
Determine the type of frame to be transmitted or being received and ensure that the frame is formatted accordingly
Data Link Layer
Scrambling Ensures that the spread spectrum power is widely dispersed when transmitting data
CRC
Scrambler
Calculates the CRC on the 32-bit input and automatically appends this value to transmitted frames
Phy Interface
10 bit output
10 bit input
Physical Layer output to SATA Phy
Responsible for ensuring data received and sent from the SATA Phy are correctly ordered
8b/0b encoder
Transmitter clock is encoded with the data to allow the data to be decoded
8b/10b decoder
Primitive Suppression
Primitive Suppression ensures the spread spectrum power is widely dispersed when unscrambled primitives are transmitted
Primitive de-Suppression
Data Link Control Logic - inserts / removes primitives from the data

Figure 3: Proposed System Design for Serial ATA on an FPGA

such communication, with the input and output data most likely stored in first-in-first-out (FIFO) buffers.

CRC

Cyclic Redundancy checking (CRC) allows errors within frames to be detected. The CRC of a frame is a Dword (32-bit) field which follows the last Dword of the contents of an FIS and precedes the end of frame (EOF) primitive. The CRC calculation is performed between the SOF and EOF primitives and covers all the FIS data between these primitives and prior to encoding for transmission (scrambling).

The CRC is calculated using Dwords. Therefore if an FIS contains an odd number of words (16-bits) it should be padded with zeros to produce a Dwords before being used in the calculation. The CRC is calculated using the following polynomial:

$$G(X) = X^{16} + X^{15} + X^{13} + X^4 + 1. \qquad (2)$$

The CRC is initialized with the value 52325032h before the calculation begins. Using such a design would require the CRC and scrambler units to operate at 8 times the clock speed of the rest of the system. Therefore an alternative design which utilises parallel LFSRs is proposed for use with the CRC and Scrambler units.

A parallel implementation of the CRC unit allows the CRC value to be calculated on Dword (32-bit) quantities. This allows the Link layer clock to be set at the same speed as the input from the Transport layer which also operates on Dword quantities. To allow the device and host to calculate the correct CRC the CRC unit is preset to a known value to ensure that the host and device being their calculation of the CRC from the same point.

Scrambling

To reduce the generation of electro-magnetic interference (EMI), scrambling is utilised. EMI occurs when the power output of a device is centred on a limited number of repeated transmissions at a high switching speeds. This has the effect of focussing the power output over a narrow frequency range, as shown in figure 6, which can damage or interfere with devices at the receiving end. Scrambling or spread spectrum techniques, are employed by Serial ATA to remove the high frequency components from the transmitted signal and so spread the effective power over the spectrum. This is shown in figure 7.

There are two scramblers used in Serial ATA. One scrambles the payload data and the other is used to suppress repeated primitives. All Data characters between the SOF and EOF must be scrambled prior to transmission. The scrambling of all the data characters is performed on Dwords by XORing the data to be transmitted with the output of a linear feedback shift register (LFSR). The LFSR implements the following polynomial:

$$G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1. \qquad (1)$$

The shift register is initialised to the hexadecimal value 0xFFFF prior to the first shifted output and before the SOF primitive.

The suppression of repeated primitives is carried out twice in a row. The CONT (continue) primitive is transmitted followed by random data. Any data following the CONT primitive is ignored by the receiver as it is assumed to be the same as that already received.

The scrambler unit also performs its operation in parallel on Dword quantities for the same reasons as stated for the CRC unit. The scrambling of data is performed by generating a pseudo-random value and XORing it with the data to be scrambled. The registers which store the feedback data for the scrambler generator unit are preset prior to scrambling. Passing the scrambled data back through another instance of the preset scrambler allows the scrambled data to be descrambled. Therefore the same design is used for both scrambling and descrambling with only the data input to the scrambler unit differing.

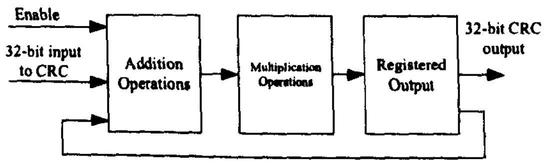
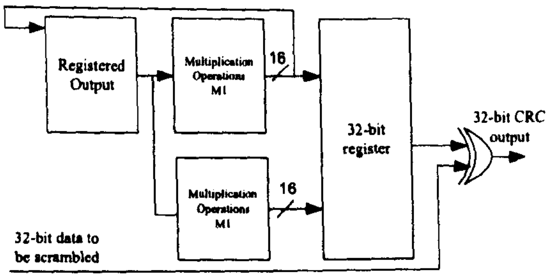


Figure 4: CRC unit design


Registered Output
Multiplication Operations MI
16
Multiplication Operations MI
16
32-bit register
32-bit CRC output
32-bit data to be scrambled
Figure 5: Diagram Scrambler/ Descrambler unit design

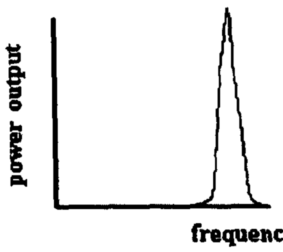


Figure 6: Radiated power output without spectrum techniques applied

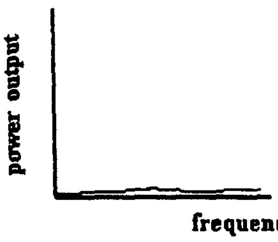


Figure 7: Radiated power with spread spectrum techniques applied

Encoder & Decoder

The 8b/10b encryption is used to encode the clock and data information together. This allows the clock to be extracted during data reception, removing the need to transmit clocking information with the data. Both the encoder and decoder are based upon the original design developed by Widmer and Franaszek (4), as described in their original paper. The original Widmer and Franaszek 8b/10b encoder and decoder used twelve control characters, but Serial ATA 8b/10 encoders use a subset of only four of these control characters.

Designs for both the encoder and decoder are shown in Figs 8 and 9. Both operate at a clock rate four times faster than that of the rest of the Serial ATA controller. Their maximum clock rates are fixed by the Serial ATA specification to an upper limit of 150MHz.

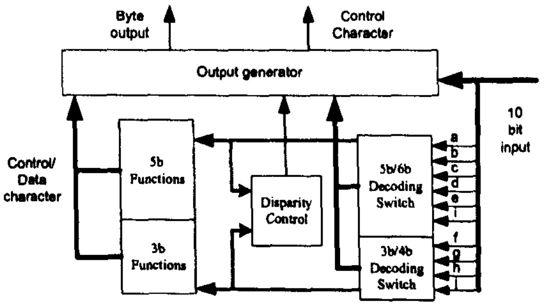


Figure 8: The 8b/10b encoder (pipeline stages and clock lines omitted for clarity)

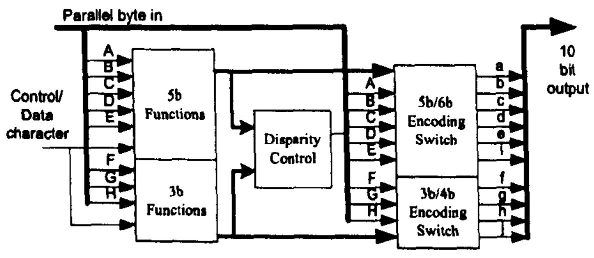


Figure 9: The 10b/8b decoder (pipeline stages and clock lines omitted for clarity)

FUNCTIONAL TESTING

Testing was performed using Verilog testbenches running on ModelSim SE. Simulations were run at both module and system level to verify functional operation of the design. For the CRC, scrambler and primitive suppressor modules, self checking testbenches were generated to test all possible combinations of input and output. These modules were shown to operate correctly in response to all possible input combinations.

A system level testbench was generated to verify operation of the design in response to PIO and DMA transactions. By passing the generated output back through the design, the generation of the serial output and Dword length data was verified to operate correctly. Whilst the results which were generated in this way showed the device to operate correctly, further testing in hardware is required to verify the design's operation.

SYNTHESIS & TIMING ANALYSIS

Throughout the design process Synplify Pro 6.4 was used to synthesise the design, targeting the Altera Stratix EP1S25 FPGA, which was always selected as the default target for all synthesis and place and route tools which were used. FPGA selection was determined by the sponsors, and was selected to meet given cost, performance and availability of the devices at the time the project was commencing. Place and route along with timing analysis were performed using Quartus II version 3 from Altera.

Having run the place and route tool on the synthesised link layer design, timing analysis of the resulting layout showed that both the encoder and decoder did not meet the timing constraints specified in the constraints file. Further, the top speed for both the encoder and decoder designs was less than that required by the Serial ATA standard. As a result of this failure to meet the timing requirements of the design both the encoder and decoder code were modified to incorporate a more pipelined architecture. As a result of this modification to the design, and having re-synthesised the design and processed the new circuit design through Quartus all of the components of the Link layer were found to meet the relevant timing constraints.

The data Link layer, physical control logic and the DMA interface all successfully passed synthesis and timing analysis. We were unable to test and verify the design in hardware due to the need for a suitable Gigabit Serial Phy. As a result, testing was carried out through the use of a testbench. This tested the devices ability to handle both PIO, and simple DMA read and write operations and verified that the device followed the Serial ATA specification with regard to data transfer.

As the aim of the project was to produce a generic design of a Serial ATA controller for FPGA targets, it was decided to target the design at devices from both Altera and Xilinx to determine whether the design could be readily transferred between manufacturers and devices. A summary of the area reports is given below in Table 2. All the results shown here were generated using Synplify Pro 7.3.

TABLE 2 - Comparisons of device area usage between devices

Device Manufacturer	Device	Percentage Area Usage (%)
Xilinx	Virtex 2pro XC2VP7 – 7	21
Xilinx	Virtex 2 CS144 -6	74
Altera	Stratix EP1S25 -7	11
Altera	Cyclone EP1C3 -7	47

As can be seen in Table 2 above, the preferred target device requires a total utilisation of only 11%. This leaves ample space on the FPGA for implementation of multiple other IP blocks that will use the SATA controller.

All of the devices in Table 2 met the timing requirements during post synthesis timing analysis. During the timing analysis process, it was noted that better timing performance was found using the Altera devices. However, improvements in performance would be expected using the Xilinx toolchain. In each of the cases given in Table 2, the Altera devices offered approximately a 10% higher maximum clock rate over the Xilinx devices. Further checks were not carried out as the Altera devices were the preferred choice.

CONCLUSIONS

A fully simulated implementation of a Serial ATA host controller has been implemented and shown to meet the timing constraints necessary to permit the device to operate on an Altera Stratix device. Through simulation and timing analysis it has been shown that this device complies with the Serial ATA standard for the implementation of a Serial ATA controller.
The implemented design is currently undergoing further verification and hardware testing to verify operation prior to making the IP available commercially. Future developments of the IP will follow the Serial ATA roadmap with the design and development of second and third generation devices in the future.

Whilst the simulated testing which was performed has shown the design to operate correctly, implementation and testing in hardware is still necessary to prove this. The Serial ATA host controller has however been fully synthesised, with an appropriate DMA controller which supports PIO transfers and simple memory read write operations, which should allow the design to be tested in hardware relatively quickly and easily once the necessary SERDES are added. In addition to this, improvements to the design may yet be made to improve the timing and layout results to permit the design to be fitted to a smaller area of the device. The design may also be implemented as one IP block in a SoC at some point in the future,

ACKNOWLEDGEMENTS

We would like to thank A2E Ltd for both their financial assistance and technical support in this research. The authors would also like to thank the financial support of the Engineering and Physical Sciences Research Council in the UK

REFERENCES

1. Silicon Image Inc.: Serial ATA for RAID and Enterprise Applications. Silicon Image (2003)

2. Hartney, M.: Implementing Serial ATA Technology. Presented at WinHEC March 22nd (2002)

3. Serial ATA Working Group.: Serial ATA: High Speed Serialized AT Attachment. Revision 1.0a (2003)

4. Widmer, A.X & Franaszek, P.A.: A DC-Balanced, Partitioned Block, 8b/10b Transmission Code. IBM Journal of Research & Development, vol 27 No. 5, September 1983, pp440-451

FPGA based RAID 6 hardware accelerator

Michael Gilroy
ISLI
Alba Centre,
Livingston, UK, EH54 7EG
+441506463393
mgilroy@a2etech.com

James Irvine
Strathclyde University
George St
Glasgow UK
+441415484072
j.m.irvine@strath.ac.uk

William Berrie
A2E Ltd.
Adaptive House
Livingston, UK, EH54 6AX
+441506463393
wberrie@a2etech.com

ABSTRACT

Hard disk storage capacity has continued to rise whilst at the same time the cost per megabyte continues to fall. This, combined with increased usage of digital storage for documents, photography and video for both home and business use has led to increased need for reliable data storage system. Redundant arrays of inexpensive disks (RAID) have proven to offer the best characteristics for reliable storage. However, to date RAID based systems have been limited by their support for only single disk erasure tolerance.

This paper introduces an efficient hardware RAID 6 controller on an FPGA capable offering support for uninterrupted access during double disk erasures and recovery.

Categories and Subject Descriptors

B.m [**Hardware Miscellaneous**]: Applications: uses of FPGAs to achieve high performance

General Terms

Algorithms, Performance, Design.

Keywords

RAID 6, redundancy, FPGA, hardware acceleration.

1. INTRODUCTION

The volume of data stored digitally on hard disk drives, and the capacity available from such drives has increased rapidly in recent years. This has been fuelled by the increased popularity of MP3 players, digital photography and recording. This data is commonly held on a single hard disk drive with irregular if any backup being made. Should the hard disk fail, the user loses all their data. In businesses whilst data storage and backup policies may be implemented these tend to offer slow recovery in the event of multiple disk failures. Both personal and business users need a cheap, reliable, fast and easy to use system to provide data recovery from disk failures. Currently many users utilise redundant arrays of independent disks (RAID) based systems to provide this performance.

RAID based systems can increase the redundancy of the system to provide the capability to recover data from a disk which no longer operates (catastrophic failure) or the recovery of lost data due to a single unrecoverable read error. Catastrophic failures occur infrequently in modern hard disk drives with the mean time to failure for a single disk of around 1.4 million hours for the highest reliability drives [10]. However, the probability of an unrecoverable read access from a disk is far greater, and increases with the number of disks in the array and the capacity of the disks, as shown in figure 1. Although arrays of over 16 drives are uncommon, there is a clear relationship between the disk capacity, array size and the probability of an unrecoverable read error. As disk capacities increase so too will the probability of a read error. Lower quality drives have an increased probability of failure of several orders of magnitude and for storage capacities of over 1TB will in all probability result in a read error every time the array has to be rebuilt [11]. Should a disk fail the standard RAID based systems will fail completely if a read error occurs during the rebuild process.

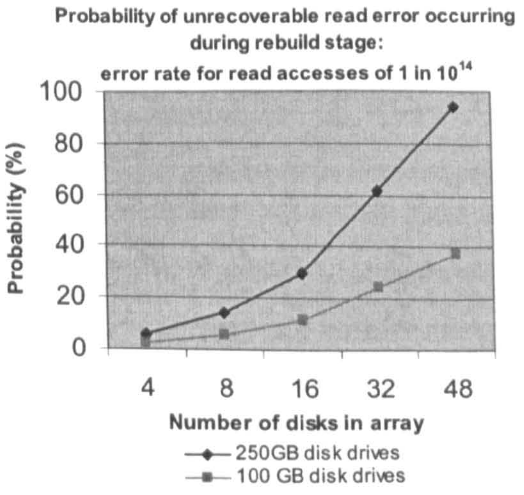


Figure 1. The probability of an unrecoverable error increases with both disk size and the number of disks in the RAID array.

With the continued increase in disk capacities and increased probability of a read error standard RAID based systems may no longer provide sufficient reliability to recover successfully from catastrophic disk failures.

Support for the recovery from up to two catastrophic failures, two read errors or a single catastrophic failure and a read error can be provided through the use of a storage system implementing a RAID 6 algorithm. The technology to provide this additional protection has been available for a number of years, however the cost, proprietary nature of the implementations and IO performance have limited the uptake of these systems. In larger array systems more sophisticated solutions have been developed to improve system performance, however, the cost and complexity of such systems preclude them from consideration of smaller businesses and consumers.

We introduce a hardware based RAID 6 controller implemented on an FPGA to improve the performance when compared to a software only solution, reduce cost and improve reliability.

2. RAID

RAID provides the means to combine multiple hard disks in a single logical unit to offer high availability, performance or a combination of both. This provides better resilience and performance than a single disk drive.

The benefits of RAID include:

- Protection against data loss.
- Provision of real-time data recovery with uninterrupted access due to drive failure or recovery.
- Increased system uptime and network availability.
- Multiple drives working in parallel increases system performance.

2.1 Standard RAID Algorithms

Six RAID levels were originally proposed [12]. Of these RAID levels 0, 1 and 5 are the most commonly implemented. RAID 0 or striping does not provide recovery for data should a drive fail but does improve read/write access speeds by distributing data across the discs. RAID 0 is normally combined with RAID 1 (mirroring) to gain the benefit of improved read/write access times with the advantage of adding the ability to recover lost data should a drive fail. The efficiency of RAID 1 is poor as you need one mirror disk for each data disk. RAID 5 offers the best performance to cost ratio, requiring one disk for checksum data and two or more data. The RAID 5 checksum is a parity calculation making generation and recovery possible by a simple field of XOR gates.

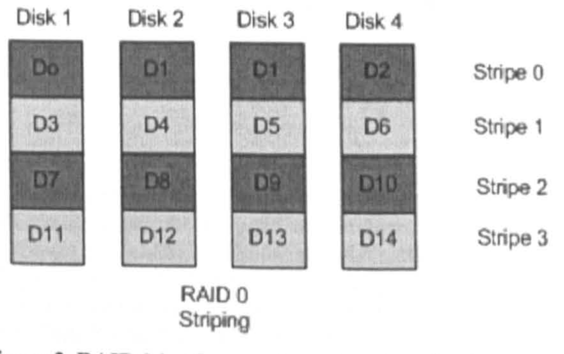


Figure 2. RAID 0 implementation. Data is striped across the four disk array to improve I/O performance.

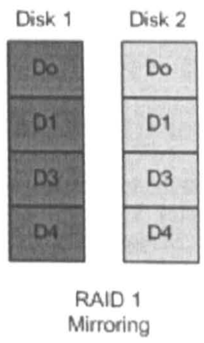


Figure 3. RAID 1 implementation. Data is written onto both disk drives.

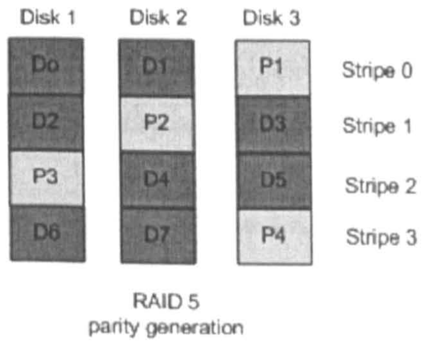


Figure 4. RAID 5 implementation. Data and associated parity are striped across the drives. This is the smallest RAID 5 array type with the data and parity locations being rotated on each stripe.

The need for improved RAID performance was originally tempered with the financial cost of such systems. RAID 5 required the lowest outlay in purchase cost due to the requirement for only a single extra drive. However, as hard disk drive prices have fallen and capacity increased so to has the amount of data stored on such devices. Whilst the mean time to failure (MTTF) of disk drives has improved, with less than 1% of disk drives failing a year [11], there is an increased risk of data loss due to unrecoverable read errors due to the increased storage capacities.

RAID 5 has also benefited from low cost easily implemented hardware accelerators. These have been incorporated onto motherboards and add-in cards for PC systems and provided a fast and efficient solution of RAID storage. The cost of the additional hardware requirements necessary to implement the more complex RAID 6 algorithms long prohibited the widespread adoption of this technology. Low cost FPGAs such as the Altera Cyclone or Xilinx Spartan series offer the means to rapidly develop and implement improved hardware accelerated RAID algorithms.

2.2 RAID 6

RAID 6 has been loosely defined as offering support for the recovery of any 2 disk erasures. RAID 6 increases the reliability of a RAID system through its ability to recover data with up to 2 disk failures without any downtime. There have been a number of algorithms proposed for the implementations of RAID 6 including, EVEN-ODD encoding [4] and Reed-Solomon (RS) coding [6].

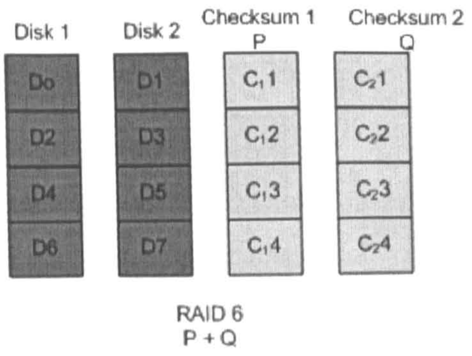


Figure 5. RAID 6 RS based design. Shows the smallest array size with two data disks and two checksum disks.

We propose an RS encoding scheme for RAID 6 operations and optimise it for implementation in an FPGA based hardware accelerator. We are concerned with the development of a RAID 6 hardware accelerator which offers performance better than that achievable in software in terms of CPU utilisation and rebuild speed. Furthermore, the reconfigurability of FPGAs provide the means to easily adapt the system design to cope with possible future standardisation of RAID level 6 [8].

2.3 The need for RAID 6

Whilst the risk of double disk failure remains low for small arrays, should a single drive fail it is becoming increasingly likely that an unrecoverable read error will occur during the rebuild phase. This is particularly true for lower reliability, cheaper hard disk drives. Whilst fully functioning, a RAID 5 array can recover from an unrecoverable read error on one drive, or a single catastrophic disk failure. RAID 5 copes with one simultaneous failure. Any failure from that point until the completion of the rebuild will cause a failure of the array. Should an unrecoverable read error be found during the rebuild stage, the stripe will be lost and may only be recovered by returning the system to the last known good backup, assuming one was made [9].

RAID 6 can recover from a bad block failure and a single drive failure simultaneously increasing the likelihood of a successful rebuild. Just as RAID 5 offered the best performance to cost ratio for single disk erasures, RAID 6 offers the best performance to cost ratio for double disk erasures.

2.4 Reed-Solomon coding

Reed-Solomon coding [12] adds check digits (c_i) calculated over a Galois field from a set of data digits (d_j) such that:

$$c_i = \sum_{j=1}^{n} d_j \times f_{ij}$$

Galois field arithmetic is used to calculate the missing data with n checksums required to solve for n unknowns, i.e. n checksums can correct n erasures. Addition and subtraction operations are both performed as XOR operations. Multiplication and division are more complex and are calculated using logarithms. Unlike regular logarithms, the log of any non-zero element of a Galois Field is an integer, this allows exact multiplication and division of Galois Field elements. Multiplication is calculated as:

$$a \times b = i\log(\log(A) + \log(B))$$

Division is calculated as:

$$\frac{a}{b} = i\log(\log(A) - \log(B))$$

For our implementation a Galois Field of GF(2^4) is proposed. This results in a small logarithm table suited to a small, easily replicated lookup tables. This structure is compact and easily implemented on an FPGA and can be readily scaled to larger array structures.

If the lost data is a checksum, the checksum can be recalculated from the existing data.

$$c_1 = d_i \oplus d_j$$
$$c_2 = (d_i \otimes i) \oplus (d_j \otimes j)$$

If a single error occurs, calculate the c1 (parity), setting the missing data to 0. Missing data is calculated by subtracting the new parity from the original XOR.

If there are two missing items calculate s1 and s2, the modified checksums from setting the missing data to 0 then:

$$c_1 \oplus s_1 = d_i \oplus d_j$$
$$c_2 \oplus s_2 = (d_i \otimes i) \oplus (d_j \otimes j)$$

The missing data is calculated from:

$$(d_i \otimes i) \oplus (d_j \otimes j) = c_1 \oplus s_2 \otimes i$$
$$\Rightarrow (d_i \otimes i) \oplus (d_j \otimes j) = (c_2 \oplus s_2) \oplus [(c_1 \oplus s_1) \otimes i]$$

and:

$$(di \otimes i) \oplus (d_j \otimes j) = c_2 \oplus s_2$$
$$\Rightarrow (c_2 \oplus s_2) \oplus \frac{(c_1 \oplus s_1) \otimes i}{i \oplus j} \quad (9)$$

3. System Architecture

3.1 RAID 6

The RAID 6 algorithm has been manipulated such that the accelerator hardware requirements are reduced to a series of XOR gates, lookup tables and flip-flips. Using a Galois Field of 2^4 reduces the hardware requirements and simplifies the design of the encoder and decoder blocks.

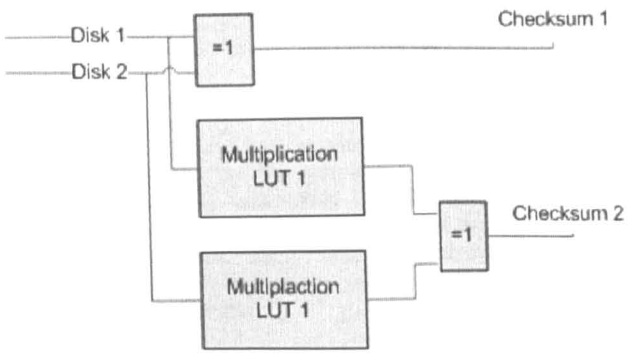


Figure 6. RAID 6 encoder logic.

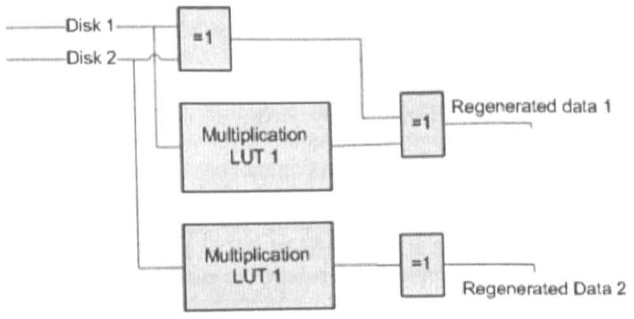


Figure 7. Decoder logic.

These encoder and decoder blocks can be instantiated repeatedly to support larger or smaller data bus paths. For our test system we operated using a 32-bit data path which required 8 encoder and decoder blocks. The encoder and decoder blocks operated at clock speeds in excess of 250MHz for Altera Cyclone devices, with a latency of 3 clock cycles. Our encoder/decoder block for a 32-bit wide data path required only 564 logic cells to implement in the Altera Cyclone device family.

3.2 Test System Design

To test the performance of the RAID 6 hardware accelerator, it was implemented on an Altera Stratix EP1S25 based PCI bus expansion card controlled by a simple DMA controller. This PCI based accelerator card was added to a PC based test system to determine the performance of our hardware solution compared with the existing RAID 6 software solution. Our test platform consisted of; a serverclass computer running Linux kernel 2.6.9 under Fedora Core 3, an AMD Opteron processor, 1 GB of RAM, 1 parallel ATA drive for the operating system and 4 Serial ATA drives for the RAID array.

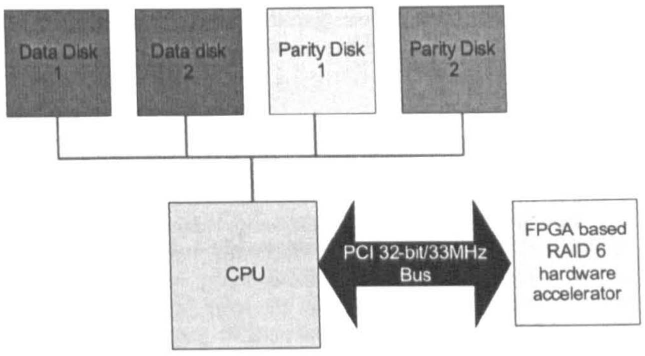


Figure 8. System design

Our hardware accelerator was controlled via a modified revision of the Linux RAID 6 driver which replaced the standard software algorithms for calculating data and checksum values with calls to the hardware controller.

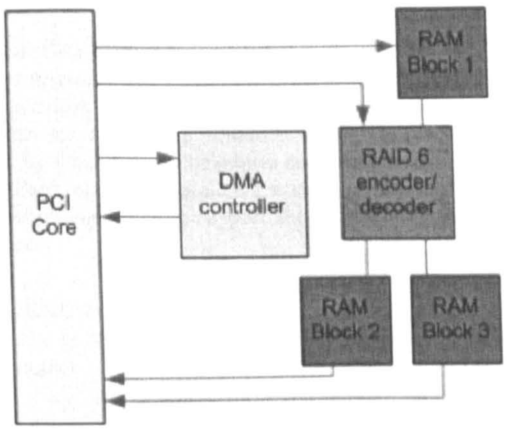


Figure 9. RAID 6 hardware acceleratator

The first disk worth of data is loaded into a RAM block, when the second disks data is read; the checksum or lost data is calculated using the encoder/decoder blocks. Data is loaded into the output RAM blocks and written when the RAM blocks are full or all the data has been processed.

4. Hardware Verification

To verify that the hardware accelerator operated correctly within our test system a RAID 6 array was generated and verified using the Linux mdadm tools [5] and our modified RAID 6 device driver. Using the hardware accelerator this we were successfully able to:

- Generate a RAID 6 array
- Build a Linux File system
- Mount the RAID 6 array
- Read and write file to the array

Verification that the data written successfully to the drive was performed by running a diff on the written file and the original. In this and all other tests each of the four disk drives utilised 10 GB partitions, offering 20 GB of usable data storage space.

With the RAID 6 array functioning correctly under normal circumstances, each disk was removed and rebuilt separately. Files were written to and read from the array whilst each disk was missing (operating in the degraded mode) and during the rebuild stage. Once the array was rebuilt the files on the array were compared with the original files to verify that the data had been successfully recovered. These steps were repeated with each combination of double disk erasure. Again the array successful re-generated the missing data.

5. Analysis

Comparison of the hardware accelerator performance was made with software RAID 6 which is now incorporated into the Linux 2.6 kernel. As can be seen from Table 1 the hardware accelerator currently matches the performance of the software implementation for building the RAID array and the recover of a single disk erasure. However, during double disk erasures the hardware accelerator out performs the software. This is due to the software algorithm recovering data one disk at a time requiring double the number of read/write transactions. The hardware as it is able to calculate the missing data for both failed drives simultaneously and so the rebuild speed is reduced only by the time required to write the second disks worth of data. It is to be expected that the performance of the hardware accelerator will increase with an increase in the bus speed and data path width.

Table 1. Comparison of Software and hardware RAID 6 performance.

	Software RAID 6	Hardware RAID 6
Array generation speed	12 MB/sec	12 MB/sec
Data recovery – 1 erasure	12 MB/sec	12 MB/sec
Data recovery – 2 erasure	7 MB/sec	11 MB/sec

Whilst building the previous tests showed the RAID 6 algorithm operated successfully under all single and double disk erasures, true RAID performance is more commonly measured using benchmarking software [8, 10], we selected BONNIE due to it's ease of use and natively designed for Linux [9]. Bonnie performs a series of tests on a file of known size, performing millions of read/write operations to the array to provide information on CPU utilisation, and access speeds during block read/write accesses and random access across the device.

To ensure that the data could not be held in RAM a 2GB file was used by BONNIE during testing.

As can be seen from Table 2. the hardware generally offered comparable access speeds for sequential outputs. However the performance during double disk erasures is vastly improved when using the hardware accelerator. In addition to this, the hardware accelerator was able to rebuild the lost drive whilst BONNIE was running at approximately 1.8 MB/sec compared with a rebuild speed of 1 MB/sec for software RAID 6.

For sequential input accesses again the hardware RAID 6 far out performed the software RAID 6 during double disk erasures. The only exception to this was during random seeks which offered a poor response from both software and hardware.

6. Future work

Our RAID 6 hardware accelerator is currently limited by the PCI bus transfer rate and the hard disk drive being located on a separate bus from the RAID 6 controller. Moving the RAID 6 accelerator onto higher speed buses such as PCI-X or PCI-express should offer higher bandwidth for data transfers. This should provide improved performance and enable the controller to support larger array sizes. By adding the SATA controller to the hardware accelerator all transactions to the hard disk drives will be performed via the RAID 6 controller offering improved read/write performance and reducing CPU utilisation. Finally differing Reed-Solomon implementations may be generated to test the performance and limitations of each in such a system.

Our current system does not implement disk striping (RAID 0). An investigation into the use of disk striping may be carried out in the future, however as the P+Q data is written as part of the stripe, we expect the benefit of this approach to be negligible.

7. Conclusion

We have demonstrated a working RAID 6 controller on an FPGA which provides performance equivalent to or better than that achievable by a software solution. RAID 6 is currently being adopted by a number of companies and may in the near future be standardised, our solution allows systems to be developed today and be easily upgraded to support any required standardisation in the future.

8. Acknowledgement

Our thanks to A2E Ltd. and the EPSRC for funding and support in this project.

9. References

[1] AC&NC, URL http://www.acnc.com/benchmarks.html, Last Accessed 12th September 2005

[2] Bray, T. *BONNIE user manual*, URL http://www.textuality.com/bonnie/intro.html, Last accessed 12th September 2005

[3] N. Brown, http://neilb.web.cse.unsw.edu.au/source/mdadm/

[4] M Blaum et al. *EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures*, IEEE Transactions on Computers, Vol 44, No 2, PP 192 - 202, February 1995.

[5] IO Zone, URL: http://www.iozone.org/, Last accessed 12th September 2005.

[6] M.H. Jing et al. *A fast error and erasure correction algorithm for a simple RS-RAID*, 2001, pp333-338

[7] Karp, M: *All about RAID*, Network World Newsletter, 7th July 2005, URL available: http://www.networkworld.com/newsletters/stor/2005/0704stor2.html
Last accessed 22nd September 2005.

[8] Paul Luse, Mark Schmisseur, *Understanding Intelligent RAID 6*, Technology@Intel magazine, May 2005

[9] Maxtor Corporation, *Atlas 15K II SAS datasheet*, 2005 Maxtor Corporation.

[10] Maxtor Corporation, *DiamondMax 10 datasheet*, 2005 Maxtor Corporation.

[11] D.A. Patterson, G.A. Gibson, R.H. Katz, *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, Proceedings, ACM SIGMOD Conference, pp. 109116, July 1988.

[12] I. S. Reed and G. Solomon. *Polynomial codes over certain finite fields*. J. SIAM, 8(2):300-304, June 1960

[13] Wicker S., Bhagarv, V., *Reed-Solomon codes and their application*, IEEE press, 1994.

Table 2. Comparison of BONNIE results between hardware and software RAID 6 sequential outputs and inputs.

Mode	Sequential Output						Sequential Input					
	Per Char		Block		Rewrite		Per Char		Block		Rewrite	
	MB/ sec	% CPU	MB/ sec	% CPU	MB/ sec	% CPU	MB/ sec	% CPU	MB/ sec	% CPU	MB/ sec	% CPU
Software normal operation	14226	33.7	21330	4.7	14314	18.8	6780	24.5	31522	33.9	244.3	3.3
Hardware normal operation	14815	30.3	21528	4.1	13683	16.1	6745	24.4	32353	34.4	244.1	3.0
Software Single disk erasure	11103	29.8	18297	6.5	14087	17.5	6236	22.6	28225	32.4	223.3	4.1
Hardware Single disk erasure	12020	24.7	19996	4.4	13508	15.8	6316	22.9	28186	28.1	268.6	3.0
Software Single disk erasure	6089	14.5	20649	4.6	5883	7.2	5532	20.0	7800	28.6	256.9	3.9
Hardware double disk erasure	10577	24.7	19736	4.3	12561	14.7	5767	20.9	27709	29.5	257.5	3.3

RAID 6 HARDWARE ACCELERATION

Michael Gilroy

Institute for System Level Integration
Alba Campus, Livingston,
UK, EH54
email: mgilroy@a2etech.com

James Irvine

Mobile Communications Group
University of Strathclyde
Glasgow, UK, G1 1XW
email: j.m.irvine@strath.ac.uk

ABSTRACT

Hard disk storage capacity has continued to rise whilst at the same time the cost per megabyte continues to fall. This, combined with increased usage of digital storage for documents, photography and video for both home and business use has led to increased need for reliable data storage system. Redundant arrays of inexpensive disks (RAID) have proven to offer the best characteristics for reliable storage. However, to date RAID based systems have been limited, due to cost and circuit complexity, by their support for only single disk erasure tolerance. FPGAs allow us to overcome these difficulties and allow support for more complex storage algorithms.
This paper introduces an efficient FPGA based hardware RAID 6 accelerator providing uninterrupted access during all single and double disk erasures and recovery.

1. INTRODUCTION

The volume of data stored digitally on hard disk drives, and the capacity available from such drives has increased rapidly in recent years. This has been fuelled by the increased popularity of MP3 players, digital photography and recording. This data is commonly held on a single hard disk drive with irregular if any backup being made. Should the hard disk fail, the user loses all their data. In businesses whilst data storage and backup policies may be implemented these tend to offer slow recovery in the event of multiple disk failures. Both personal and business users need a cheap, reliable, fast and easy to use system to provide data recovery from disk failures. Currently many users utilise redundant arrays of independent disks (RAID) based systems to provide this performance.
RAID based systems can increase the redundancy of the system to provide the capability to recover data from a disk which no longer operates (catastrophic failure) or the recovery of lost data due to a single unrecoverable read error. Catastrophic failures occur infrequently in modern hard disk drives with the mean time to failure for a single disk of around 1.4 million hours for the highest reliability drives [10]. However, the probability of an unrecoverable read access from a disk is far greater, and increases with the number of disks in the array and the capacity of the disk drives, as shown in figure 1. There is a clear relationship between the disk capacity, array size and the probability of an unrecoverable read error. As disk capacities increase so too will the probability of a read error. Lower quality drives have an increased probability of failure of several orders of magnitude and for storage capacities of over 1TB will in all probability result in a read error every time the array has to be rebuilt [11]. Should a disk fail the standard RAID based systems will fail completely if a read error occurs during the rebuild process.

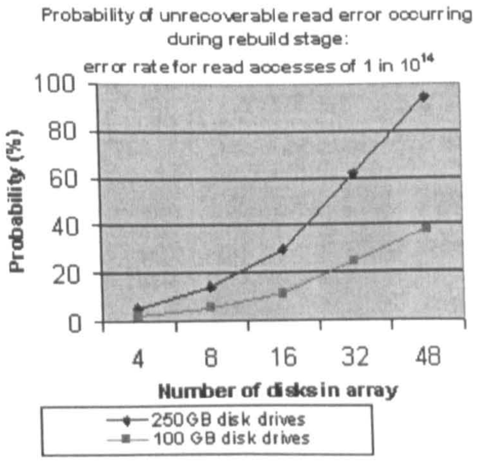


Fig. 1. The probability of an unrecoverable read error for RAID arrays utilising 250GB and 100GB disk drives.

With the continued increase in disk capacities and increased probability of a read error standard RAID based systems may no longer provide sufficient reliability to recover successfully from catastrophic disk failures.
Support for the recovery from up to two catastrophic failures, two read errors or a single catastrophic failure and a read error can be provided through the use of a storage system implementing a RAID 6 algorithm. The technology to provide this additional protection has been available for a number of years. The cost, proprietary nature of the implementations and IO performance, however, limited the uptake of these systems. In larger array systems more

sophisticated solutions have been developed to improve system performance, although, the cost and complexity of such systems preclude them from consideration by smaller businesses and consumers.
The prohibitive cost of RAID 6 controllers in the past have been due, in part, to the increased chip density and circuit complexity of the hardware accelerator and also the high cost of hard disk drives. We resolve both these problems via an FPGA based RAID 6 accelerator utilising low cost Serial ATA (SATA) hard disk drives. The increased reliability of RAID 6 algorithm allows the low cost and lower reliability drives to be used in a reliable system. Although for the system described in this discussion the accelerator was implemented on a Stratix FPGA the accelerator can readily be implemented on lower cost Cyclone II FPGAs reducing the overall cost.

2. BACKGROUND

RAID provides the means to combine multiple hard disks in a single logical unit to offer high availability, performance or a combination of both. This provides better resilience and performance than a single disk drive.
The benefits of RAID include:

- Protection against data loss.
- Provision of real-time data recovery with uninterrupted access due to drive failure or recovery.
- Increased system uptime and network availability.
- Multiple drives working in parallel which increases system performance.

2.1. Standard RAID algorithms

Six RAID levels were originally proposed [12]. Of these RAID levels 0, 1 and 5 are the most commonly implemented. RAID 0 or striping does not provide recovery for data should a drive fail but does improve read/write access speeds by distributing data across the discs. RAID 0 is normally combined with RAID 1 (mirroring) to gain the benefit of improved read/write access times with the advantage of adding the ability to recover lost data should a drive fail. The efficiency of RAID 1 is poor as you need one mirror disk for each data disk. RAID 5 offers the best performance to cost ratio, requiring one disk for checksum data and two or more data.
The RAID 5 checksum is a parity calculation making generation and recovery possible by a simple field of XOR gates.

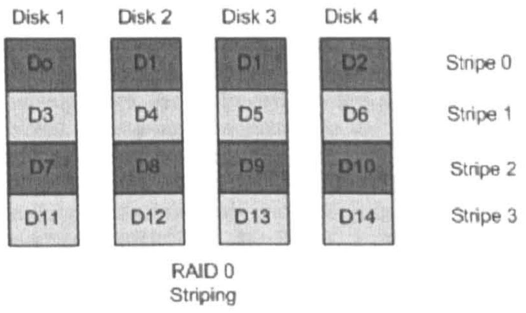


Fig. 2. RAID 0 implementation. Data is striped across the four disk array to improve I/O performance.

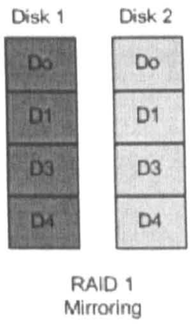


Fig. 3. RAID 1 implementation. Data is written onto both disk drives

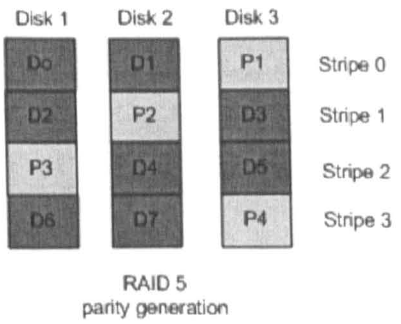


Fig. 4. RAID 5 implementation. Data and associated parity are striped across the drives. This is the smallest RAID 5 array type with the data and parity locations being rotated on each stripe

The need for improved RAID performance was originally tempered with the financial cost of such systems. RAID 5 required the lowest outlay in purchase cost due to the requirement for only a single extra drive. However, as hard disk drive prices have fallen and capacity increased so too has the amount of data stored on such devices. Whilst the mean time to failure (MTTF) of disk drives has improved, with less than 1% of disk drives failing a year [11], there is an increased risk of data loss due to unrecoverable read errors due to the increased storage capacities.

RAID 5 has also benefited from low cost easily implemented hardware accelerators. These have been incorporated onto motherboards and add-in cards for PC systems and provided a fast and efficient solution of RAID storage. The cost of the additional hardware requirements necessary to implement the more complex RAID 6 algorithms has long prohibited the widespread adoption of this technology. Low cost FPGAs such as the Altera Cyclone or Xilinx Spartan series offer the means to rapidly develop and implement improved hardware accelerated RAID algorithms.

3. RAID 6

RAID 6 has been loosely defined as offering support for the recovery of any 2 disk erasures. RAID 6 increases the reliability of a RAID system through its ability to recover data with up to 2 disk failures without any downtime. There have been a number of algorithms proposed for the implementations of RAID 6 including, EVEN-ODD encoding [4] and Reed-Solomon (RS) coding [6].

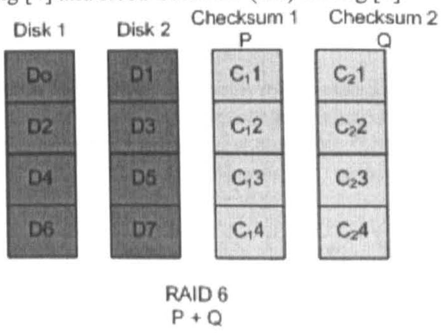


Fig. 5. RAID 6 RS based design. Shows the smallest array size with two data disks and two checksum disks.

We propose an RS encoding scheme for RAID 6 operations and optimise it for implementation in an FPGA based hardware accelerator. We are concerned with the development of a RAID 6 hardware accelerator which offers performance better than that achievable in software in terms of CPU utilisation and rebuild speed. Furthermore, the reconfigurability of FPGAs provide the means to easily adapt the system design to cope with possible future standardisation of RAID level 6 [8].

3.1. The need for RAID 6

Whilst the risk of double disk failure remains low for small arrays, should a single drive fail it is becoming increasingly likely that an unrecoverable read error will occur during the rebuild phase. This is particularly true for lower reliability, cheaper hard disk drives. Whilst fully functioning, a RAID 5 array can recover from an unrecoverable read error on one drive, or a single catastrophic disk failure. RAID 5 copes with one simultaneous failure. Any failure from that point until the completion of the rebuild will cause a failure of the array. Should an unrecoverable read error be found during the rebuild stage, the stripe will be lost and may only be recovered by returning the system to the last known good backup, assuming one was made [9].
RAID 6 can recover from a bad block failure and a single drive failure simultaneously increasing the likelihood of a successful rebuild. Just as RAID 5 offered the best performance to cost ratio for single disk erasures, RAID 6 offers the best performance to cost ratio for double disk erasures.

4. REED-SOLOMON CODING

Reed-Solomon coding [12] adds check digits (c_i) calculated over a Galois field from a set of data digits (d_j) such that:

$$c_i = \sum_{j=1}^{n} d_j \times f_{ij} \tag{1}$$

Galois field arithmetic is used to calculate the missing data with n checksums required to solve for n unknowns, i.e. n checksums can correct n erasures. Addition and subtraction operations are both performed as XOR operations. Multiplication and division are more complex and are calculated using logarithms. Unlike regular logarithms, the log of any non-zero element of a Galois Field is an integer, this allows exact multiplication and division of Galois Field elements. Multiplication is calculated as:

$$a \times b = i\log(\log(A) + \log(B)) \tag{2}$$

Division is calculated as:

$$\frac{a}{b} = i\log(\log(A) - \log(B)) \tag{3}$$

For our implementation a Galois Field of GF(2^4) is proposed. This results in a small logarithm table suited to a small, easily replicated lookup tables. This structure is compact and easily implemented on an FPGA and can be readily scaled to larger array structures.
If the lost data is a checksum, the checksum can be recalculated from the existing data.

$$c_1 = d_i \oplus d_j \tag{4}$$
$$c_2 = (d_i \otimes i) \oplus (d_j \otimes j) \tag{5}$$

If a single error occurs, calculate the c1 (parity), setting the missing data to 0. Missing data is calculated by subtracting the new parity from the original XOR.

If there are two missing items calculate s1 and s2, the modified checksums from setting the missing data to 0 then:

$$c_1 \oplus s_1 = d_i \oplus d_j \qquad (6)$$

$$c_2 \oplus s_2 = (d_i \otimes i) \oplus (d_j \otimes j) \qquad (7)$$

The missing data is calculated from:

$$(d_i \otimes i) \oplus (d_j \otimes j) = c_1 \oplus s_2 \otimes i \qquad (8a)$$

$$\Rightarrow (d_i \otimes i) \oplus (d_j \otimes j) = (c_2 \oplus s_2) \oplus [(c_1 \oplus s_1) \otimes i] \qquad (8b)$$

and:

$$(di \otimes i) \oplus (d_j \otimes j) = c_2 \oplus s_2 \qquad (9a)$$

$$\Rightarrow (c_2 \oplus s_2) \oplus \frac{(c_1 \oplus s_1) \otimes i}{i \oplus j} \qquad (9b)$$

Whilst these calculations are computationally intensive on a standard CPU, they may be reduced to a series of multiplications, divisions and XOR operations on an FPGA.

5. SYSTEM ARCHITECTURE

5.1. RAID 6

The RAID 6 algorithm has been manipulated such that the accelerator hardware requirements are reduced to a series of XOR gates, lookup tables and flip-flips. Using a Galois Field of 2^4 reduces the hardware requirements and simplifies the design of the encoder and decoder blocks.

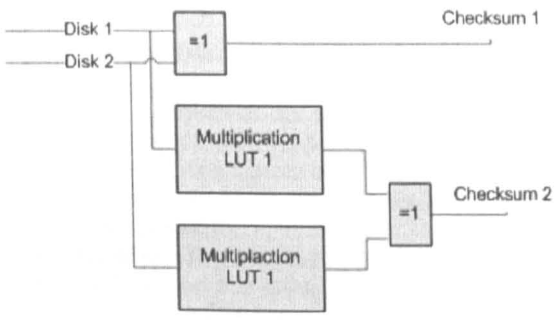


Fig. 6. RAID 6 encoder logic.

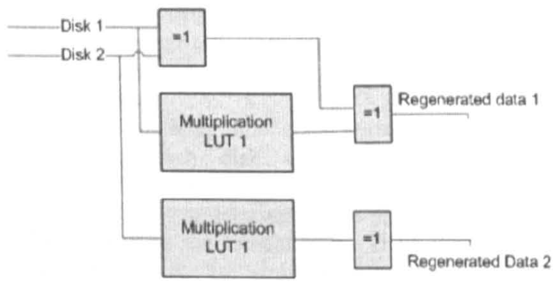


Fig. 7. Decoder logic.

The encoder/decoder blocks operate on a 4-bit wide data bus. This reduces the size of the lookup tables for multiplication and division operations and simplified the design. However, this limits the maximum size of the array to 16 disks, which was considered to be a reasonable trade-off as our test system could only support 8 disks. Minimal changes would be required to alter the algorithm to support larger arrays. To reduce overall resource utilisation, the Reed-Solomon codec shares lookup tables, and control signals, between the encoder and decoder functions.
The encoder and decoder blocks may be instantiated repeatedly to support larger or smaller data bus paths. Our test system operated on a 64-bit data path utilising 16 encoder and decoder blocks. The encoder and decoder blocks operated at clock speeds in excess of 250MHz for Altera Cyclone devices, with a latency of 6 clock cycles. Our encoder/decoder block for a 64-bit wide data path required only 3598 logic cells to implement in the Altera Cyclone device family.

5.2. Test System Design

To test the performance of the RAID 6 hardware accelerator, it was mapped to an Altera Stratix EP1S25 based PCI bus expansion card controlled by a simple DMA controller. This PCI accelerator card was added to a PC based test system to determine the performance of our hardware solution compared with the existing RAID 6 software solution. Our test platform consisted of: a server class computer running Linux kernel 2.6.15.2 on an AMD Opteron processor with 1 GB of RAM; 1 parallel ATA drive for the operating system; and 4 Serial ATA drives with the RAID 6 array.

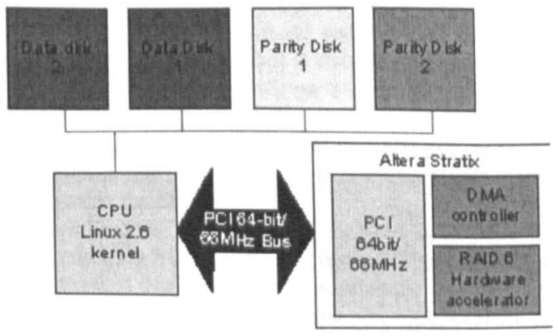


Fig. 8. System design

The hardware accelerator resource utilisation (post fitting) for the entire design was 12729 logic cells (68% utilisation), of which 4281 logic cells were required for the RAID 6 hardware encoder/decoder. The hardware accelerator suffers from a 6 clock latency between processing the data and generating the encoded output. However, as the test system read data in 2Kb chunks over

the PCI bus, this delay in generating the encoded data was inconsequential in this instance. The accelerator was therefore clocked at 66MHz to match the PCI bus clock rate.
Our hardware accelerator was controlled via a modified revision of the Linux RAID 6 driver which replaced the standard software algorithms for calculating data and checksum values with calls to the hardware controller.

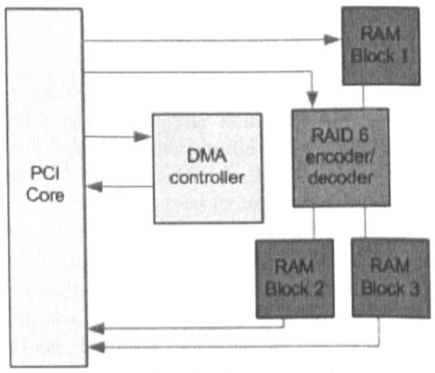


Fig. 9. RAID 6 hardware acceleratator

The first disk worth of data is loaded into a RAM block, and when the second disks data is read, the checksum or lost data is calculated using the encoder/decoder blocks. Data is loaded into the output RAM blocks and written when the RAM blocks are full or all the data has been processed.

6. HARDWARE VERIFICATION

To verify that the hardware accelerator operated correctly within our test system a RAID 6 array was generated and verified using the Linux mdadm tools [5] and our modified RAID 6 device driver. Using the hardware accelerator we were successfully able to:

- Generate a RAID 6 array
- Build a Linux File system
- Mount the RAID 6 array
- Read and write files to the array

Verification that the data written successfully to the drive was performed by running a diff on the written file and the original. In this and all other tests each of the four disk drives utilised 10 GB partitions, offering 20 GB of usable data storage space.
With the RAID 6 array functioning correctly under normal circumstances, each disk was removed and rebuilt separately. Files were written to and read from the array whilst each disk was missing (operating in the degraded mode) and during the rebuild stage. Once the array was rebuilt the files on the array were compared with the original files to verify that the data had been successfully recovered. These steps were repeated with each combination of double disk erasure. Again the array successfully re-generated the missing data.

7. ANALYSIS

Comparison of the hardware accelerator performance was made with software RAID 6 which is now incorporated into the Linux 2.6 kernel. As can be seen from Table 1 the hardware accelerator outperforms the software algorithm in all modes of operation. Although generation of a RAID array and rebuilding a double disk failure would intuitively be expected to operate at the same rate, the differences in performance observed are due to the additional requirements necessitated when building the array. The speed up over a single erasure recovery is due to the need to read an extra disks worth of data. Data throughput was found to be limited by the device driver and control mechanisms used to load data onto the accelerator card. Throughput is expected to be improved by transition to a higher speed bus, however, the PCI bus was not the limiting factor for the four disk array.

Table 1. Comparison of Software and hardware RAID 6 performance.

	Software RAID 6	**Hardware RAID 6**
Array generation speed	17 MB/sec	23 MB/sec
Data recovery – 1 erasure	17 MB/sec	23 MB/sec
Data recovery – 2 erasure	15 MB/sec	25 MB/sec

Whilst the previous tests showed the RAID 6 algorithm operated successfully under all single and double disk erasures, true RAID performance is more commonly measured using benchmarking software [8, 10]. We selected BONNIE++ due to its ease of use, support of multi-gigabyte testing and that it was natively designed for Linux [9]. BONNNIE++ performs a series of tests on a file of known size, performing millions of read/write operations to the array to provide information on CPU utilisation, and access speeds during block read/write accesses and random access across the device.
To ensure that the data could not be held in RAM a 4GB file was used by BONNIE during testing. The results of this testing showed that the hardware accelerator was capable of faster rebuild speeds during high data throughput at the cost of higher CPU utilisation. This is in part due to the operating system being able to perform more background tasks as the device driver sleeps while the hardware is processing data, as opposed to the software only solution which performs all calculations on the processor.
To verify that the CPU utilisation dropped when using the hardware accelerator we repeated the BONNIE++ and using the Linux system monitor, measured the percentage of CPU time consumed by the RAID 6 driver. During single disk erasure, CPU utilisation was found to drop by a

third when compared with the software only driver. This drop increased to one half of the software only solutions CPU consumption during double disk erasures.
The overall performance of the hardware accelerator would be increased through the use of a higher speed bus and providing direct access to the hard disk controller to the accelerator.

8. FUTURE WORK

Our RAID 6 hardware accelerator is currently limited by the device driver. However, on arrays of over 6 disk drives the PCI bus speed currently being utilised will limit data transfer rate, although this should be overcome through the use of the PCI-X bus. Additionally, as array size increases the CPU over head required to setup and run the hardware accelerator increases to unacceptable levels, the optimal solution will therefore be achieved by adding a number of hard disk controller chips onto the same board as the FPGA and giving the FPGA direct access to these controllers. Thanks to our implementation on the FPGA we are well on our way to achieving this goal using an embedded NIOS processor to offload all setup and control features for the RAID 6 accelerator.

9. CONCLUSION

We have demonstrated a working RAID 6 controller on an FPGA which provides reliable data storage and outperforms software based solutions providing higher data throughput and lower CPU utilisation. RAID 6 is currently being adopted by a number of companies and may in the near future be standardised, our solution allows systems to be developed today and be easily upgraded to support any required standardisation in the future.

10. ACKNOWLEDGEMENT

Our thanks to A2E Limited. and the EPSRC for funding and support in this project.

REFERENCES

[1] AC&NC, URL http://www.acnc.com/benchmarks.html, Last Accessed 12th September 2005

[2] Bray, T. *BONNIE user manual*, URL http://www.textuality.com/bonnie/intro.html, Last accessed 12th September 2005

[3] N. Brown, http://neilb.web.cse.unsw.edu.au/source/mdadm/

[4] M Blaum et al. *EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures*, IEEE Transactions on Computers, Vol 44, No 2, PP 192 - 202, February 1995.

[5] IO Zone, URL: http://www.iozone.org/, Last accessed 12th September 2005.

[6] M.H. Jing et al. *A fast error and erasure correction algorithm for a simple RS-RAID*, 2001, pp333-338

[7] Karp, M: All about RAID, Network World Newsletter, 7th July 2005, URL available: http://www.networkworld.com/newsletters/stor/2005/0704stor2.html
Last accessed 22nd September 2005.

[8] Paul Luse, Mark Schmisseur, *Understanding Intelligent RAID 6*, Technology@Intel magazine, May 2005

[9] Maxtor Corporation, *Atlas 15K II SAS datasheet*, 2005 Maxtor Corporation.

[10] Maxtor Corporation, *DiamondMax 10 datasheet*, 2005 Maxtor Corporation.

[11] D.A. Patterson, G.A. Gibson, R.H. Katz, *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, Proceedings, ACM SIGMOD Conference, pp. 109116, July 1988.

[12] I. S. Reed and G. Solomon. *Polynomial codes over certain finite fields*. J. SIAM, 8(2):300-304, June 1960

[13] Wicker S., Bhagarv, V., *Reed-Solomon codes and their application*, IEEE press, 1994.

IP CORE FOR RAID 6 HARDWARE ACCELERATION

Michael Gilroy, James Irvine, Gideon Riddell

Institute for System Level Integration
Alba Centre, Livingston, UK, EH
mgilroy@a2etech.com

University of Strathclyde
Mobile Communications Group Glasgow, UK, G1 1XW
j.m.irvine@strath.ac.uk

A2E Limited
Adaptive House, Livingston UK, EH54 6AX
griddell@a2etech.com

ABSTRACT

As storage requirements and magnetic disk densities increase the need for reliable storage solutions also increase. This IP core, written in Verilog HDL, provides a small and efficient hardware accelerator for performing RAID 6 calculations to provide uninterrupted access to data during single and double disk failures.

In this paper we describe the implementation and verification of a RAID 6 IP block. We present an example system implemented on an FPGA to demonstrate the capabilities of the IP block and verify its operation in hardware.

Keywords – RAID 6 Core, reconfigurable, data reliability

1. INTRODUCTION

The digital storage requirements for both consumer and high end systems continue to increase rapidly year on year. This is being driven by the widespread adoption of digital TV, photography music etc., plus increased legislation on business to retain data over longer time periods.

Coupled with this increase in storage requirements is the need for ensuring availability and reliability of the data and delivering this at as low a cost as possible.

Redundant arrays of independent disks (RAID) have, since the 1980s, provided the means to store and recover lost data efficiently. Of the original RAID levels, RAID 5 provided the optimal solution based upon the cost and reliability for systems with 3 or more hard disk drives. The performance benefits offered by RAID 5 based solutions are slowly being eroded by the increased risk of data loss due to multiple simultaneous disk errors, be they unrecoverable read errors or disk drive failures (erasures).

In this paper we discuss the limitations of RAID 5, the benefits of adopting RAID 6, present a RAID 6 IP block the verification process, and an example RAID 6 based application.

2. RAID OVERVIEW

RAID allows the combination of multiple hard disk drives to provide a combination of one or more of the following characteristics:

- Protection against data loss.
- Provision of real-time data recovery with uninterrupted access due to drive failure or recovery.
- Increased system uptime and network availability.
- Multiple drives working in parallel which increases system performance.

Various RAID levels were proposed by Paterson et al in their 1981 paper [11] describing RAID architectures. Of these, RAID 5 offered the best performance in terms of reliability when compared with the overall system cost, data availability, redundancy overhead and data throughput.

RAID 5 utilises a single redundant disk which contains parity data to allow data recovery for and single disk failure or read error. The parity calculation may be readily implemented in software or in via a simple field of XOR gates in hardware.

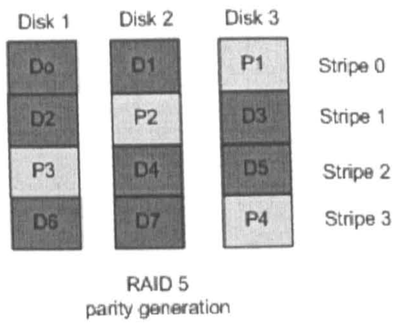


Fig. 1. RAID 5 implementation. Data and associated parity are striped across the drives. This is the smallest RAID 5 array type with the data and parity locations being rotated on each stripe

The need for greater redundancy in array based systems was originally limited by the purchase cost and the low probability of simultaneous multiple disk failures. However, as disk capacities increase and disk arrays increase in density, the likelihood of an unrecoverable read error or multiple simultaneous disk failures occurring increase.

The mean time to failure (MTTF) of disk drives has improved rapidly with less than 1% of disk drives failing a year [11]. However, the probability of an unrecoverable read error occurring whilst restoring a disk array has increased due to the storage capacities now available from hard disk drives.

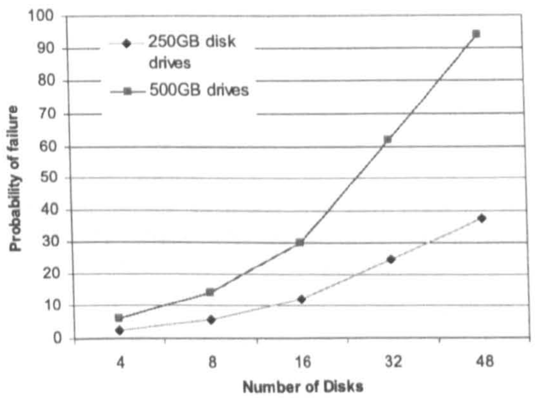


Fig. 2. Shows the probability of a read error occuring during an array rebuilding a single lost disk. Disks have a bit error rate for read accesses of 1 in 10^{14}

Disk arrays using large numbers of disks, or a small number of large disk drives increases the probability of errors occurring in a RAID 5 disk array. The use of double disk redundancy, RAID 6, reduces the risks of this situation occurring.

3. RAID 6

RAID 6 whilst not defined in the original RAID definitions has been loosely defined as offering support for the recovery of any 2 disk erasures. A number of algorithms proposed for the implementations of RAID 6 including, EVEN-ODD encoding [4] and Reed-Solomon (RS) coding [6].

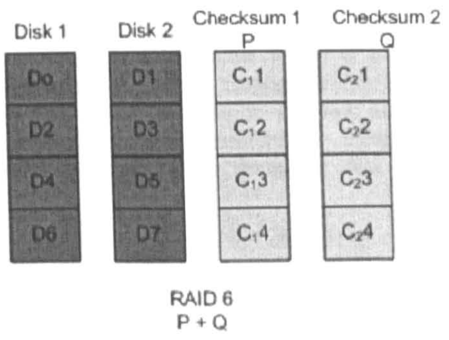


Fig. 3. RAID 6 RS based design. Shows the smallest array size with two data disks and two checksum disks.

Our solution utilises a Reed-Solomon based encoding scheme over a Galois field of 16, which necessitates the use of two redundant disk drives [12]. The use of Reed-Solomon codes instead of one of the other algorithms was determined by the ease of implementation and the similarities between this algorithm and RAID 5. RAID 5 may be considered special case of Reed-Solomon with a redundancy of 1, and therefore the RAID 6 implementation may make use of data striping and other RAID 5 optimisations for updating stripes.

4. RAID 6 IP BLOCK

Our RAID 6 IP block is based upon a pipelined Reed-Solomon encoder and decoder with a controlling state machine and local memory.

Reed-Solomon coding is performed over a Finite or Galois field. Galois field arithmetic is well suited to hardware implementation as the results of all multiplications and division are guaranteed to be real numbers. The use of Galois field arithmetic also makes the algorithm time consuming in software.

All Galois field addition and subtraction is performed by an XOR operation. Multiplication and division may be performed utilising the logarithms and anti-logarithms. These are readily implemented as lookup tables in hardware.

The RAID 6 IP block does not perform error detection on the incoming data stream. Data errors and disk failures are instead indicated by the CRC checks performed by the hard disk drives on all read blocks.

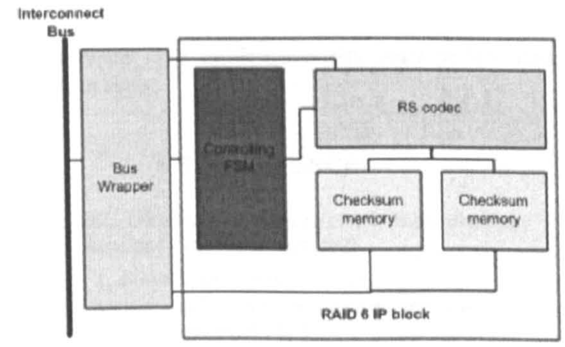


Fig. 4. RAID 6 IP block interconnection. The interconnect bus may be a direct connection to an on-chip device or a Avalon or other bus provided a suitable wrapper is included

Data is read into the RAID 6 IP block from one disk at a time until the checksum memory is filled all or all the data is read. The checksum memory stores the temporary results of the Reed-Solomon calculations. These are fed back as each new disk is read. Once the final disk has been read the calculated checksum(s) may be output to the appropriate location.

The Reed-Solomon coding scheme which has been utilised supports up to 16 disks and requires that data be encoded or decoded 4-bits at a time. By implementing multiple encoder/decoder blocks in parallel various data bus widths may be supported. Our test system verified the operation of both 32-bit and 64-bit wide data paths.

5. VERIFICATION

Verification of the IP block was performed by both simulation and implementation on an FPGA and testing using standard software tools. Simulation of the RAID 6 RTL via a Verilog simulation allowed verification of correct operation of the encoder for all valid combinations of disks arrays from 4 to 16 disks. Simulation also showed proper data recover for all combinations of single and double disk failures.

Synthesis of the design on to a number of Altera FPGAs showed that the RAID 6 IP block was capable of operating at speeds of up to 400MHz using 32-bit and 64-bit wide data paths. Ensuring data availability was found to be the limiting factor to the hardware design.

To verify the hardware implementation provided a speed up over software based RAID 6 algorithms the RAID 6 IP block was synthesised onto an FPGA PCI development platform. This allowed direct comparison of the hardware with software based algorithms.

6. TEST PLATFORM

Our test platform consisted of: a server class computer running Linux kernel 2.6.15.2 on an AMD Opteron processor with 1 GB of RAM; 1 parallel ATA drive for the operating system; and 4 Serial ATA drives with the RAID 6 array. Testing of the RAID 6 IP block was carried out on an Altera Stratix EP1S25 based PCI development card connected to the test platform. The RAID 6 IP block was configured to perform its data accesses via the PCI bus accessing data from main memory.

The standard Linux RAID 6 software driver was modified to support the use of the hardware accelerator on the PCI bus. All software based calculations of the Reed-Solomon coding were replaced with calls to the hardware.

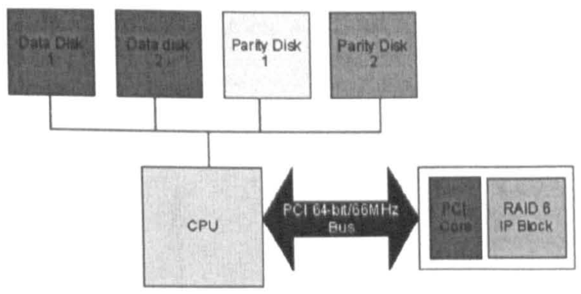


Fig. 5. PCI RAID 6 accelerator based upon the RAID 6 IP block and implemented on an Altera Stratix PCI development board

7. RESULTS & ANALYSIS

Our hardware test platform allowed the hardware system to be tested and compared with existing software only solutions. Standard Linux based benchmarking tools were utilised to determine the data throughput and CPU utilisation. BONNIE++, a common benchmarking tool showed that the hardware performed favourably when compared to the software only implementation and it was noted that there was a significant drop in CPU utilisation, between 33% and 50%, when using the hardware.

8. CONCLUSIONS

RAID based storage solutions provide the means to provide low cost, reliable data storage systems. In this paper we have demonstrated an implementation of a RAID 6 IP block to provide superior data reliability, implemented and tested the IP in an FPGA and shown that this hardware solution provides performance benefits over software only implementations even at low data throughput rates.

9. REFERENCES

[1] AC&NC, URL http://www.acnc.com/benchmarks.html, Last Accessed 12th September 2005

[2] Bray, T. *BONNIE user manual*, URL http://www.textuality.com/bonnie/intro.html, Last accessed 12th September 2005

[3] N. Brown, http://neilb.web.cse.unsw.edu.au/source/mdadm/

[4] M Blaum et al. *EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures*, IEEE Transactions on Computers, Vol 44, No 2, PP 192 - 202, February 1995.

[5] IO Zone, URL: http://www.iozone.org/, Last accessed 12th September 2005.

[6] M.H. Jing et al. A fast error and erasure correction algorithm for a simple RS-RAID, 2001, pp333-338

[7] Karp, M: All about RAID, Network World Newsletter, 7th July 2005, URL available: http://www.networkworld.com/newsletters/stor/2005/0704stor2.html
Last accessed 22nd September 2005.

[8] Paul Luse, Mark Schmisseur, *Understanding Intelligent RAID 6*, Technology@Intel magazine, May 2005

[9] Maxtor Corporation, *Atlas 15K II SAS datasheet*, 2005 Maxtor Corporation.

[10] Maxtor Corporation, *DiamondMax 10 datasheet*, 2005 Maxtor Corporation.

[11] D.A. Patterson, G.A. Gibson, R.H. Katz, *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, Proceedings, ACM SIGMOD Conference, pp. 109116, July 1988.

[12] I. S. Reed and G. Solomon. *Polynomial codes over certain finite fields*. J. SIAM, 8(2):300-304, June 1960

[13] Wicker S., Bhagarv, V., *Reed-Solomon codes and their application*, IEEE press, 1994

Appendix II:

RAID 6 tutorial

May, 2006

Overview: This document provides a tutorial for the coding theory, mathematics and hardware solutions for Reed-Solomon based RAID 6. It aims to provide all the information necessary to enable a hardware or software engineer to develop a RAID 6 solution based upon the use of Reed-Solomon coding theory. This document was written as a tutorial primarily for the benefit of the sponsor and to gather in one place the information and understanding developed by the RE whilst designing hardware RAID 6 solutions based upon Reed-Solomon coding theory.

Michael Gilroy

EngD 4th Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

Reed-Solomon codes are known for their application in compact disks and space communication systems. This same technique lends itself readily to application as a coding scheme for RAID disk arrays. Although Reed-Solomon codes have been used for many years, hardware costs and complexities have precluded their utilisation for RAID based storage solutions.

This tutorial intends to provide an overview of the coding theory, mathematical techniques and the application of these in hardware RAID acceleration. Specifically, details on the design of a RAID 6 based Reed-Solomon encoder and decoder are presented. Whilst the mathematical and coding theory is summarised here, a complete understanding of the theory is not necessary to allow the design and implementation of a RAID 6 hardware controller.

1.1 Requirements

The requirements for a redundant storage system are that:

For *n* storage devices D_1, D_2, .., D_n each of length *k* bytes let there be *m* checksum devices C_1, C_2, .., C_m, each of *k* bytes in length [1], [2].

The required system response is that for all combinations of up to and including *m* storage device failures (erasures), all data may be recovered from the *n* remaining storage devices.

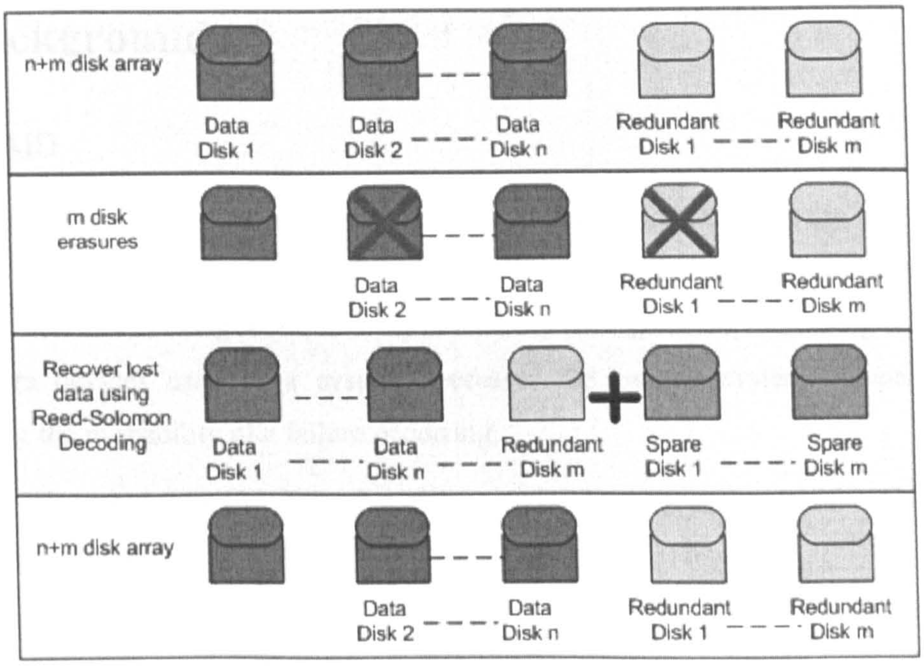


Figure 1. Shows the required system response. Any m devices may fail, but the system can recover the lost data.

2 Background

2.1 RAID

Redundant Arrays of Inexpensive Disks (RAID) allow a number of storage devices to be combined to improve throughput, system reliability and fault tolerance [3]. Whilst there are many benefits to using multiple disks in an array configuration, increasing the number of storage devices used in a system decreases the overall system's dependability, increasing the probability of a failure occurring.

The mean time to failure (MTTF) of a disk array is inversely proportional to the number of devices in an array.

$$MTTF_{array} = \frac{MTTF_{Device}}{Number\ of\ Devices} \tag{1}$$

While a disk array may allow multiple disk drives to be combined to make a larger and faster responding storage device, this inevitably increases the likelihood of a disk failing. To accommodate this, and provide the means to recover from a disk failure, redundant disks may be added to increase an array's tolerance to faults. The most popular RAID levels are 0, 1, and 5 [4]. RAID 1, also known as mirroring, writes all data to two or more storage devices. RAID 0 or striping allows multiple disk drives to be combined together writing data linearly across devices to provide improved data throughput. Striping does not add redundancy to an array and a striped array is not truly a RAID array.

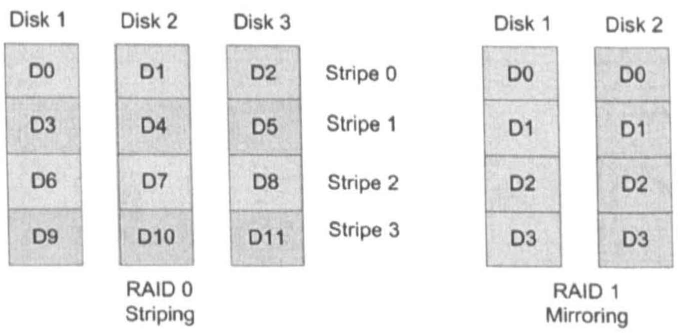


Figure 2. Shows RAID 0 and RAID 1 arrays

RAID 5 provides the optimal solution for single erasure recovery for three or more storage devices, in terms of disk utilisation, requiring only 1 redundant disk drive for *n* data disk drives. Data in a RAID 5 array is striped across the array in equal block lengths, *k* bytes long with each block known as a strip, and the parity of each of these blocks is calculated and written to the spare disk.

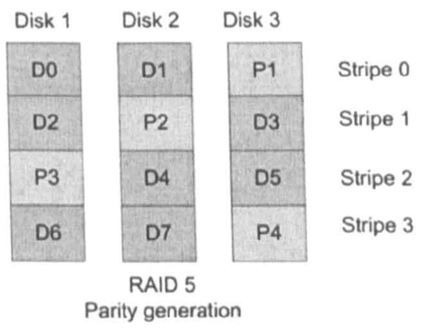


Figure 3. Shows a RAID 5 array

Recovery from a single erasure may be calculated by subtracting the lost data from the parity data (or regenerating the parity). All calculations are performed modulo 2 equating to a series of XOR operations. This architecture may be readily converted to a hardware implementation using a field of XOR gates to perform parity calculations or data recovery.

To provide protection from more than a single erasure more complex solutions than basic parity must be utilised .The second check must allow recovery from any second erasure. A number of algorithms have been proposed for this including, EVENODD [5], and Reed-Solomon coding [6].We discuss only Reed-Solomon coding theory in this tutorial.

3 Coding Theory

3.1 Background

Reed-Solomon codes were first described by Irving Reed and Gustave Solomon in their 1960 paper "Polynomial Codes over Certain Finite Fields" [8]. Reed-Solomon codes are constructed and decoded through the use of finite field arithmetic, specifically Galois Fields, named after the French mathematician Evariste Galois who discovered many of their important features.

3.2 Galois Field Theory

A Galois Field is a field of finite order or cardinality [9]. A Galois Field of q elements is usually denoted as GF(q). The number of elements in a finite field must be of the form p^m, where p is a prime integer and m is a positive integer, and is denoted as GF(p^m) or GF(q). By giving its size, a field is described completely as for any q of the form p^m, the field is unique up to its isomorphism: only fields of the power of the prime exist.

The order of an element α in a GF(q) is the smallest positive integer m such that $\alpha^m = 1$. An element with order (q-1) in GF(q) is known as a primitive element in GF(q). There is always at least one primitive element in a GF(q). As the (q-1) consecutive powers of α must be distinct, they form the (q-1) non-zero elements of GF(q) [9].

Table 1. Shows all non-zero Galois Field elements in GF(q)

Non-zero GF(q) elements	A, $\alpha^2, \ldots, \alpha^{q-2}$
Primitive element GF(q)	$\alpha^{q-1}=1$
Field limit	$\alpha^q = \alpha$

The non-zero group of elements in a Galois Field GF(p^n) is cyclic, as can be seen from the table above once the primitive element has been reached the sequence repeats. Multiplication over a Galois Field is most easily described by the exponential representation:

$$\alpha^x . \alpha^y = \alpha^{(x+y)} \tag{2}$$

To obtain the polynomial representation of the Galois Field the exponential representations of the nonzero elements of GF(q) are reduced modulo the primitive polynomial. The primitive polynomial is an irreducible polynomial in GF(q) if the smallest positive integer n for which p(x) divides (x^n -1) is n=(q^m-1)

Construction of the complete Galois Field may be made by finding one of the irreducible polynomials of the field. For small fields this may be done by a brute force method. This is illustrated below by calculating the primitive elements for GF(4) and GF(8).

3.2.1 Calculating primitive polynomial GF(4)

Our prime field is GF(2) represented by 0 and 1. This is shown as 4 = 2^2. All addition and multiplication is performed modulo 2. To find the irreducible polynomial of degree 2 in GF[x] by a brute force method, list all the quadratic rings of degree 2 with the coefficients of 0 and 1:

- x^2
- $x^2 + 1$

The only irreducible polynomial in this list is x^2 +1 as it does not factor to zero

3.2.2 Calculating primitive polynomial GF GF(8)

Our prime field is again GF(2), since $8 = 2^3$, and may be represented by the values 0 and 1. Again the coefficients can only be 0 and 1, so now all possible cubic polynomials with coefficients of 0 and 1 have to be checked to find the irreducible primitives.

- $x^3 + 1$
- $x^3 + x + 1$
- $x^3 + x^2 + 1$
- $x^3 + x^2 + x + 1$

Here there are two irreducible polynomials. Either may be chosen as the primitive polynomial as both representations are isomorphic.

The polynomial representation of a Galois Field is commonly used for addition operations. Over a Galois Field the associative and commutative laws apply for addition and multiplication operations. This means that both addition and subtraction are the same and may be performed by a bitwise XOR operation.

3.2.3 Galois Field example

The exponential and polynomial representations of a Galois Field of GF(8) may be constructed as follows:

The exponential representation for the field will have (q -1) nonzero elements. For this example q is 8, the seven nonzero elements would be:

$$\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$$

$p(x) = x^3 + x + 1$ is a primitive binary polynomial in GF(2^3). If α is a root of $p(x)$ then:

$$\alpha^3 + \alpha^1 + 1 = 0 \tag{3}$$

which may be re-written to give the value of α^3.

$$\alpha^3 = \alpha^1 + 1 \tag{4}$$

Using equation (2) the polynomial representation for remaining elements may be determined:

i.e. Exponential representation:

$$\alpha^4 = \alpha^3 . \alpha^1 \tag{5}$$

therefore the polynomial representation may be calculated as:

$$(\alpha + 1)(\alpha) = \alpha^2 + \alpha \tag{6}$$

The complete exponential and polynomial representations of the GF(8) are given in the table below:

Exponential Representation	Polynomial Representation
1	1
α^1	α
α^2	α^2
α^3	$\alpha + 1$
α^4	$\alpha^2 + \alpha$
α^5	$\alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1$
α^6	$\alpha^3 + \alpha^2 + \alpha = \alpha^2 + 1$
0	0

3.3 Reed-Solomon Codes

Reed-Solomon codes are examples of linear block codes. The code is constructed by splitting the original data or message into equal length blocks; each block is further divided into k symbols. The code generates an extra $2t$ redundant blocks which may be used to detect or correct errors.

3.3.1 Reed-Solomon Encoding

A number of approaches exist for the generation of Reed-Solomon codes. The original approach is very simple. For a packet of k information symbols, $\{m_0, m_1, ..., m_{k-1}\}$, taken from the finite field GF(q) the symbols may be used to construct the polynomial $p(x)=m_0 + m_1x^{k-1} + ... + m_{k-1}x^{k-1}$. The Reed-Solomon codeword c is formed by evaluating the polynomial $p(x)$ at each of the q elements in a finite field of GF(q).

$$c=(c_0, c_1, ..., c_{q-1})=[p(0), p(\alpha), ..., p(\alpha^{q-1})] \tag{7}$$

By letting the k information symbols take on all possible values, all the possible code words may be found. There will only ever be q different values as they are found over the GF(q). There will be q^k valid code words in any Reed-Solomon code.

Reed-Solomon codes are linear as the sum of any two polynomials of degree (k-1) is another polynomial of degree less than or equal to (k-1). The number of information symbols is also known as the dimension of the code. As each code word has q coordinates, the code length $n = q$. Linear codes are usually denoted by their length and dimension as (n,k) codes.

Each code word is related to a system of linear equations in k variable:

$$\begin{aligned}
P(0) &= m_0 \\
P(\alpha) &= m_0 + m_1 \alpha + m_2\alpha^2 + \ldots + m_{k-1} \alpha^{k-1} \\
P(\alpha^2) &= m_0 + m_1 \alpha^2 + m_2\alpha^4 + \ldots + m_{k-1} \alpha^{2(k-1)} \\
&\vdots \\
P(\alpha^{q-1}) &= m_0 + m_1 \alpha^{(q-1)} + m_2\alpha^{2(q-1)} + \ldots + m_{k-1} \alpha^{(k-1)(q-1)}
\end{aligned} \tag{8}$$

Any k of these expressions can be used to construct a system of k equations in k variables. The first k expressions would form the following system:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & \alpha & \cdots & \alpha^{(k-1)} \\ 1 & \alpha^2 & \cdots & \alpha^{2(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{k-1} & \cdots & \alpha^{(k-1)(k-1)} \end{bmatrix} \cdot \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{k-1} \end{bmatrix} = \begin{bmatrix} P(0) \\ P(\alpha) \\ P(\alpha^2) \\ \vdots \\ P(\alpha^{k-1}) \end{bmatrix} \qquad (9)$$

By computing the determinate of the coefficient matrix, (10), it can be shown that the system has a unique solution for the k information symbols:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & \alpha & \cdots & \alpha^{(k-1)} \\ 1 & \alpha^2 & \cdots & \alpha^{2(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{k-1} & \cdots & \alpha^{(k-1)(k-1)} \end{bmatrix} \qquad (10)$$

The determinate of the matrix reduces to a Vandermonde matrix when co-factor expansion is performed on the top row of the matrix. All Vandermonde matrices are non-singular, therefore any k expressions can be used to determine the values of the information coordinates.

Whilst it is possible to detect as well as correct errors in the received data, this is not necessary in a RAID 6 controller where its errors are directly detected by the hard disk controller. The model being used is that of an erasure. Erasing a Reed-Solomon code word is equivalent to removing the linear equation in (8). As only k correct expressions are required to recover data, up to $q - k$ of the code words may be erased and still allow recovery of the lost data.

This original approach to generate Reed-Solomon codes has been mostly superseded by the generator polynomial approach. This is the same approach used in cyclic codes. A code is said to be cyclic if it meets the requirement that, for any code word c=(c_0, c_1,...., c_{n-1}) the cyclically shifted word c'=(c_1,....., c_{n-1}, c_0) is also a code word [7].

If an (n,k) code is cyclic the code may always be defined using a generator polynomial:

$$g(x)=(g_0, g_1x, \ldots, g_{n-k}x^{n-k}) \tag{11}$$

The code word is calculated by multiplication of the message with the generator polynomial.

$$c(x) = m(x)g(x) \tag{12}$$

The generator polynomial for a t error correcting code within a field of GF(q) must have roots $2t$ consecutive powers of the primitive element, α.

$$g(x) = \prod_{j=1}^{2t}\left(x - \alpha^j\right) \tag{13}$$

Cyclic Reed-Solomon codes GF(q) have a code length of (q-1), one co-ordinate less than achieved by the original method. This is useful for digital design as a field GF(256) may be represented by an 8-bit sequence.

4 Reed-Solomon for hardware RAID 6

The diagram below shows a disk array with 2 data disks and 2 checksum disks. The data disks are split into blocks, these blocks are encoded using a Reed-Solomon code over a Galois Field GF(2^m), therefore encoding and decoding is performed using m-bit long symbols.

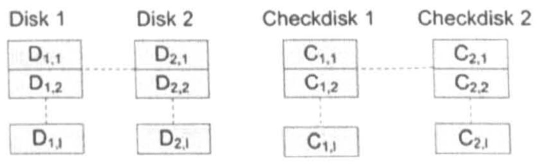


Figure 4. Shows basic RS RAID 6 array

The benefit of using Galois Fields for encoding in a digital system is that all multiplication, division, addition, and subtraction operations result in a fixed length binary word. This is because the field length, unlike the set of integers, is finite. Therefore if encoding is performed on an 8-bit symbol all possible arithmetic operations will result in a 8-bit code word.

To demonstrate the application of Reed-Solomon codes for RAID 6, the process undertaken to generate an encoder and decoder for a field of GF(16) is presented here. This limits the number of storage devices to which data may be written to 15, however, the same techniques may be applied to different Galois Fields of order GF(p^n).

4.1 Galois Field Arithmetic

The following rules hold for arithmetic operations over a Galois Field:

1. The associative law for addition:

$$(a+b)+c=a+(b+c) \tag{14}$$

2. The commutative law for addition

$$(a+b)=(b+a) \tag{15}$$

Addition is equivalent to subtraction over the Galois Field. Therefore both may be performed through the use of bitwise XOR operations.

3. The associative law for multiplication

$$a \cdot b = b \cdot a \tag{16}$$

4. The commutative law for multiplication

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \tag{17}$$

5. The distributive law

$$(a+b) \cdot c = a \cdot c + b \cdot c \tag{18}$$

6. Multiplication is performed by addition of the exponent. This may be performed as a logarithm with base α where the addition is performed by conventional addition.

$$\alpha^i \cdot \alpha^j = \alpha^{(i+j)} \Leftrightarrow \log_\alpha \alpha^i + \log_\alpha \alpha^j = \log_\alpha \alpha^{(i+j)} \tag{19}$$

7. Division is performed by subtraction of the exponent. This may be performed as a logarithm with base α where the subtraction operation is performed by conventional subtraction.

$$\alpha^i \div \alpha^j = \alpha^{(i-j)} \Leftrightarrow \log_\alpha \alpha^i - \log_\alpha \alpha^j = \log_\alpha \alpha^{(i-j)} \tag{20}$$

4.2 Reed-Solomon encoder

A disk array with *n* data storage devices, d_1, d_2, ...d_n, each *k* bytes has *m* associated checksum devices. Each checksum device, C_1,C_2, ...C_m, also hold *k* bytes of data. For any *m* device failures for both data and checksum devices then the failed devices can be reconstructed from the non-failed devices.

Reed-Solomon coding adds check digits (c_i) calculated over a Galois Field from a set of data digits (d_j) by multiplication with a generator function g_{ij} such that:

$$c_i = \sum_{j=1}^{n} d_j \times g_{ij} \qquad (21)$$

For RAID 6 two redundant or checksum drives are required, m=2, giving two syndromes, c_1 and c_2. These two checksums are commonly known as the p and q checksums.

The first checksum equates to a parity calculation across the data storage devices, equivalent to the RAID 5 calculations:

$$c_1 = d_0 \oplus d_1 \oplus \cdots \oplus d_{n-1} \qquad (22)$$

The first checksum may therefore be calculated by performing a bitwise XOR operation on each data storage disk. The second checksum is more complicated, requiring multiplication by the generator polynomial:

$$\begin{aligned} c_2 &= g^0 d_0 \oplus g^1 d_1 \oplus \cdots \oplus g^{n-1} d_{n-1} \\ &= d_0 \oplus 2^1 d_1 \oplus \cdots \oplus 2^{n-1} d_{n-1} \end{aligned} \qquad (23)$$

Multiplication over a Galois Field is performed by addition of the exponents. This multiplication may be performed by using lookup tables of the logarithmic expansion of the multiplicands, and adding the results modulo (2^m-1). These logarithm tables become exponentially larger as the value of m increases, as such this solution is best used when $m \leq 8$.

Table 2. Logarithm Table GF(16)

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
gflog[i]	-	0	1	4	2	8	5	10	3	14	9	7	6	13	11	12

Table 3. Inverse Logarithm table GF(16)

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
gfilog[i]	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9	-

Multiplication therefore may be performed as:

$$g^{(n-1)}d_{(n-1)} = \text{gfilog}[\ \text{gflog}(g^{(n-1)}) + \text{gflog}(d_{(n-1)})] \qquad (24)$$

When updating an entire stripe of data the operations performed are: two multiplication operations, one arithmetic addition and a divide per storage device followed by adding the results using an XOR operation as for the first checksum. This is significantly more computationally intensive than a simple XOR operation.

4.2.1 Updating a strip

Whilst equations 22 and 23 are appropriate when calculating the checksum of an entire stripe, they are inefficient for calculating the checksum when only one of the data disks, a strip, has to be updated.

If the data word on a single storage device has to be updated to d'_j from d_j then the checksum is recomputed by applying the function $g_{i,j}$:

$$c'_i = c_i - (d_j \times g_{ij}) + (d'_j \times g_{ij}) \qquad (25)$$

This means that both the checksum disks and the data disk which has to be updated have to be read to calculate the new checksums. So a write operation to a single disk drive requires three read operations and three write operations.

4.3 Reed-Solomon Decoder

When up to m devices fail the system is reconstructed as follows:

- For each data device d_j which fails, construct a function to restore the words in d_j from the words in the non-failed devices
- Recompute any failed checksum devices c_i with g_i

From these operations the entire system is reconstructed.

4.3.1 Re-Calculating Single erasures

For single disk erasures one of two methods to recalculate the lost data is used.

If the lost data is a checksum, the checksum can be recalculated from the existing data using equations 22 and 23. This is the same method as for encoding the data originally.

If the lost data is not a checksum, set the missing data to 0 and calculate the parity. Missing data is calculated by subtracting the new parity from the original by performing a bitwise XOR operation on with the original data and the p checksum.

4.3.2 Re-Calculating double erasures

If both failed storage devices contain checksum data then recalculate the checksum using equations 22 and 23.

If there are two missing items calculate s_1 and s_2, the modified checksums from setting the missing data to 0 then:

$$c_1 \oplus s_1 = d_i \oplus d_j \tag{26}$$

$$c_2 \oplus s_2 = (d_i \otimes i) \oplus (d_j \otimes j) \tag{27}$$

The missing data is calculated from:

$$\begin{aligned} &(d_i \otimes i) \oplus (d_j \otimes j) = c_1 \oplus s_2 \otimes i \\ &\Rightarrow (d_i \otimes i) \oplus (d_j \otimes j) = (c_2 \oplus s_2) \oplus [(c_1 \oplus s_1) \otimes i] \end{aligned} \tag{28}$$

and

$$\begin{aligned} &(di \otimes i) \oplus (d_j \otimes j) = c_2 \oplus s_2 \\ &\Rightarrow (c_2 \oplus s_2) \oplus \frac{(c_1 \oplus s_1) \otimes i}{i \oplus j} \end{aligned} \tag{29}$$

5 References

[1] Plank J. S., A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like systems, Software -- Practice & Experience, 27(9), September, 1997, pp. 995-101

[2] James S. Plank and Ying Ding, Note: Correction to the 1997 Tutorial on Reed-Solomon Coding, *Software, Practice & Experience*, Volume 35, Issue 2, February, 2005, pp. 189-194.

[3] D.A. Patterson, G.A. Gibson, R.H. Katz, *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, Proceedings, ACM SIGMOD Conference, pp. 109-116, July 1988.

[4] Chen P., Lee E. K., et al., RAID: High-Performance, Reliable Secondary Storage, ACM Computing Surveys, 26(2):145-185, 1994

[5] M Blaum et al. *EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures*, IEEE Transactions on Computers, Vol 44, No 2, PP 192 -202, February 1995.

[6] Jing M. H., Chen Y. H., Liao J. E., A fast error and erasure correction algorithm for a simple RS RAID, Beijing. 2001 International Conferences on Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Volume 3, 29 Oct.-1 Nov. 2001 Page(s):333 - 338 vol.3

[7] S Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*, IEEE Press, New York, 1994.

[8] S. Reed and G. Solomon, *Polynomial Codes over Certain Finite Fields*, SIAM Journal on Applied Mathematics, vol. 8, pp. 300-304, 1960.

[9] J. M. Howie, *Fields and Galois Theory*, Springer, 2006

Appendix III:

RAID 6 IP codec design and implementation

December, 2003

Overview: This document discusses the design and implementation of the most basic Reed-Solomon (RS) encoder and decoder for use in a RAID 6 hardware accelerator. This was the first task of the engineering doctorate and implemented the smallest possible implementation of a RAID 6 hardware controller. The aim of this design was to produce an optimised design for supporting a RAID array with two data disks and two checksum disks and to introduce the research engineer to the theory behind RAID 6.

Michael Gilroy

EngD 2nd Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

This document describes the algorithm and hardware implementation of a simple Reed-Solomon encoder and decoder to be used in a RAID 6 hardware accelerator. The implementation selected provides support for two data disks and two checksum disks in a RAID 6 disk array.

2 RAID 6 Algorithm

A number of algorithms may be considered for the implementation of double disk redundancy (RAID 6) systems [1], [2]. However, for this implementation we consider only Reed-Solomon codes [3]. This method has been selected as it is readily implemented in hardware requiring only XOR operations, look up tables, adders and subtractors to implement a Reed-Solomon encoder and decoder.

The algorithm proposed for initial implementation of a RAID 6 device will be optimised for operation with four hard disk drives, two data and two checksum disks. However, the principle of the design lends itself to expansion to larger disk arrays, which will be implemented in future designs.

Although the theory of Reed-Solomon codes and Galois field arithmetic have been established for some years, the idea of utilising this theory for storage systems has been largely discarded due to the cost of implementation in hardware. Furthermore, the idea of using a Galois Field GF (2^2) is commonly discarded or used only to demonstrate the theory in its simplest form.

This implementation is used to provide an optimised hardware implementation of a Reed-Solomon codec for a four disk RAID 6 array.

2.1 Reed-Solomon encoder

The disk array has two data disks and two checksum disks. The encoder will operate on a word length of 2 bits. The first checksum, C_1 is simply the parity of the two data bits.

$$c_1 = d_0 \oplus d_1 \qquad (1)$$

The second checksum is calculated using a generator polynomial as follows:

$$c_2 = g^0 d_0 \oplus g^1 d_1 = d_0 \oplus 2^1 d_1 \qquad (2)$$

Multiplication over a Galois field is performed by addition of the exponents. This multiplication may be performed by using lookup tables of the logarithmic expansion of the multiplicands, and adding the results modulo (2^m-1).

Table 1. Logarithm Table GF(4)

I	0	1	2	3
Gflog[i]	-	1	2	3
Gfilog[i]	0	1	3	-

Multiplication therefore may be performed as:

$$g^{(n-1)}d_{(n-1)} = \text{gfilog}[\ \text{gflog}(g^{(n-1)}) + \text{gflog}(d_{(n-1)})] \qquad (3)$$

Division is performed by subtracting the logarithmic values prior to taking the inverse logarithm:

$$g^{(n-1)}d_{(n-1)} = \text{gfilog}[\ \text{gflog}(g^{(n-1)}) - \text{gflog}(d_{(n-1)})] \qquad (4)$$

2.2 Reed-Solomon Decoder

2.2.1 Re-Calculating Single erasures

If the lost data is a checksum, the checksum can be recalculated from the existing data using equations (1) and (2). This is the same method as for encoding the data originally.

If a single data disk or the first checksum fails we re-calculate the lost data by taking the parity of the remaining data.

2.2.2 Re-Calculating double erasures

If both failed storage devices contain checksum data then we recalculate the checksum using equations (1) and (2).

If there are two missing items calculate the modified checksums, s_1 and s_2, from setting the missing data to 0 then:

$$s_1 = d_0 \oplus d_1 \tag{5}$$

$$s_2 = (d_{i0} \otimes 1) \oplus (d_1 \otimes 2) \tag{6}$$

The missing data is calculated from:

$$\begin{aligned} &(d_0 \otimes 1) \oplus (d_1 \otimes 2) = c_1 \oplus s_2 \otimes 1 \\ &\Rightarrow (d_0 \otimes 1) \oplus (d_1 \otimes 2) = (c_2 \oplus s_2) \oplus [(c_1 \oplus s_1) \otimes 1] \end{aligned} \tag{7}$$

and

$$\begin{aligned} &(d_0 \otimes 1) \oplus (d_1 \otimes 2) = c_2 \oplus s_2 \\ &\Rightarrow (c_2 \oplus s_2) \oplus \frac{(c_1 \oplus s_1) \otimes 1}{1 \oplus 2} \end{aligned} \tag{8}$$

3 System Design & Test

3.1.1 Software Modelling

A software model of the algorithm was generated to verify the implementation as valid, as well as to generate test data for hardware simulation. This test programme showed that the algorithm was implemented correctly and formed the basis of the hardware implementation. It was noted at this point that by modifying the lookup tables it would be possible to give the result of equations 3 and 4 for multiplication and division of two numbers in one table. This would remove the need for arithmetic addition and subtraction to be performed in the hardware implementation and reduce the system latency. This results in a larger table look up as shown in Table 2 and Table 3. However, this larger look up table removes the need for the addition and subtraction operations which would otherwise be required.

Table 2. Expanded GF multiplication table

Multiplicand	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
Multiplier	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
Product	0	0	0	0	0	1	2	3	0	2	3	1	0	3	1	2

Table 3. Expanded GF division table

Dividend	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
Divisor	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
Quotient	0	0	0	0	0	1	3	2	0	2	1	2	0	3	3	1

3.1.2 Hardware Desgn

The hardware was implemented as separate encoder and decoder modules with only one producing valid data outputs at any given time. The encoder implementation outputs encoded data after a one clock delay. The encoding function was reduced to two XOR operations and two multiplicative table look ups.

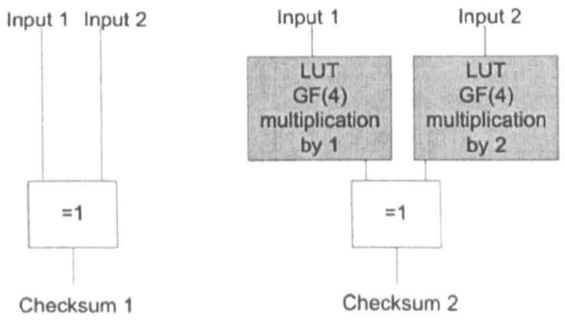


Figure 1. Checksum generation logic

The data inputs to the encoder, d_1 and d_2, are both two bits wide. All possible combinations of 2 bit data input were input in simulation to the encoder function. The results of this encoding were compared with the results generated by the software model to verify the encoder operated correctly.

The decoder included an encoder for regenerating lost checksum data and additional logic to perform the operations required to solve for all combinations of single and double disk failure. The output decoded data was selected based upon the disk or disks which were indicated as having failed. There was a latency of 3 clock cycles in the decoder logic due to the increased operations required to decode the output. Simulation of the decoder using the previously encoded logic showed that the decoder was able to recover all valid combinations of single and double disk failures.

3.2 RAID 6 codec performance

By instantiating 16 encoder and decoder modules in parallel it was possible to encode and decode 32-bit wide data inputs. The data width has been parameterised to adjust to the desired data width to suit the application and may be controlled at the top level of the design by setting the data bus width parameter.

To verify that the hardware design was suitable for implementation on an FPGA target the design was synthesised using Altera's Quartus II version 4.0 targeting the Cyclone device family. The encoder and decoder functions were found to synthesise and run at clock rates in excess of 250MHz, exceeding the 150MHz clock rate of the current first generation Serial ATA devices which we hope to target with this design.

4 Conclusions

This error correction algorithm has been simulated in both hardware and software to verify the operation of the proposed design. The error correction algorithm has been coded in software and will operate at 800 Mbps on a 2GHz Pentium 4 processor under Windows. The Reed-Solomon codec has been coded in Verilog and targeted at the Altera Cyclone family. The codec operates at or above 250 MHz in hardware. An optimal RAID 6 encoder and decoder for four disk arrays has been demonstrated. However, this design will need to be modified to support larger disk arrays in the future.

5 References

[1] M Blaum et al. *EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures*, IEEE Transactions on Computers, Vol 44, No 2, PP 192 -202, February 1995

[2] Chan-Ik Park, *Efficient Placement of Parity and Data to Tolerate Two Disk Failures in Disk Array Systems* IEEE transactions on parallel and distributed systems, Vol. 6, No. 11, November 1995 pp1177-1184

[3] I. S. Reed and G. Solomon. *Polynomial codes over certain finite fields.* J. SIAM, 8(2):300-304, June 1960

Appendix IV:

RAID 6 Accelerator Hardware Development

July, 2004

Overview: This document relates to work performed from the commencement of the EngD project work in October 2003 until May 2004 under the project title of "Techniques for ubiquitous reliable data storage". This document's purpose is to bring together both the background research, algorithm development and implementation and initial results and analysis of the work to this stage of the project.

This document discusses the implementation of a PCI bus based RAID 6 hardware accelerator proof of concept design. The design and test of individual design blocks is discussed, the overall system architecture is presented and the results of testing and verification discussed.

Michael Gilroy

EngD 2nd Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

The aim of this development phase was to produce a proof of concept design that could be used as a technology demonstrator of a functioning hardware RAID 6 accelerator. The only restrictions placed on the design of the hardware was that no third party IP blocks could be used. This was due to a desire to reduce the cost of the final product and to encourage development of new IP blocks for the sponsoring company.

The proof of concept design was to be developed on a PCI development board with onboard FPGA [1]. An Altera Flex PCI development card was available from the project outset: this device is capable of supporting PCI data transfers of up to 64 bits at 66 MHz. Using this device, it was planned that a hardware accelerator operating on PCI 32 bit wide data bus at 33 MHz would be developed. From this design, it was hoped that the design could be easily modified to accommodate 64 bit wide data transfers and then increase the clock rate to 66 MHz.

Upon completion of this phase in development, it was envisaged that a PCI-X based accelerator card with on board cache and SATA controllers would be designed and built. This would provide the means to develop and test the first production model. The original roadmap for the hardware development is shown below in Table 1.

Table 1. Hardware roadmap for RAID6 controller

Q1 2004	Q3 2004	Q4 2004	Q1 2005	Q3 2005
Proof of Concept Demonstration	Receive Prototype board for testing 4 disk RAID6 accelerator card	Complete testing of prototype, begin manufacture	First devices of production model ready for sale	Development of 15 disk RAID 6 accelerator card complete

Production models would be purchased using Altera's hardcopy programme to convert the FPGA design into an ASIC offering a lower unit cost than an individual FPGA and would reduce the expense and area of the accelerator components. Having begun production, larger more complicated arrays were to be supported using an updated hardware accelerator which would support up to 16 disks. Beyond this stage it was hoped the product would prove successful, and, that an SoC would be developed which would allow the unit cost to drop and allow the accelerator card to be added directly into PC motherboards.

2 System design

Only the smallest RAID 6 arrays had to be tested and demonstrated for the proof of concept design. To test the RAID 6 codec in hardware the smallest RAID array containing 4 hard disk drives was selected to form the basis for all testing in the prototype stage. The hardware accelerator was to be implemented on the PCI based FPGA accelerator. This would provide a PCI bus interface, basic DMA controller and RAID 6 encoder and decoder.

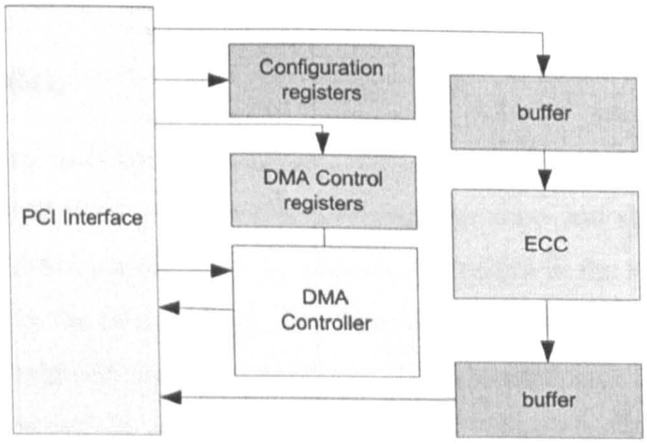


Figure 1. RAID 6 accelerator FPGA design blocks

The basic test platform used for both hardware and software development is shown below. The test platform was a standard PC system with a Pentium II processor running Linux kernel 2.6 [2]. The hardware accelerator was to be configured and controlled by the host CPU using a modified software RAID 6 driver [4]. All data was to be read into local memory by the CPU prior to being accessed by the hardware accelerator. Access to the 80 GB hard disk drives was provided by two Silicon Image Si3112 PCI SATA controllers [3]. Lack of direct access to the SATA controllers by the RAID 6 controller and the need for the CPU to configure and control the accelerator was recognised as being an inefficient design, however, it reduced the amount of custom hardware that had to be developed for the initial prototyping stage.

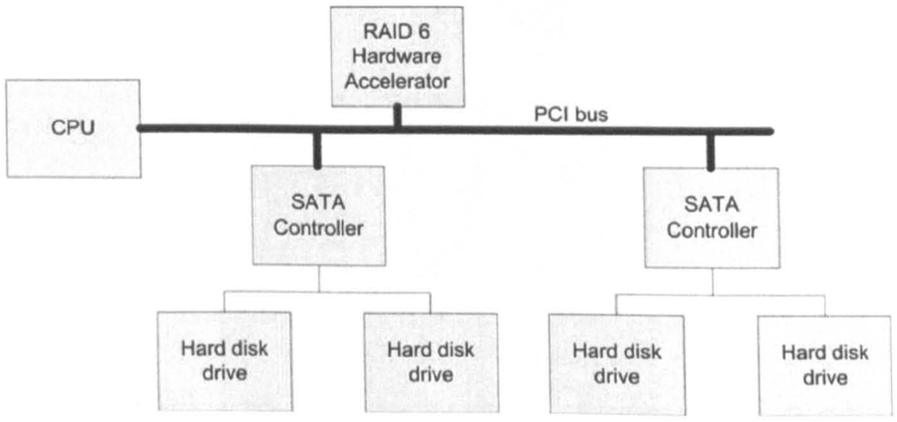


Figure 2. Shows the test system architecture used in proof of concept testing

3 FPGA system design and test

3.1 RAID 6 codec

The RAID 6 encoding and decoding was performed by a Reed-Solomon codec developed for this project. The IP block was verified through simulation and shown to encode and decode all possible data combinations. A wrapper was added to the basic Reed-Solomon codec for this phase in the development. This wrapper added memory and control logic to the Reed-Solomon codec to facilitate its use as a hardware RAID 6 codec. The Reed-Solomon codec operated on 2-bit wide data inputs. Sixteen of these modules were instantiated in parallel to provide support for 32-bit wide data inputs, matching the data bus width of the PCI bus.

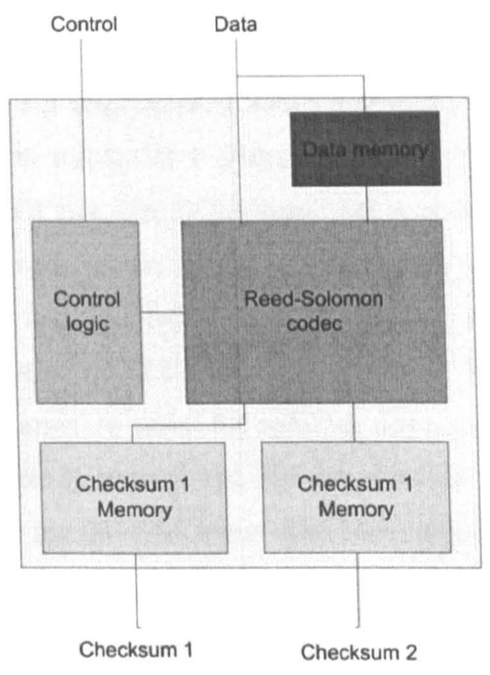


Figure 3. RAID 6 hardware codec. The additional control and memory RAM blocks added to the basic Reed-Solomon codec are shown here.

As the Reed-Solomon codec expected to receive two disks worth of data simultaneously, the data memory was added to buffer the first disks worth of data prior to the reception of the second disks worth of data. The need to buffer the data limits the performance of the RAID 6 codec to operating of blocks of data up to the buffer size. The buffer limits were set to 4096 bytes of memory per memory block. This was the largest memory blocks that could be fitted on the FPGA with the rest of the system logic. Larger memory modules would have been desirable, as the software provided data in blocks of quadruple this size, however it was not possible to achieve this on the Flex FPGA. The memory blocks were implemented as RAM modules. RAM blocks were chosen over FIFO buffers to assist with the PCI bus transfers as there was no guarantee that a burst transfer would be successful and it may be necessary to resend data from the last successful block of data transferred.

The RAID 6 codec was shown to operate in simulation for all possible data inputs. This was a repeat of the testing performed for the Reed-Solomon codec prior to the extra logic being added.

3.2 DMA controller

A simple DMA controller was implemented which performed memory accesses from the PCI bus, passed the data to the RAID 6 codec, and wrote back the encoded data to a memory location on the PCI bus. The DMA controller was designed to connect directly to the PCI IP core and provide the necessary control signals to access data over the PCI bus. The DMA controller is controlled by a set of registers accessible through the PCI configuration register space or via PCI memory transactions. This double mapping of the control registers was performed to assist the software development as configuration read and write accesses had already been tested and were known to work. Memory transfer support was to be added to the driver at a later date once initial hardware testing had been performed.

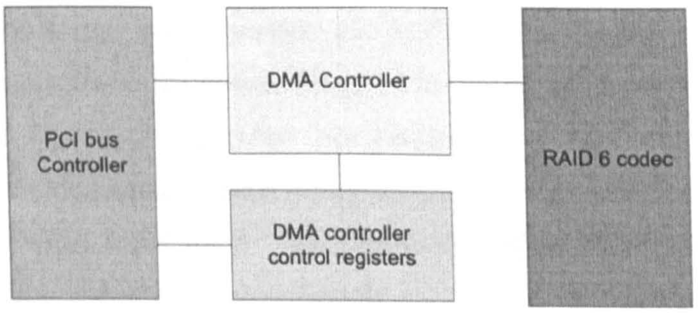


Figure 4. DMA controller is configured via the PCI bus by the host computer. It then reads blocks of memory over the PCI bus passes the data to the RAID 6 codec, and writes back the encoded/decoded data over the PCI bus.

3.3 PCI IP block

The development of a custom PCI controller to provide PCI bus master and target functionality was made necessary due to the constraint of not utilising third party IP blocks. The PCI interface was based upon revision 2.3 of the PCI local bus specification, for full details of the PCI bus and its operation please refer to this specification [5].

The implemented PCI interface supports

- 32-bit wide data transfers at up to 33MHz.
- 32-bit addressing mode
- 32-bit wide data transfers.
- Bus master/target support

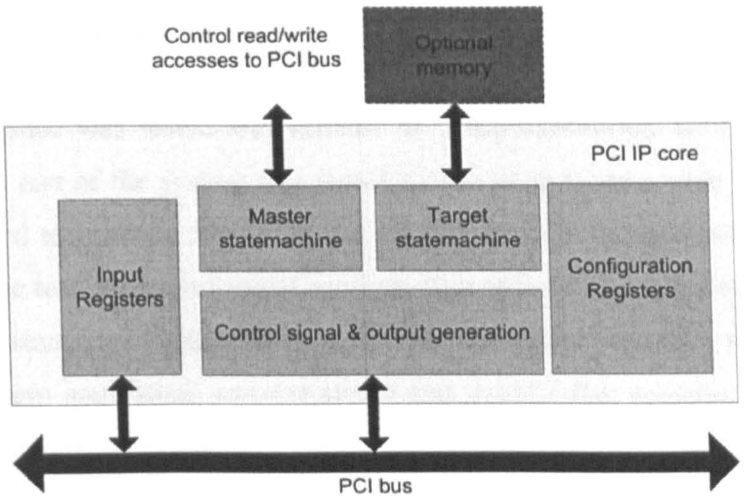


Figure 5. PCI IP block implements both PCI master and target functionality. The core is configured via the configuration registers at synthesis. The target may be connected to a memory block accessed by PCI memory read/write transactions.

The PCI controller design was separated into a PCI target controller and PCI master controller. The target functionality was designed to operate independently of the master functionality. The PCI target controller was designed such that is may be configured through the PCI configuration register block to appear as any desired target device or offer any required target functionality. The target was configured, for the purposes of this project to appear as a custom encryption device.

The PCI master controller was designed to provide support for both single data transfers and burst data transfers. The PCI master controller ensures that all data transfers adhere to the PCI bus standard and controls the flow of data and addressing.

For our system the target memory mappings were connected to the DMA controller control and status registers. The configuration registers were also extended to map to these same registers.

4 Verification & testing

The RAID 6 codec was tested and verified as being operational using a stand alone simulation. The rest of the system was tested in one large system wide test bench. This reduced the need to produce models of the separate system components to model each individually. The test bench performed configuration of the PCI controller, mimicking the operations at system boot performed on a PC, then the DMA controller was initialised to perform checksum generation, recover single and double disk erasures and test all the functions performed by the host PC.

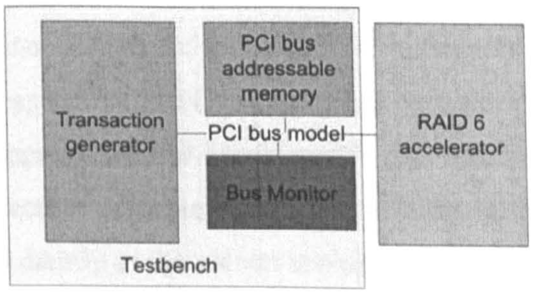


Figure 6. System test bench. Incorporated a bus monitor to verify PCI transactions and a region of memory accessible over the model of the PCI bus.

Monitoring of the PCI bus to verify that transactions conformed to the PCI bus requirements was incorporated into the test bench. This allowed a large number of PCI transactions to be carried out with little in the way of manual checking being required. A model of a region of memory accessible via the PCI bus was added to enable the data and checksum data to be stored and accessed by the RAID 6 accelerator.

To verify that the PCI core operated correctly proved to be both time consuming and awkward. A Verilog test bench was generated and initial simulations of the target and master device showed that the PCI core operated as expected and handled all supported transaction correctly. However, when the target controller was implemented on the PCI development board the test PC failed to boot into Linux. The development card offered no means by which to examine directly the PCI bus transactions which were taking place and the Flex FPGA did not offer support for Altera's SignalTap II on chip logic analyser.

To assist in the debug of the hardware at this point it was requested that a PCI development board based upon either the Cyclone or Stratix FPGA device families be purchased as these offered support for SignalTap II. This request was not met by the sponsor at this time.

To allow development to continue it was necessary to be able to observe the transactions occurring on the PCI bus. To this end a PCI riser was purchased which allowed a logic analyser to be connected between the PCI bus and the FPGA development board. Unfortunately this proved to be an unsuccessful solution. When the riser was connected the FPGA failed to respond to PCI transactions and the computer would fail to boot. This was later determined to be a result of the PCI riser increasing the routing length between the PCI bridge and device beyond the maximum valid routing length.

By connecting the PCI bus riser to the logic analyser and inserting into a spare PCI bus slot it was possible to capture valid PCI transactions between the various PCI devices. This data was used to improve the test bench models and bus monitoring and assisted in the location of and correction of a number of design faults in the target PCI controller. The faults found related mainly to the correct timing for setting and de-asserting the PCI control signals.

Re-testing the target PCI controller on the test PC still resulted in the failure for the system to complete the initial boot sequence. It was found that the PCI bus control signals, trdy and devsel, indicating the readiness of the target to respond to initial configuration read and write accesses were being held after the PCI bus transactions should have completed. This error was not observed during simulation and appeared to be the result of a timing failure in the FPGA.

A considerable amount of time was spent trying to resolve this timing issue. Through close reference to the timing constraints for Altera's own PCI core for the Flex FPGA it was determined that the routing delay for certain PCI bus control signals failed to meet the minimum allowable setup and hold timing requirements. Reliance on the automatic place and route tools failed to produce a successful placement for the PCI IP block. By adding more stringent timing requirements to the internal signals controlling and connecting to those failing to meet the timing constraints it was possible to force the design to meet the timing requirements.

The target PCI core was then shown to operate correctly, responding to PCI configuration accesses and allowed access to a small memory space that was added to the design.

Verification of the PCI master controller proved to be equally complicated as for the target controller once more due to lack of visibility of the bus and FPGA transactions. A simple programme was written in C to assist in the testing of the PCI master. The PCI based RAID 6 accelerator was detected by the programme, a region of memory for read and write transactions was setup and the DMA controller was configured to encode two blocks of 4096 bytes of data and write back the results to the reserved memory locations.

The PCI master controller was shown to work successfully during these tests. Furthermore the RAID 6 codec and DMA controller were shown to operate correctly. The test programme was extended to verify the operation of the RAID 6 codec for all modes of operation. These tests were all passed successfully.

5 RAID 6 hardware software integration

A device driver was developed to allow the RAID 6 hardware accelerator to be utilised on a PC running the Linux 2.6 kernel. When attempting to build RAID 6 arrays using this device driver the test PC repeatedly crashed or locked in an unusable state. It was not possible to capture the PCI bus activity at the time of the failure, nor were the software logs of any use in debugging the failed system response.

After further debugging of the software driver it was determined that the failure was in part due to incorrect implementation of the device driver. Ultimately though, the device driver proved to be unstable and unreliable. It was not possible to conclusively state that the cause of this was exclusively a result of the software or hardware. Without the means to reliably capture the PCI bus transactions when failures occurred, verify the register transactions on the FPGA and determine the state of both the hardware and software at the time at which a failure occurred it was not possible to determine the cause of the observed system failures.

6 Hardware RAID 6 performance

Minimal results were gathered for the RAID 6 hardware accelerator when implemented on the FPGA. Using the software RAID tools available in Linux we were able to determine that during the build speed during the generation of a RAID 6 array was between on a 5 and 31 MB/s. We were not able to successfully complete testing with the BONNIE benchmarking tool which we had used to quantify the performance of the software RAID 6 solution [6].

7 Conclusions

The PCI bus IP core which was generated has been shown to operate in basic bus transactions, however, for more complex burst transactions it is not possible to say for certain that the IP block functions correctly. The RAID 6 codec has been tested and verified as operating correctly by simulation. It has also been observed as operating correctly during testing on the FPGA development card.

It was not possible to completely verify the hardware RAID 6 accelerator on the available FPGA development board with the facilities available for testing purposes. Further testing and development work will be required to verify the hardware operation and this would be best aided through the use of a more modern FPGA development platform which would allow on-chip logic analysis to be performed.

8 References

[1] Altera, Flex PCI development board,

[2] L. Torvalds et al., Linux 2.6 Kernel, www.kernel.org

[3] Silicon Image, Sil3112 PCI to Serial ATA Controller datasheet, Silicon Image, Inc. August 2003

[4] M Gilroy, RAID 6 software development, July 2004

[5] PCI Special Interest Group, PCI Local Bus Specification, Revision 2.3, March 29, 2002

[6] Bray, T. BONNIE user manual,
URL http://www.textuality.com/bonnie/intro.html, Last accessed 12th September 2005

Appendix V:

RAID 6 Software Development

July, 2004

Overview: This document introduces the software RAID 6 device driver and the changes made to it to enable it to act as a hardware device driver for a custom RAID 6 hardware accelerator. The hardware software integration is also presented along with recommendations for future work based on this initial driver development phase.

Michael Gilroy

EngD 2nd Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

To complement the design of the hardware RAID 6 accelerator a software based RAID 6 device driver was developed to provide a benchmark for determining the benefits derived from the use of the RAID 6 hardware accelerator. It was also hoped that a software only solution could be used in an embedded system to perform the RAID 6 algorithm and so produce a low cost equivalent of the hardware system. It was also beneficial as it allowed the development of a software driver to proceed without the need of the completed hardware accelerator.

Software development was initially carried out by Mr William Berrie targeting the Apple Power Mac G5. This decision was made as the sponsoring company was in discussions at the time with a potential customer for a software driver for the SATA controllers for Apple Power PCs. Development of software drivers for the Apple stopped after 6 weeks due to lack of support available for software driver development and the intended customer having abandoned the project. At this point it was decided to continue software development for Linux.

Linux was selected due to ease of access to the source code and, in particular, the recent release of a preliminary RAID 6 algorithm for the 2.6 kernel [1]. It had been hoped that by modifying this RAID 6 software driver source code that support for the RAID 6 hardware accelerator could be added and additional RAID 6 algorithms could be tested in a PC running Linux. It was noted from the outset that the software RAID 6 drivers did not work reliably and significant reworking of the code was required to ensure incorporate the RAID 6 algorithm and get the software drivers to work.

2 Software RAID 6

The software RAID 6 driver provided the following functionality:

- Register the driver with the Linux kernel. Allowing the use of standard Linux RAID array tools to generate and maintain RAID arrays.
- Determine the best algorithm for the current hardware. The algorithm utilised to encode or decode data on the system was selected by measuring the throughput achieved on a number of different implementations, optimised for certain processor architectures, and selecting the algorithm which offered the highest data throughput rates.
- Access to disk drives.
- RAID 6 array generation functions
- RAID 6 data recovery functions

The first task undertaken was to introduce a software implementation of the algorithm used by the hardware RAID 6 accelerator. This allowed a familiarity to be developed with the operation of the device driver as well as assisting in determining how and where to add support for the hardware RAID 6 accelerator.

During the testing of this algorithm a number of problems with the software driver were identified. The major issue noted was that the original driver did not work over a prolonged time period. The exact cause of this problem was never resolved. A number of updates to the source code were released during the testing phase and many of these were incorporated into our test software driver and managed to improve data throughput and system performance. Other issues included missing functionality for certain types of disk updates, although these were not needed for testing purposes, and the use of both old and new styles of kernel commands. As the 2.6 kernel had only recently been released, there was little literature providing support for development of source code for the kernel as yet.

The CPU utilisation for the software RAID 6 device driver was between 10-20% during array generation. The software performed far better than had originally been expected. Indeed when operating on small file sizes the software performs exceptionally quickly thanks to the large amount of cache memory available. As the array size increases the software is limited by the inefficiency in the early release of the driver. The software driver produced data throughput rates of between 25-50 MB/s during testing using the BONNIE benchmarking tool [2]. The generation of the RAID 6 array operated at data throughput rates of 10-20 MB/s. This figure was determined by performing queries with the software RAID mdadm tool [3].

3 Device driver for RAID 6 accelerator

Three main changes had to be made to the the existing software RAID 6 driver to enable it to be used as a hardware device for the RAID 6 accelerator. First the driver had to check for the presence of the hardware accelerator. This was performed by scanning the attached PCI bus devices to locate the hardware accelerator. When detected instead of loading the software RAID 6 functions, replacement functions which utilised the hardware accelerator had to be added. Finally an interrupt handler had to be added to handle PCI interrupts generated by the hardware.

The software driver provided three separate functions for RAID 6 calculations

1. A checksum generation function for calculating the two checksums
2. Single disk recovery, used to recover missing data during a single disk erasure
3. Double disk recovery, used to recover missing data during a double disk erasure

Three equivalent functions were generated for use with the hardware accelerator. Each of these functions converted the memory location for data into a PCI bus accessible address, configured the hardware accelerator to perform the correct encode or decode operations, initiate DMA transactions and sleep until the PCI interrupt was set.

Initial testing of the device driver with the hardware accelerator caused the test PC to crash and no debug information could be retrieved from the test system.

The cause of the system failures was determined to be a result of the incorrect use of various work queues and spin locks in the device driver. While waiting for the hardware interrupt from the hardware accelerator, to indicate completion of the RAID 6 calculations, the device driver was being woken by a software interrupt and entering an invalid state. The cause of the locking was determined by moving the software driver from using the interrupt based control to continually polling the DMA status register to determine when the hardware had completed all operations. Using this method it was possible to build a RAID 6 array. Unfortunately, the performance achieved using this method was very poor, resulting in array build speeds of around 100KB/s.

By placing semaphores before the other locking mechanisms used within the device driver, it was possible to utilise the interrupt based solution. The other locking mechanisms were kept to keep to a minimum the differences between our modified device driver and the standard device driver. This improved the data throughput achieved through utilisation of the RAID 6 hardware accelerator and the array was generated at between 10-25MB/s.

Although we were able to build and rebuild RAID 6 disk arrays using the hardware accelerator, as the disk storage capacity of the array was increase and data was written to the array, the test system would crash and the test system would require to be rebooted. The cause of this fault could not be determined as it could be the result of a fault in the implementation of the PCI IP core, a fault in the modifications added to the device driver, or a fault inherent in the original device driver being exasperated through the use of the hardware.

4 Conclusions

Simple software programmes to drive the hardware appear to show the system operates correctly. However, these tests may be producing false positive results and hiding underlying problem with the hardware. Alternatively, the modified device driver could have introduced or be highlighting existing software errors. Once a more complete and stable software driver is available it may be possible to re-investigate the exact cause and nature of these errors.

The software RAID 6 solution works reliably although inefficiently. The implementation of a model of the hardware RAID 6 accelerator implementation of the encoding and decoding algorithms was successful. The use of the software RAID 6 driver as a hardware device driver for the hardware accelerator has been investigated and found wanting. The main issues noted was the use of inappropriate locking mechanisms used in the software driver which appear to be unusable with the hardware. Due to the lack of readily available documentation on the new Linux kernel, the complexity of the software driver and continued uncertainty over whether the hardware or software is responsible for the system failures continued development can not continue until either more appropriate testing facilities become available and more complete documentation for coding device drivers for Linux kernel 2.6 are available.

5 References

[1] L. Torvalds et al., Linux 2.6 Kernel, www.kernel.org

[2] Bray, T. *BONNIE user manual*,
URL http://www.textuality.com/bonnie/intro.html, Last accessed 12th September 2005

[3] N. Brown, http://neilb.web.cse.unsw.edu.au/source/mdadm/

Appendix VI:

Review of Arran stage II

April, 2006

Overview: This document presents the results of the successful testing and benchmarking of the hardware RAID 6 accelerator, under the A2E project codename of Arran stage II. Having presented the results, a number of potential products based upon the technology are proposed and the benefits and risks associated with each discussed.

Michael Gilroy

EngD 4th Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

A hardware RAID 6 accelerator has been developed and shown to operate for small disk arrays at speeds which out perform software based solutions. This document discusses current performance benchmarks observed for the RAID 6 hardware accelerator and from this, predicts performance figures for a number of derivative products based on the technology. These potential technologies are compared and contrasted with current and forthcoming products available from various competitors in these markets.

This document aims to detail the available markets, predict times to market and recommends a course of action to bring the RAID 6 technology to market.

2 RAID 6

2.1 Technology Overview

Storage requirements for home and business use continue to increase year on year. The protection and reliability of this data is becoming more critical. Currently Redundant Arrays of Independent Disks (RAID) storage technologies allow the combination of multiple disk drives to improve read/write performance or through the addition of redundant drives provide the ability to recover from disk failures.

RAID 6 provides the ability to recover from up to two simultaneous disk failures. This necessitates two redundant drives in the array to hold the checksums generated by the RAID 6 encoding algorithm. A Reed-Solomon coding scheme is used to encode data using a hardware accelerator to increase performance.

Whilst the likelihood of a double disk failure occurring is relatively low, the problem is that during recovery of a single disk error the probability of an unrecoverable read error occurring during the rebuild process is very high. For larger arrays, and particularly for lower cost disk drives, the probability is that this will happen every time the array is rebuilt.

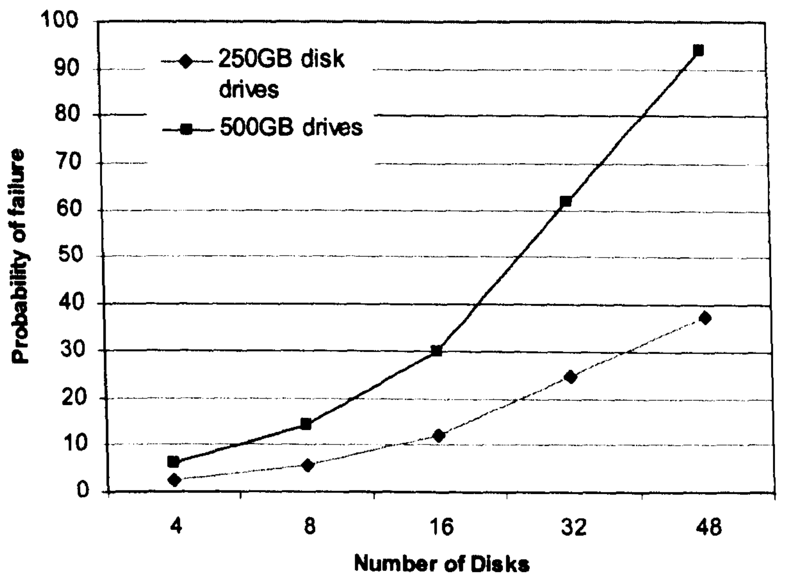


Figure 1. Graph shows the probability of disk failures occurring on various sized arrays and drives

2.2 Proof of Concept Architecture Overview

The proof of concept hardware has demonstrated the performance benefits of using hardware acceleration over a software only solution. The hardware accelerator has consistently outperformed the software only solution for rebuilding arrays and during double disk failures.

The proof of concept hardware accelerator provided only RAID 6 acceleration, disk access, and setup and control of the array were handled by the host PC. The hardware accelerator operated over a PCI bus operating at 66 MHz with a bus width of 64 bits. Control was handled by a simple DMA controller.

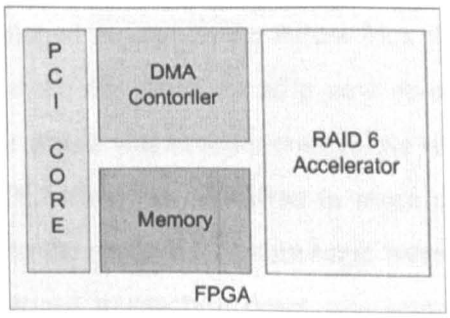


Figure 2. Hardware accelerator implementation

Our test computer was configured as follows:

- Opteron 64-bit processor
- Linux Fedora Core 3 Kernel 2.6.15.2
- Si3124 four port SATA controller
- Four 7200rpm 80GB DiamondMax Plus SATA disk drives
- Arran RAID 6 hardware accelerator on an Altera Stratix FPGA

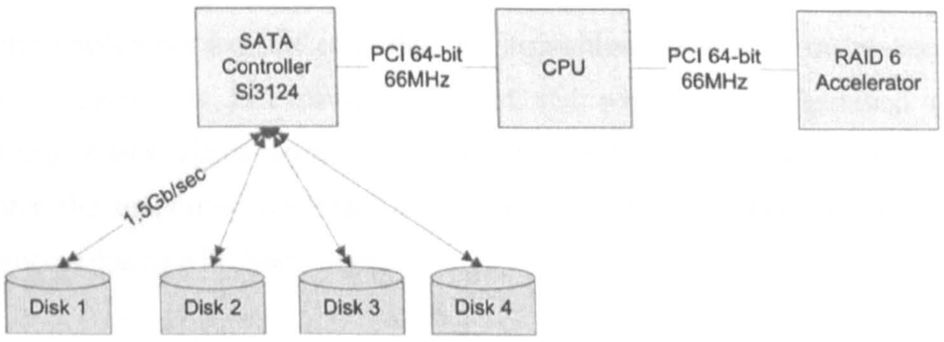


Figure 3. Test computer configuration

3 System Development

3.1 Hardware

The hardware design was implemented and tested in stages. The PCI IP core was developed and tested first. The remainder of the system was connected to the PCI IP core as required to enable more of the functionality of the system to be developed and tested. The hardware development was targeted at an Altera Stratix EPS125 on a PCI-X development board [1].

3.1.1 PCI IP Core

The original PCI core, developed for use on the Altera Flex PCI development board, was used as a starting point for the development of a new re-usable IP core. Based upon requirements for a PCI core which was broadly compatible with the Xilinx PCI core, the top level interface of the PCI core was modified to more closely resemble the Xilinx solution. Minimal changes to the original PCI core logic were made [2]. Configuration of the PCI core to support various transaction types was setup through the configuration registers.

The PCI core provides an interface to both PCI target and master functionality. The IP core functionality is controlled via the PCI configuration space registers at synthesis. This register space is setup via Verilog parameters.

Initial testing and verification of the PCI core was carried out independently of the remainder of the system. The target interface of the core was tested by verifying that the configuration space registers could be read and written to. This was carried out on a Linux based server via standard command line instructions. With these instructions it was possible to detect the PCI device and read and write the configuration registers successfully. Using SignalTap it was possible to observe the PCI bus transactions and verify that the supporting test bench accurately reflected PCI bus transactions and the setup process during a PC boot sequence.

Table 1. Basic Linux PCI command line instructions to allow PCI devices to be read and written

Command	**Description**
Lspci	Command for showing all PCI busses and the attached PCI devices.
Setpci	Command allows individual PCI device register to be read or written

Testing of memory read/write operations and bus mastering was carried out as part of the overall RAID 6 accelerator testing process.

3.1.2 RAID 6 IP Core

The algorithm selected for the initial design was a Reed-Solomon coding over a Galois Field GF(2^2) [3]. This limited the maximum number of storage devices to four, and therefore provided the optimal solution for a RAID 6 array with four storage devices. Although highly limited, the algorithm was the simplest and easiest to implement, requiring the fewest hardware resources. Multiple instances of the encoder and decoder logic could be instantiated in parallel to provide support larger or smaller data bus inputs. This design was re-used initially to minimise the amount of new design work required.

A replacement design was later implemented to provide support for up to 16 disk drives. Again the smallest implementation was selected performing all operations over a Galois Field GF(2^4). This modified RAID 6 codec was implemented differently from the original codec. The new RAID 6 codec performs operations on 4-bit wide input data. As data is read one disk at a time, the codec processes data sequentially. The intermediate checksums are then stored in local memory until the next disk's worth of data is available.

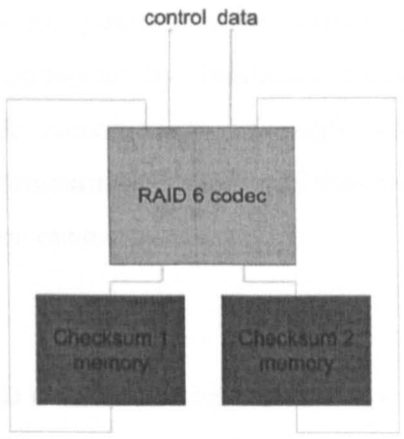


Figure 4. Shows the modified RAID 6 codec.

The codec requires the following control signal inputs:

Signal name	width	Description
data	4 bit	Input data to be encoded or decoded
Data_num	4-bit	Disk number for the current data input
Checksum_gen	1-bit	Checksum generation mode
Single_disk_failure	1-bit	Single erasure recovery mode
Double_disk_failure	1-bit	Double erasure recovery mode
ploc	4-bit	Number of disk holding the first checksum
qloc	4-bit	Number of the disk holding the second checksum
Bad_pos1	4-bit	Location of first erasure
Bad_pos2	4-bit	Location of the second erasure
rd_checksum	2-bit	Select the checksum/decoded data to read out

The codec was tested for all possible combinations for encoding and decoding operations. The testing was performed by simulation, encoding all possible input value combinations, simulating all combinations of single and double disk failure and recovering the lost data. By comparing the recovered data with what was originally input, it was shown that the functions operated correctly.

3.2 Software

Previously the software RAID 6 driver had been modified to utilise the RAID 6 hardware accelerator. Since this modified driver had been generated, several iterations of the unmodified driver had been released with a number of performance improvements. The latest driver was therefore used for development of a new device driver for the hardware RAID 6 accelerator.

The modified driver replaced software operations for checksum generation and recovery of failed disks with functions which configured the hardware accelerator to perform the same operations. The latest Linux Device Drivers book was also available at this time [4]. This proved to be invaluable in assisting to modify the software driver to support hardware. The software driver was found to use spinlocks throughout the driver to control atomic operations. Whilst this locking mechanism can be quicker than the alternatives in a purely software architecture, they are not conducive to operation with hardware and hardware interrupts. All the spinlocks within the software driver were replaced with mutexs which can be used in a hardware device driver. No other modifications were made to the software driver

3.3 Hardware software integration

Initial testing of the hardware with the modified software driver were positive. The driver allowed RAID 6 arrays to be built, written to and read back from. Unfortunately it was noted that data corruption would occur when writing files greater than 10 MB to the disk. Also the array was found to contain inconsistencies after being rebuilt

With the Stratix based FPGA development board it was possible to capture the transactions on the PCI bus to observe if the failures were a result of disk errors.

However, there was no easy means by which to determine if or when an error was occurring. To resolve this a test programme was written in C to check for different errors that could potentially be caused by the hardware. This programme generated known data in two regions of memory to represent two disks worth of data. These memory regions were made three times the necessary size with the data written to the middle section and the remainder of the data cleared to all zero. Two more regions of memory were created to store the checksum data. This additional memory was once more three times the size required and all cleared to zero prior to enabling the hardware. By making the memory regions larger than required it was hoped that any buffer underflows or overflows would be detected by checking if the memory region which should have contained all zeros contained data.

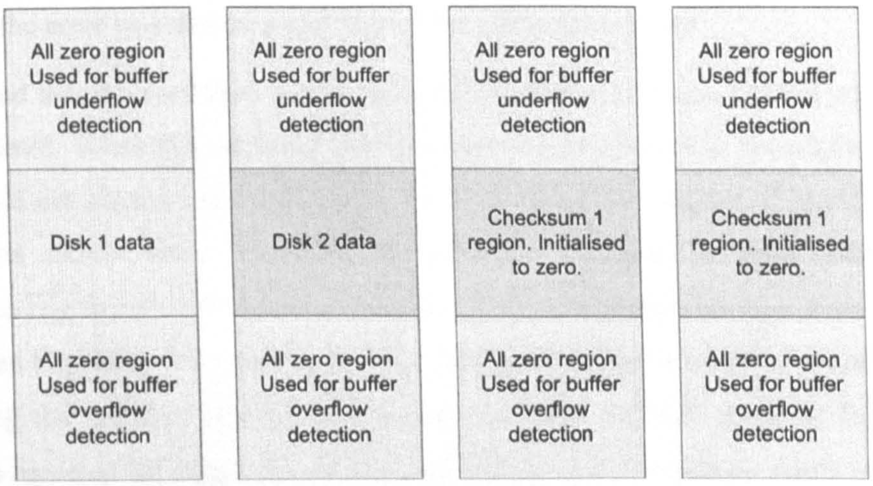

All zero region Used for buffer underflow detection
Disk 1 data
All zero region Used for buffer overflow detection
All zero region Used for buffer underflow detection
Disk 2 data
All zero region Used for buffer overflow detection
All zero region Used for buffer underflow detection
Checksum 1 region. Initialised to zero.
All zero region Used for buffer overflow detection
All zero region Used for buffer underflow detection
Checksum 1 region. Initialised to zero.
All zero region Used for buffer overflow detection

Figure 5. Shows the memory regions used to model the operations of the software driver. The hardware was expected to begin data accesses from the middle of the memory region to allow buffer overflow and underflow to be detected.

The data held at all disk 1 locations was the same. The same was done for disk 2. This allowed us to know the exact data expected to be written to checksum 1 and checksum 2. Once the memory locations were configured the hardware was initialised to perform the RAID 6 checksum generation function and to write back the generated data to the checksum memory space. The software test programme then verified the result and that no data was written to the underflow or overflow regions.

When run for 1000 iterations, there were found to be no errors recorded in the data received, nor were there any buffer overflows or underflows detected. Setting the test programme to run until an error was recorded eventually found a buffer overflow after several hours of operation. Repeated testing showed the same problem occurred over shorter and longer time periods. There appeared to be no correlation to the time the device was operational and the buffer overflow occurring.

Despite being able to capture the PCI bus transactions with SignalTap II, it was not possible to capture the buffer overflow on the logic analyser. The most likely reason for this being that the trigger was being missed due to the time it took to offload the previous bus transaction. The test programme was modified to add a wait period, of 10 milliseconds, between performing each checksum generation. This provided sufficient time to allow SignalTap to output all the PCI transactions that took place during the previous checksum generation operation. This programme was left running for several days until the error was detected and SignalTap captured the error.

It was found that the hardware was not properly flushing its output buffers when a target abort occurred. When this occurred, the hardware would attempt to resend the remaining data but did not always reset the counter for controlling the number of blocks of data to send to the correct value. Therefore the hardware attempted to send additional data beyond the region of valid memory locations, and occasionally to start from the wrong offset when beginning the next transaction. This was verified in simulation and corrected. Re-running the software test for all operations showed no fault could be found. These tests were repeated for single disk and double disk recovery. Again no errors were found.

This buffer overflow could have resulted in the system crashes which were observed in the original project. Returning to testing with the hardware device driver, the reliability of the array improved and it was possible to build and rebuild RAID 6 arrays successfully. However, when run for a period of hours the array would fail and occasionally the system would crash. By observing the error logs when the system was operational, it was found that a spinlock was being utilised by the device driver. On closer examination of the software driver, it was found that the spinlock was being inserted into the driver through a macro added via a header file. Replacing this section of code with our mutex, the device driver became completely stable and all testing and benchmarking tools operated successfully using the hardware accelerator.

3.4 Benchmark results

Performance was measured using Bonnie++, which measures data throughput achieved through a variety of read and write operations [5].

To determine the maximum throughput achievable using the test system the four drives were configured as just a bunch of disks (JBOD) , or RAID 0 array and running Bonnie ++. This showed a maximum block read throughput of approximately 209 MB/s and a maximum write throughput of approximately 197 MB/s. This individual drives were limited to read and write speeds of around 50 MB/s.

Repeating the performance checks with Bonnie++ using software RAID 6 and hardware RAID 6 showed a definite improvement in using the hardware accelerator over the software only solution when a double disk failure or single disk failure had occurred. The hardware accelerator also generated the RAID array around 5-10MB/s faster than the software only solution, and performed data recover at a higher rate also. The complete Bonnie++ results for three iterations in each mode for both software RAID 6 and the hardware accelerated RAID 6 are shown in the tables below.

It was noted the software was prone to unusual effects and suffered large drops in performance for short periods which resulted in low data throughput rates. This was not observed when using the hardware accelerator.

The hardware accelerator appeared from these results to be using massive amount of CPU time when performing the Bonnie++ benchmarking tests. Through the use of cron, the Linux resource utilistation monitoring programme, it was found that this CPU utilisation was actually being used by Bonnie++ and not the hardware device driver. This was found to be a result of Bonnie continually checking if the data had been processed or not. The actual CPU utilisation of the hardware driver was found to be around 3-4% during all operational modes. The software driver CPU utilisation was found to be around 33% higher during normal operation increasing to over 50% higher during double disk failures. This was more in line with expectations.

Table 2. Hardware results generated using Bonnie++

	Size (MB)	Sequential output						*Sequential Input*				Random Seeks		Num Files	Sequential Create						Random Create					
		Per Char		Block		Re-Write		Per Char		Block					Create		Read		Delete		Create		Read		Delete	
		Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU		Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU
RAID6hw_workinga	4000	31917	79	58573	25	22921	5	29321	69	59562	7	259.6	0	16	4603	100	+++++	+++	+++++	+++	4736	99	+++++	+++	13671	97
RAID6hw_workingb	4000	31963	80	59130	25	22553	5	29619	69	59463	7	251	0	16	4002	99	+++++	+++	+++++	+++	4137	100	+++++	+++	11548	98
RAID6hw_workingc	4000	32154	80	59507	25	22246	4	29668	69	59347	7	232.7	0	16	4092	100	+++++	+++	+++++	+++	4229	100	+++++	+++	11741	98
RAID6hw_1faileda	4000	28830	73	29170	7	16800	4	28854	74	74910	11	153.6	0	16	4857	99	+++++	+++	+++++	+++	5129	100	+++++	+++	14787	97
RAID6hw_1failedb	4000	28792	73	28908	7	16996	4	28791	74	74837	11	155.9	0	16	4788	99	+++++	+++	+++++	+++	5058	100	+++++	+++	14809	97
RAID6hw_1failedc	4000	28716	73	29334	7	16782	4	28786	74	75679	11	154.1	0	16	4735	100	+++++	+++	+++++	+++	5020	99	+++++	+++	14987	99
RAID6hw_2faileda	4000	30139	78	33527	8	17550	4	31573	83	76531	12	116.8	0	16	4631	99	+++++	+++	+++++	+++	4907	99	+++++	+++	14229	97
RAID6hw_2faileda	4000	30139	78	33527	8	17550	4	31573	83	76531	12	116.8	0	16	4631	99	+++++	+++	+++++	+++	4907	99	+++++	+++	14229	97

Table 3. Software results generated using Bonnie++

	Size (MB)	Sequential output						Sequential Input				Random Seeks		Num Files	Sequential Create						Random Create					
		Per Char		Block		Re-Write		Per Char		Block					Create		Read		Delete		Create		Read		Delete	
		Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU		Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU	Size K/sec	% CPU
RAID6sw_workinga	4000	32058	70	68427	14	22080	5	28554	67	59098	7	254.9	0	16	4000	99	+++++	+++	+++++	+++	4207	99	+++++	+++	12062	100
RAID6sw_workingb	4000	14313	31	66110	14	22320	5	29970	70	59161	8	244.9	0	16	2207	55	+++++	+++	+++++	+++	2134	50	+++++	+++	2952	25
RAID6sw_workingc	4000	30765	71	68706	14	22467	5	28638	69	59312	8	261.9	0	16	4013	100	+++++	+++	+++++	+++	4221	100	+++++	+++	11634	98
RAID6sw_1faileda	4000	14393	31	30734	7	17932	4	27279	67	92891	13	159.8	0	16	1551	33	+++++	+++	+++++	+++	1337	27	+++++	+++	7038	48
RAID6sw_1failedb	4000	29563	66	31188	7	18407	4	29824	75	92153	13	170.8	0	16	4117	99	+++++	+++	+++++	+++	4334	100	+++++	+++	12099	97
RAID6sw_1failedc	4000	28397	64	31859	7	18371	4	29727	74	97118	13	170.6	0	16	3961	100	+++++	+++	+++++	+++	4132	100	+++++	+++	11752	99
RAID6sw_2faileda	4000	13148	29	32884	7	17449	4	21580	64	67390	11	121.9	0	16	1606	38	+++++	+++	+++++	+++	1593	35	+++++	+++	5036	41
RAID6sw_2failedb	4000	26965	59	29740	6	16561	4	27441	83	70199	11	130.3	0	16	4043	100	+++++	+++	+++++	+++	4262	100	+++++	+++	11990	100
RAID6sw_2failedb	4000	27028	60	32150	7	17670	4	27732	82	72099	11	126.8	0	16	4772	99	+++++	+++	+++++	+++	5117	98	+++++	+++	15134	97

4 Product Solutions

This section introduces and discusses the benefits and risks associated with the development of several potential products based upon the hardware RAID 6 accelerator. Whilst the current focus of continued design work will be based upon the existing hardware platform using the PCI/PCI-x bus, the following options consider what a final product may look like.

4.1 Hardware accelerator

4.1.1 Product Overview

The current hardware accelerator may be developed into a commercial product providing hardware RAID 6 acceleration via a PCI-X bus. This should provide a performance boost over software in all modes of operation. The proof of concept implementation currently is limited by the PCI bus, moving to the higher speed PCI-X bus should ensure hardware consistently outperforms the software based RAID 6 algorithm.

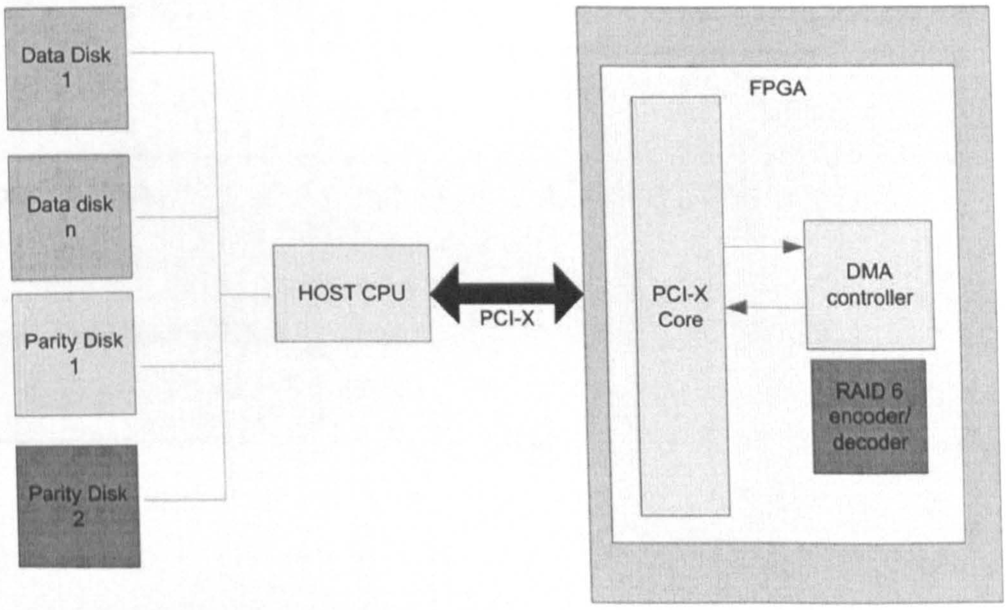


Figure 6. Shows the RAID6 hardware accelerator

4.1.2 Estimated Costs and requirements

To facilitate development, the purchase of an Altera Cyclone II (or other, equivalent, low cost FPGA) PCI-X development board would be required to qualify the existing design on these devices. This would also provide schematic diagrams for the board which could be used to reduce the design work required for the final product. Total cost for this board would be \$995[1]. Conversion from the current Stratix FPGA design to a cyclone based implementation should take one week.

The current PCI Core would have to be developed to support PCI-X operation to improve data throughput to maximise performance. This is estimated to take 3-4 weeks.

Development of the board schematics and layout may be carried out in house at A2E.

Overall BOM costs for the final product would be dominated by the FPGA and board manufacture costs. Worst case cost for the FPGA would be \$149[2].

Aim for a purchase price less than 50% of current RAID 6 controller cards.

Table 4. Commercially available hardware RAID controller card prices

	Part Description	Price
Intel SRCS16	6-port SATA RAID 0, 1, 5 controller	£244.97
Adaptec 2410SA	4 port SATA-2 RAID 0, 1, 5 controller	£322.93
Promise SuperTrak EX8300	8-port SATA-2 RAID 0, 1, 5, 6	\$370-400

[1] Prices from Arrow US website for full price purchase

[2] Prices from Arrow US for one off purchase

4.1.3 Risk Assessment

Both the PCI Core and RAID 6 hardware have been tested and verified already. Expansion to support PCI-X is not seen as a major risk as a development board for this is currently available. Also transition from a 64-bit PCI bus to PCI-X should be possible after boards have been manufactured. Use of Cyclone II development boards will allow the design to be tested and verified prior to manufacture and reduce risk of failure and reduce development time. The time required to develop this product is lower than any of the other options and will allow an early time to market.

RAID 6 controllers are currently available, based around Intel's IOP 331/333 processors, offering higher data throughput than currently achieved on the FPGA based accelerator, thanks to the integration of SATA/SCSI controllers and the RAID acceleration.

To maximise the available market for this product will require the development of Windows device drivers. Currently only Linux drivers are available for this hardware. Development of Windows drivers could be time consuming and complex.

4.2 NAS solution

4.2.1 Product Overview

A basic network attached storage device could be developed based upon the existing accelerator design and utilising a NIOS processor and Avalon bus to provide access via Ethernet and USB II ports.

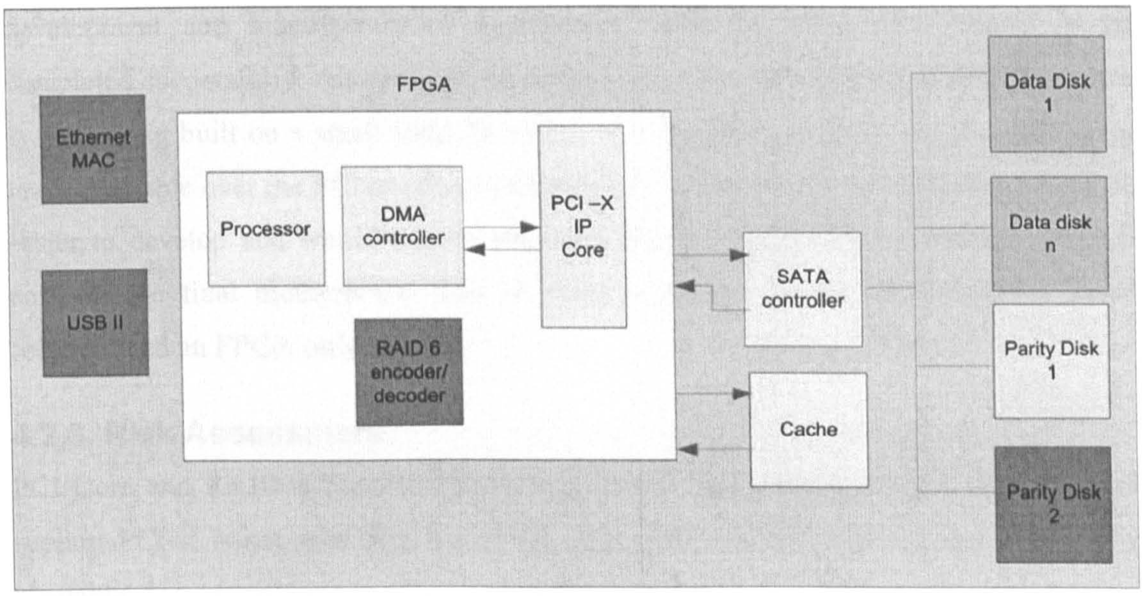


Figure 7. Shows a simple NAS device utilizing the RAID 6 accelerator and FPGA based system

An alternative system could be developed using a small form factor PC based system. This would allow a lower cost FPGA to be utilised and would require minimal changes from the existing system design.

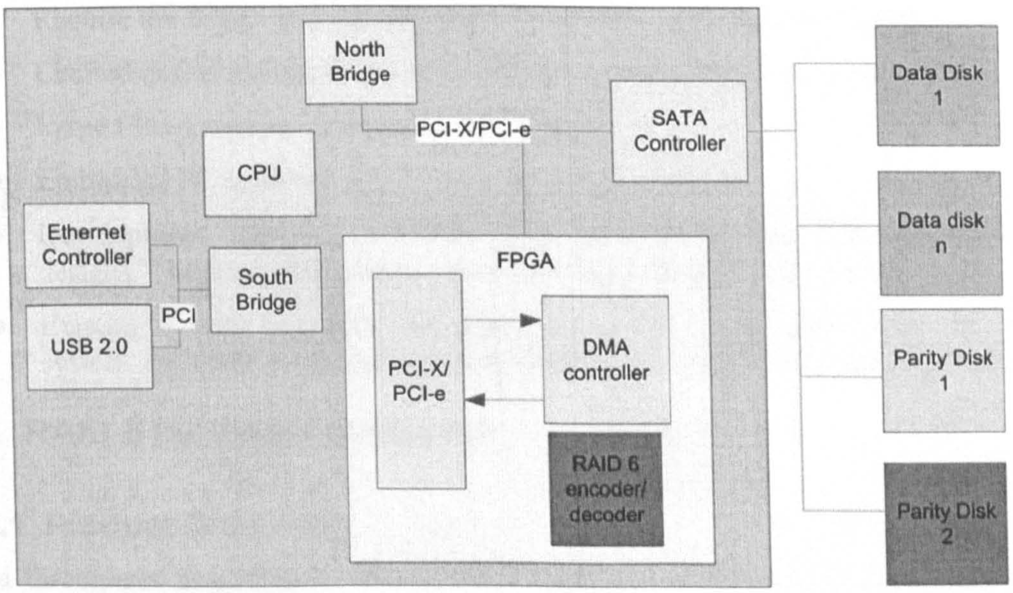


Figure 8. Shows an embedded PC system with on-board RAID 6 hardware acceleration

4.2.2 Estimated Costs and requirements

For the FPGA based solution, sections of this design could be tested using currently available cyclone development boards, however, this design will necessitate the development and manufacture of a prototype board to allow development to be completed successfully. Alternative architectures may also be considered with the entire system being built on a small form factor PC with hardware RAID 6 acceleration being made available over the PCI or other available bus. This alternative architecture would be easier to develop and would require minimal software and hardware changes. Overall costs of the final motherboard may be more expensive for an embedded PC based solution than an FPGA only solution.

4.2.3 Risk Assessment

PCI Core and RAID 6 hardware have been tested and verified already. Expansion to support PCI-X is not seen as a major risk as a development board for this is currently available.

Only one set of device drivers required for this solution minimising software development requirements.

FPGA based solution:

- Require the design and manufacture of a suitable development platform.
- Limited expansion capability of design due to high level of customisation.
- Large FPGA required increasing FPGA costs.
- Embedded PC solution:
- Development platforms available from Intel, AMD and VIA for embedded designs. These may be used to reduce development costs and times.
- Existing system has been developed and tested successfully on a PC based system, therefore necessitating minimal changes to software for this architecture.

4.3 RAID 6 hardware controller

4.3.1 Product Overview

Data throughput is limited by the read/write limits over the various buses on the PC system. Incorporation of hardware acceleration on the same expansion slot as the hard disk drive controller minimises this bottleneck.

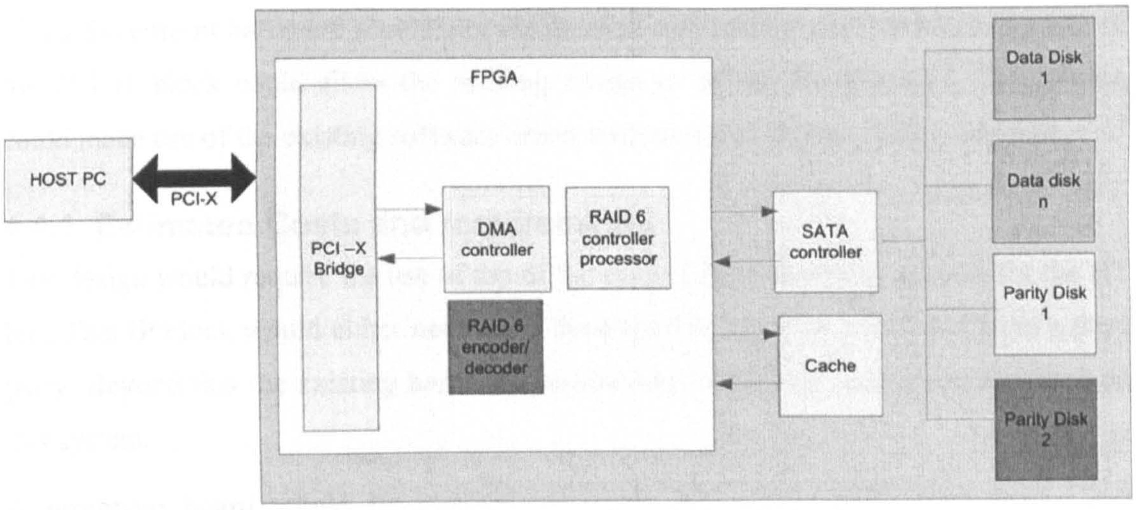


Figure 9. Shows a 4 port RAID 6 hardware controller

4.3.2 Estimated Costs and requirements

This design would require the use of an embedded processor on the FPGA with direct access to the hard disk drives and RAID 6 controller. Both processor and hardware accelerator would share access to DDR-RAM which would act as a local cache for data to be read/written from the hardware controller.

A prototype board would be required to verify the system and allow testing and performance characterisation of the hardware acceleration.

4.3.3 Risk Assessment

PCI Core and RAID 6 hardware have been tested and verified already.

Costs of a sufficiently large FPGA may prove prohibitive, further work will be required to determine what FPGAs will be appropriate.

Both Linux and Windows drivers would have to be developed.

4.4 Reconfigurable computing

AMD has made available its Hyper Transport (HT) interface which is used to allow direct access to the CPU and RAM on AMD based motherboards. The use of this bus would enable direct access to memory used by the processor and enable the hardware RAID 6 accelerator to vastly out perform the software based solution. This bus could also provide the bandwidth required to enable hardware acceleration of a wide number of algorithms.

Using the current hardware accelerator architecture and adding a HT IP block in place of the PCI IP block could allow the existing design to be rapidly deployed. This system could make use of the existing software driver with minimal changes being required.

4.4.1 Estimated Costs and requirements

This design would require the use of top of the range FPGA to enable support for the HT bus. This IP block would either need to be developed in house or purchased from a third party. Beyond this the existing hardware architecture should be readily implemented on this system.

A prototype board would be required to verify the system and allow testing and performance characterisation of the hardware acceleration.

4.4.2 Risk Assessment

The RAID 6 hardware and software driver have been tested and verified already.

The design and development of a suitable PCB may be prohibitive.

The price premium attached to this device could be sufficiently high to justify the design work, however, the potential market would be far lower than that from using a PCI-X/PCI-e add-in card.

No support for Intel based solutions.

5 Conclusions

The RAID 6 hardware accelerator has been shown to operate correctly and has been verified over prolonged time periods as being stable and functionally correct. The accelerator has been shown to be limited by the reliance on the setup and configuration by the host CPU and maximum throughput is being curtailed as there is no direct access to the hard disk controller by the hardware accelerator. The hardware accelerator reduces CPU overhead by between 33%-50% and out performs the software only solution in recovering lost data and during both single and double disk failures.

A number of products have been suggested based upon the hardware RAID 6 accelerator. The optimal solutions would be to design either a PCI-express based hardware RAID 6 controller card with direct access to the hard disk drives, or to develop an accelerator on the Hyper Transport bus to provide hardware acceleration in a manner similar to that provided by the PCI hardware accelerator.

6 References

[1] PLD Applications, PCIX sys User's Guide Version 1.1 rev0, PLD Applications, 2003.

[2] Xilinx Inc., *Hardware accelerator for RAID6 Parity generation / data recovery controller*, Application Note: Virtex-4 Family, April 19th 2006.

[3] S. Reed and G. Solomon, *Polynomial Codes over Certain Finite Fields, SIAM Journal on Applied Mathematics*, vol. 8, pp. 300-304, 1960.

[4] J. Corbet, A. Rubini, G. Kroah-Hartman, *Linux Device Drivers,Third Edition,* O'Rielly Media, Inc., 2005

[5] T. Bray, Bonnie++ documentation, URL available : http://www.coker.com.au/bonnie++/readme.html
Last accessed 14th April 2006

[6] D.A. Patterson, G.A. Gibson, R.H. Katz, *A Case for Redundant Arrays of Inexpensive Disks (RAID)*, Proceedings, ACM SIGMOD Conference, pp. 109-116, July 1988.

[7] J. L. Hennessy & D.A. Patterson, *Computer Architecture: A Quantitative Approach (Third Edition)*, Morgan Kaufmann Publishers, January 2003.

[8] P. Chen et al, *RAID: High-Performance, Reliable Secondary Storage*, ACM Computing Surveys, Vol 26, pp. 145-185, June 1994.

[9] M. Shooman, *Reliability of Computer Systems and Networks*, New York: John Wiley and Sons, 2002, pp.119-126

[10] Karp, M: *All about RAID, Network World Newsletter*, 7th July 2005, URL available:
http://www.networkworld.com/newsletters/stor/2005/0704stor2.html
Last accessed 22nd September 2005.

[11] M Blaum et al. *EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures*, IEEE Transactions on Computers, Vol 44, No 2, PP 192 -202, February 1995.

[12] C. Park *Efficient placement of parity and data to tolerate two disk failures in disk array systems*, IEEE Transactions on Parallel and Distributed Systems.

[13] Y. Nam, D. W. Kim, T. Y. Choe, and C. Park, *Enhancing write I/O performance of disk array RM2 tolerating double disk failures*, in International Conference on Parallel Processing, Aug. 2002, pp. 211--218.

[14] S. Reed and G. Solomon, *Polynomial Codes over Certain Finite Fields, SIAM Journal on Applied Mathematics*, vol. 8, pp. 300-304, 1960.

[15] Intel Corps, *Intel 80331 I/O Processor Datasheet*, Intel Corps, August 2005

[16] Xilinx Inc., *Hardware accelerator for RAID6 Parity generation / data recovery controller*, Application Note: Virtex-4 Family, April 19th 2006.

[17] J. M. Howie, *Fields and Galois Theory*, Springer, 2006

[18] Xilinx Inc., *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet DS083 (v4.5)*, October 10, 2005.

[19] Altera, *Stratix device handbook*, July 2005

[20] P. Mead, *Improving ASIC Design Verification using FPGAs and Structured ASICs*, IP/SOC 2006", December 6th-7th 2006, Grenoble.

[21] PCI Special Interest Group, *PCI Local Bus Specification*, Revision 2.3, March 29, 2002

[22] T. Bray, Bonnie++ documentation, URL available : http://www.coker.com.au/bonnie++/readme.html
Last accessed 14th April 2006

[23] Min-An Song, I-Feng Lan, and Sy-Yen Kuo, *An Area-Efficient Architecture of Reed-Solomon Codec for Advanced RAID Systems*, IEEE computing society, Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05), 2005.

[24] PCI SIG, PCI IDE Controller Specification Revision 1.0, 3/4/94

[25] AC&NC, URL http://www.acnc.com/benchmarks.html, Last Accessed 12th September 2005

[26] N. Brown, URL http://neilb.web.cse.unsw.edu.au/source/mdadm/ Last Accessed 12th September 2005

[27] Bray, T. *BONNIE user manual* ,URL http://www.textuality.com/bonnie/intro.html, Last accessed 12th September 2005

[28] IO Zone, URL: http://www.iozone.org/, Last accessed 12th September 2005.

[29] M.H. Jing et al. *A fast error and erasure correction algorithm for a simple RS-RAID*, 2001, pp333-338

[30] PCI Special Interest Group, *P1386.1/Draft 2.4*, January 12, 2001

[31] S/ Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*, IEEE Press, New York, 1994.

[32] Paul Luse, Mark Schmisseur, *Understanding Intelligent RAID 6, Technology@Intel magazine*, May 2005

Appendix VII:

Avalon bus based hardware RAID 6 accelerator

November, 2006

Overview: This document introduces the Altera Avalon bus specification and the benefits of migrating the hardware RAID 6 accelerator to this bus format. The hardware implementation is discussed and the testing and performance measurements are presented.

Michael Gilroy

EngD 4th Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

Having designed, tested and verified the operation of the hardware RAID 6 accelerator two options were considered viable for commercialising the design; selling the IP or designing and selling a piece of hardware which utilises the IP. To support both the development of a hardware RAID 6 controller and the sale of the IP it was decided to migrate the design to operate over the Avalon bus [1].

Two factors drove this decision. The first was the need for rapid development of the hardware RAID 6 controller with minimal new hardware to reduce the design time and complexity. By using the Avalon bus a large number of pre-verified IP blocks could be used in the design. The second reason, as Xilinx had released an equivalent IP block for their devices, it was considered beneficial to target the existing design at Altera devices and enable rapid and easy integration into the Altera design flow [2]. Soft IP blocks for Altera devices commonly offer both Avalon bus support as well as direct interconnectivity to the IP such as has already been produced.

2 Avalon based RAID 6 IP block

2.1 Avalon Bus Overview

The Avalon bus specification is a proprietary bus standard from Altera which provides peripheral to peripheral interconnect [1]. The Avalon bus specification defines the transfers and interconnect between a peripheral and a switch fabric. This switch fabric provides an interconnect structure between the peripherals on the bus to ensure any master can communicate with any slave without any prior knowledge of the interface for either.

The Avalon bus specification supports a variety of transfer types and a peripheral may use those which best suit its requirements. Additionally the peripheral may have any number of slave and or master ports. The Avalon bus operates synchronously and uses separate address, data and control lines.

2.2 SOPC builder Overview

SOPC builder is a graphical tool which enables system designers to integrate off the shelf intellectual property (IP) blocks and custom hardware on an Avalon bus based architecture [3]. Providing IP on the Avalon bus enables it to be packaged to be integrated with Altera's SOPC builder programme. The tools to package custom hardware in a form suited for use from within SOPC builder is included with the tool.

SOPC builder is closely linked with Altera's NIOS II processor as it enables the generation of systems utilising this processor core as well as providing access to a wide range of supporting IP blocks for various memory interfaces and other pre-verified IP blocks.

2.3 IP block design Design

The previously designed RAID 6 hardware accelerator was controlled by a DMA controller which was tightly coupled to the PCI bus controller interface[4]. Whilst this architecture was sufficient for demonstration purposes, it limited the reusability of the hardware accelerator. It was decided to modify this design to improve the reusability and structure of the IP block.

The RAID 6 codec, which coded data using a reed-Solomon codec operating over a GF(16) was left unchanged from the previous design. The modifications to be made were the integration of a DMA controller, DMA control registers and an Avalon bus interface into the IP block.

2.3.1 DMA controller

The DMA controller design had to be flexible enough to support the encoding and decoding of any combination of up to 16 disk arrays, as well as offering a ready solution for upgrading to support larger arrays in the future. The DMA controller has in the main to perform two main operations; read data from a memory location for each disk to be read, and write back either one or two encoded /decoded blocks to memory. In addition to the loading of data, the DMA controller ensures that the RS RAID 6 codec is configured correctly for performing the encoding and decoding operations on the received data.

The DMA controller is configured and controlled via a set of control registers. These provided sufficient information to ensure that any combination of disk accesses may be made. They also provide configuration data for the RS codec and control logic.

The four main functions of the control registers are:

- To provide memory address offsets for accessing data associated with the various disk drives in the system for read and write operations.
- To provide control information for the DMA accesses to ensure the correct number of memory locations are read and written as well as the correct number of bytes are accessed from each.
- To provide configuration data for the codec to ensure proper encoding/decoding operations are performed.
- To provide status information on the success or failure of the IP block or DMA controller.

The DMA controller operates under a number of assumptions. The first being that the data located at each memory offset is available from the clock cycle after the DMA controller is started and that the data located at each memory offset is of equal byte length. It is also assumed that all configuration and setup data for the RS codec will be provided prior to the DMA initiating operations, i.e. any disk erasures will have been detected prior to encoding or decoding being initiated. The IP block does not detect errors. Finally the DMA controller is configured to operate on a block size of 512 bytes or multiples there of. As a standard disk access should provide data blocks in 512 byte chunks as standard this value should always be valid. However, this may be changed by modifying a parameter in the HDL prior to synthesis.

2.3.2 Wrapper design

To support the design and development of Avalon bus based components, a simple bus wrapper was designed. As the Avalon bus offers support for a variety of transfer types, it was necessary to determine what support the hardware RAID 6 controller would need. The hardware RAID 6 accelerator requires a slave port to enable configuration of its DMA control registers. Both single and burst transfer support are desirable for the slave port. A single master port is required providing burst access to larger blocks of data for both read and write operations.

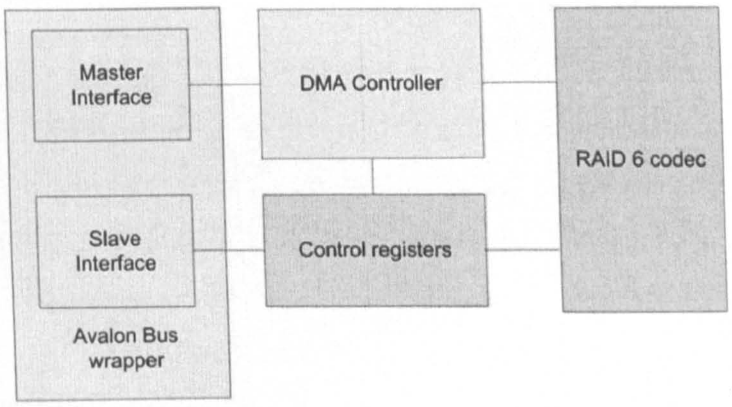


Figure 1. Avalon bus wrapper

The slave interface provides a simple state machine to control access to the control registers. Whereas the master interface merely renames and interconnects the Avalon bus signals to the DMA controller interface.

2.4 IP block generation

The completed Avalon bus based RAID 6 hardware accelerator was packaged from within the SOPC builder tool. This resulted in the hardware accelerator being made available as a reusable IP core which could be readily interconnected with any of the pre-existing IP blocks provided by Altera or third parties. In particular, the NIOS II processor may now be used to configure and control the hardware accelerator and used to verify its operation on the Avalon bus.

3 System design

The hardware system design was developed using the SOPC builder in Quartus II version 6.0. The target hardware platform was an Altera NIOS II development board with a Cyclone II EP2C25 FPGA [5]. This is a low cost development board providing access to many features, including: 16 MB flash memory, 16 MB DDR SDRAM, a Cyclone II EP2C25 FPGA, synchronous SRAM and serial port.

3.1 FPGA Architecture

To verify that the hardware accelerator had been correctly implemented on the Avalon bus, a simple system was implemented using SOPC builder. This system provides all the functionality required to use and test the hardware accelerator.

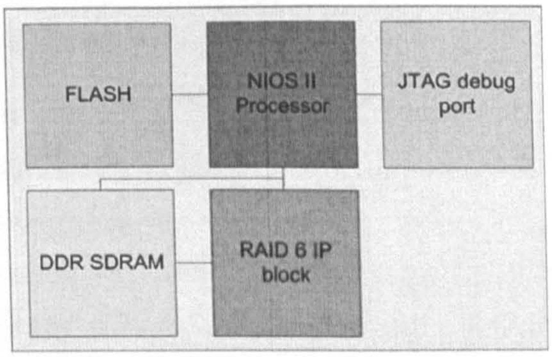


Figure 2. System architecture for testing the Avalon bus based hardware RAID 6 accelerator

3.2 Software Development

The RAID 6 IP block had to be shown to operate correctly for all supported Avalon bus transactions. A short programme was developed in the NIOS II Eclipse IDE which performed single and burst write and read transactions to the RAID 6 IP slave port. By writing and reading back these registers, the Avalon slave port functions of the wrapper were shown to operate correctly.

The operation of the wrappers master port was more complex to test and verify. During the earlier development phases of the RAID 6 IP block software verification of the RAID 6 operations had been performed by getting the hardware to encode a region of known data, and using this and the coded data testing each single and double disk erasure was tested and the returned data verified against the original data set. This same approach was taken to test the Avalon bus based implementation.

Regions of the DDR SDRAM were partitioned to represent two disk drives worth of data and two checksum disks. The RAID 6 IP block was then configured to perform checksum generations, and all combinations of single and double disk failures. Between each test the data contained on the DDR SDRAM was verified as being correct when compared to the original data. Confirmation of successful operation was output via the JTAG debug port.

The RAID 6 IP block successfully performed all these encoding and decoding operations demonstrating that the Avalon bus master port had been implemented correctly. This software programme was used to test larger disk arrays by increasing the number of partitioned regions on the DDR SDRAM to be used.

SOPC builder provides the ability to generate a simulation model based upon the hardware system and the NIOS II software. This simulation performs a number of checks to ensure that the Avalon bus specification is followed correctly. Having developed the software and demonstrated the RAID 6 IP block on the development board, the simulation was run to verify that no invalid transactions occurred on the Avalon bus. The test bench successfully completed all operations.

4 Conclusions

The RAID 6 hardware accelerator has been successfully ported to operate on the Avalon bus and packaged for easy integration into Altera's Quartus II software. Additionally, this IP block has been demonstrated on a standard Cyclone II development board interacting with standard Altera IP cores and hardware interfaces. This offers a simple, cost effective, demonstration platform for the technology and should allow future customers to rapidly integrate and test the RAID 6 hardware accelerator in their own designs.

The test system demonstrated here requires only interfaces to allow connectivity to a PC and a hard disk controller to enable the development and testing of a RAID 6 controller.

5 References

[1] Altera, *Avalon Interface Specification*, Altera, May 2005

[2] Xilinx Inc., *Hardware accelerator for RAID6 Parity generation / data recovery controller*, Application Note: Virtex-4 Family, April 19th 2006.

[3] Altera, *Altera Quartus II Handbook Volume 4: SOPC Builder*, Altera, November 2006.

[4] Gilroy, M., Review of Arran Stage II, April 2006.

[5] Altera, Nios II Development Kit, Cyclone II Edition (2C35), Altera, 2005.

Appendix VIII:

Avalon bus based hardware RAID 6 controller

December, 2006

Overview: This document discusses the implementation of a hardware RAID 6 controller using a system architecture based upon the Altera Avalon bus. The FPGA system architecture is presented along with a suitable hardware design platform and software interface.

Michael Gilroy

EngD 4th Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

Hardware RAID controllers provide support for one or more RAID levels in hardware which may be transparent to the host computer. The RAID controller provides access to hard disk drives and performs RAID calculations as data is passed to and from the hard disk drive. This removes the processing burden for any RAID calculations from the CPU to the controller and should provide better data throughput than that achievable using a software RAID algorithm.

A hardware RAID 6 accelerator IP block has been demonstrated and verified in hardware. A simple system is presented here for developing a hardware RAID 6 controller based on this IP block on an FPGA. By developing this system it is hoped that any potential bottlenecks in the system may be located and this knowledge may be used to assist in the commercialisation of the technology.

2 System Design

To support the development of a RAID 6 hardware controller a PCI Stratix FPGA development board is used in conjunction with a custom PMC daughter card. This daughter card provides access to a Serial ATA controller which may be accessed directly by the FPGA [1]. This system provides an interface to the host PC and access to four Serial ATA disk drives.

2.1 FPGA Architecture

The proposed FPGA system design is shown below in Figure 1, the hardware modules are all connected via the Avalon bus.

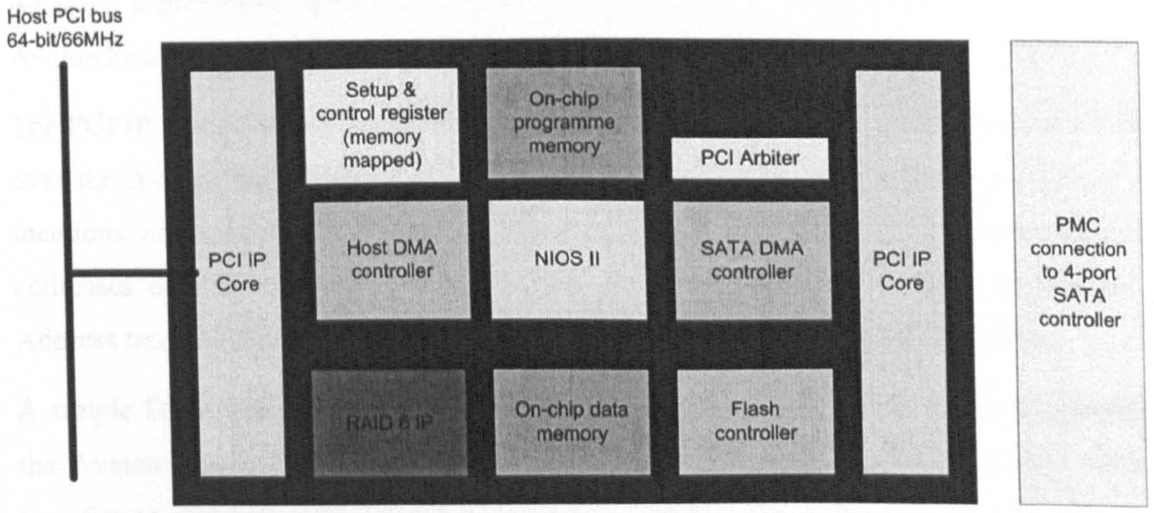


Figure 1. Avalon bus based hardware RAID 6 controller

2.2 IP block selection

Where possible, the IP blocks were selected from those available from within Altera's SOPC builder programme [2]. If unavailable, or unsuitable, then custom blocks were designed and packaged to be instantiated from within SOPC builder.

2.2.1 On-Chip memory

It had originally been hoped that the Flash and DDR SDRAM modules on the FPGA development board could be used to provide data and programme memory for the CPU as well as acting as a local cache for the hard disk data. However, it was found that the board was incompatible with the standard DDR SDRAM controller IP from Altera and that there was no Flash available for access by the FPGA. As a result on-chip memory was used as a substitute to enable development to continue. Whilst this may be faster than accessing the DDR SDRAM it, utilises additional system resources, and limits the amount of memory available for storing both software and data.

2.2.2 PCI IP block

The Altera PCI IP blocks were originally considered for use in this design to reduce design time and effort [3]. However, these IP blocks proved to use too much of the FPGA's resources and could not be made to fit on the FPGA with the rest of the system. As such, a previously generated custom PCI IP block was modified to operate on the Avalon bus.

The PCI IP block required a number of modifications to make it compatible for operating over the Avalon bus. A set of control registers were added which enabled the memory locations addressable via the PCI base address registers (BAR) to be translated to addresses on the Avalon bus. These addresses may be set by an Avalon bus master. Address translation between the Avalon and PCI bus ranges was also incorporated.

A simple DMA controller was added to the PCI core to perform data transfers between the Avalon bus and the PCI bus. A 512 byte data buffer was added to enable block transfers to progress without additional latency being added to the PCI bus transactions. To ensure that communication between the PCI bus clock domain and Avalon bus clock domain was handled correctly, a set of FIFO buffers were used to transfer data and control signals between both clock domains.

Transactions over the PCI bus to Avalon bus were tested and verified through simulations. These showed that the PCI IP block operated correctly for all read and write operations. Two instances of this PCI IP block were implemented on the FPGA. The first, operates at PCI 64-bit 66 MHz and provides communication between the host PC and the FPGA. The second operates over a PCI bus at 32-bit 33 MHz, this provide access to a Silicon Image Si3114 four port SATA controller.

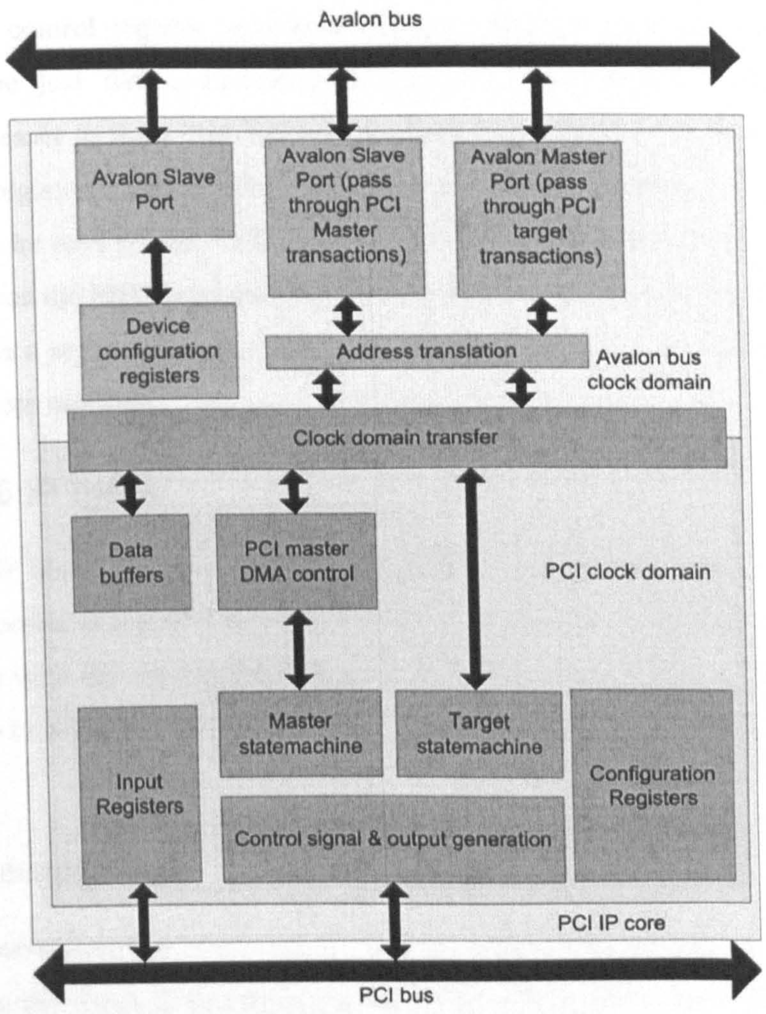


Figure 2. PCI IP block converted to run on the Avalon bus

2.2.3 PCI bus arbiter

The PCI bus arbiter which was generated grants control of the bus to each device which has asserted its request line in a round robin fashion. The arbiter is required to enable either the FPGA or SATA controller to gain control of the PCI bus interface connecting. At the PC end of the system arbitration, will be handled by an arbiter on the motherboard.

2.2.4 Setup and control registers

The setup and control register provide a memory interface which appears to the host computer to be just like a normal PCI ATA disk controller's memory interface. Read/write accesses to these registers by the host PC are handled as would normal PCI ATA control registers. Additionally, any accesses to these registers are loaded into a FIFO buffer to be read by the NIOS II processor to determine the correct response. The NIOS II accesses the FIFO registers over the Avalon bus. The NIOS has direct access to the registers via a separate Avalon slave port and can ensure the correct response is input to the appropriate registers.

2.3 RAID 6 IP block

The RAID 6 IP block has been verified and tested on the Avalon bus already. Although designed to operate as a RAID 6 codec, RAID 5 is supported by default as it is simply a RAID 6 array with the second checksum erased. Support for RAID 0 and RAID 1 had been hoped to be added by making the NIOS II processor handle the necessary operations in software.

2.4 DMA controllers

The DMA controllers are the standard Altera DMA controller IP blocks. These are configured by the NIOS II processor to transfer blocks of data to and from the on chip memory across the PCI bus.

2.5 NIOS II processor

The NIOS II processor is a 32-bit microprocessor from Altera which operates over the Avalon bus on their FPGAs [6]. In this system, the NIOS II processor was envisaged as providing the overall system control. The NIOS II would respond to disk accesses from the host PC by issuing read and write commands to the SATA controller, initialising RAID 5 or RAID 6 array calculations on the hardware accelerator, performs and manages RAID 0 and RAID 1 in software.

3 Simulation and synthesis

The PCI bus IP blocks and the RAID 6 IP blocks were verified and tested through a series of test benches. Both the PCI core and the RAID 6 IP blocks had previously been tested and shown to operate in hardware. The remainder of the system it was hoped could be tested in hardware on the development platform.

During synthesis of the design, it was found that the system was too large to be implemented on the FPGA. A number of changes were attempted to make the FPGA fit. However, it was impossible for the hardware controller to be made to fit on the FPGA. As a result of this and time constraints the hardware development has been halted.

4 Software Development

4.1 Host PC device driver

By designing the hardware RAID 6 controller to appear as a standard disk controller to the host PC it is hoped that no device driver should be necessary to enable read/write accesses to be performed to the hard disk drives attached via the RAID controller.

The selection and configuring of the hardware controllers RAID level should be made possible by developing a device driver to initialise the hardware controller to operate under the desired RAID level. This was envisaged as having to be run only once when the array was first configured with the hardware controller storing the configuration in an EEPROM or other such device.

In addition to configuration of the array, the device driver was also to detect warnings from the hardware controller when a disk failure or other error occurs. These messages should then be passed on to the user .

4.2 Hardware controller software driver

The hardware controller through the NIOS II processor was required to provide support for following operations:

- Determine the nature of received ATA commands.

- Perform read/write accesses to the hard disk drive
- Perform read/write accesses over the PCI bus retrieve/return data
- Configure RAID 6 operations
- Determine if an erasure has occurred
- Report disk errors to the operating system.

This programme was to be developed in C and slowly built up and tested in hardware to ensure all operations could be handled correctly by the NIOS II processor.

5 Analysis

A hardware development platform and FPGA system architecture for testing a complete hardware RAID 6 controller has been shown. Due to a lack of resources on the FPGA it was not possible to test the system and determine the data throughput and performance of the hardware controller.

The software requirements for the hardware controller have been examined and presented. It is hoped that for basic disk accesses and enabling testing of these, no software driver would be required for the host PC. This should enable the hardware controller software to be developed and tested more quickly than if a complete software device drive had to be developed for the PC. This should also help establish the exact cause and location of any hardware-software integration problems more rapidly as there should only be one software programme to debug at a time.

To enable development to continue, a number of solutions that may fit on the available FPGA are proposed. The RAID 6 controller may be removed from the design and the hardware controller could be used as a hard disk controller to determine the maximum throughput achievable. The PCI bus interface to the host PC may be removed and the performance of the RAID 6 IP block for encoding the hard disk drive may be measured. Between these two systems the actual data throughput for the RAID 6 controller may be extrapolated.

6 References

[1] PLD Applications, PCIX sys User's Guide Version 1.1 rev0, PLD Applications, 2003.

[2] Gilroy, M., Project Barra, October 2005.

[3] Altera, PCI Complier User Guide, Altera, December 2005

[4] Altera, Avalon Interface Specification, Altera, May 2005

[5] Xilinx Inc., Hardware accelerator for RAID6 Parity generation / data recovery controller, Application Note: Virtex-4 Family, April 19th 2006.

[6] Altera, NIOS II processor reference book, Altera, December 2005.

Appendix IX:

Project Barra

October, 2005

Overview: This document introduces the work undertaken by the Research Engineer to design a PMC daughter card to provide access to a SATA controller via a PCI bus to the hardware development board. Also presented is the work undertaken to support and provide guidance to a group of MSc students in a project which was to test the PCB and expand the RE's PCI IP block to offer PCI-X support.

Michael Gilroy

EngD 3rd Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

A PCI-X FPGA development board has been made available for design and testing of a PCI bus based RAID 6 accelerator. This development board also provides a standard PCI Mezzanine Card (PMC) interface which allows small daughter cards to be added to the existing board and connected via a PCI bus interface [1], [2].

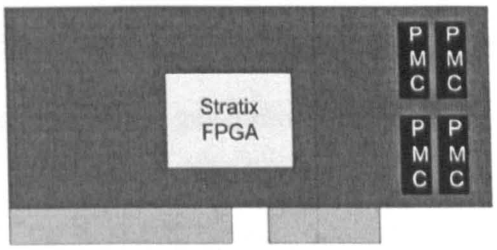


Figure 1. Shows the PCI-X development board and PMC connectors

To assist in the development of a RAID controller card it is desirable for the RAID accelerator on the FPGA to be able to access the hard disk drives directly. To achieve this functionality, it is proposed that a PMC daughter card be designed to provide access to a PCI Serial ATA controller. Additionally, it was decided that the daughter board should provide the means by which additional logic could be added to the design. To this end a mini-PCI connector was to be added to the daughter card.

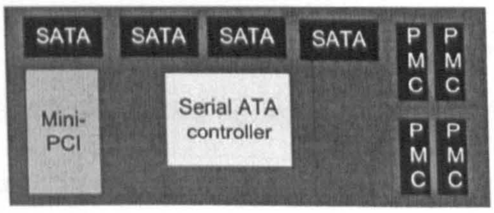


Figure 2. Proposal for PMC daughter card

2 PCB design

The schematic design for the daughter card was carried out using Orcad 10.1. The design was implemented such that either the min-PCI connector or the SATA controller would be attached. Selection of the correct device is performed by populating the correct pull-up resistors on the PCB. A Silicon Image Si3114 was selected as the SATA controller [3]. This device was provided as a free sample by Silicon Image for this project. The Si3114 provides connectivity to four first generation SATA ports via a single PCI bus connection.

To allow the PCB to connect to the PCI-X development board, the PMC connectors were placed on the bottom of the PCB with the SATA controller, SATA connectors and min-PCI connector all on the top of the board.

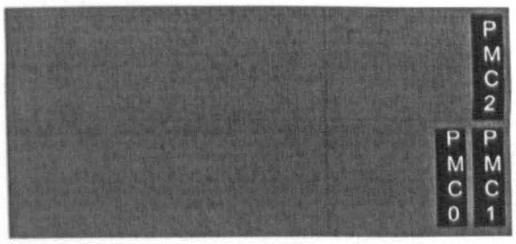


Figure 3. Daughter card PCB bottom view

Three PMC connectors were added to the PCB. Only PMC0 and PMC1 contained electrical connections. PMC2 was added to provide extra stability when the board was connected to the host board.

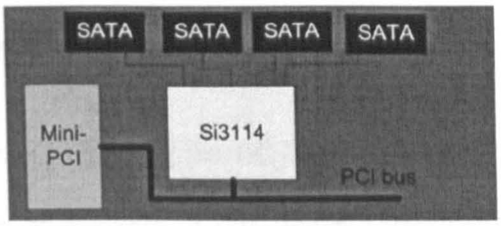


Figure 4. Daughter card PCB top view

The daughter card was implemented on a 6 layer PCB with the layout being performed in house by A2E. Two daughter card PCBs were manufactured and were hand soldered by the RE.

3 MSc Student Project

At the time this design was beginning the ISLI had requested that A2E provide an MSc project for a small group of students. It was decided that the students would verify the previously developed PCI IP block, modify it to operate as a PCI-PCI bridge and using the daughter card would demonstrate the operation of the device by connecting to a mini-PCI based wifi card. If they had time it was hoped that they could expand the design to function operate on a PCI-X bus.

Three students undertook this project, two handling hardware design and one handling the supporting software. It had been hoped that the students would be able to produce a functioning PCI-PCI bridge in a short time period to allow early access to the mini-PCI connector and perform testing of the daughter card PCB. Unfortunately the students performing the hardware design struggled to understand what they were to achieve and never managed to complete the PCI-PCI bridge, nor the expansion to add support for PCI-X. The software student managed to produce some useful tools for automatically configuring the PCI core configuration registers.

4 Conclusions

A PMC daughter card has been designed and manufactured to provide additional functionality to the FPGA development board. This board remains untested. The student project was unable to complete the expansion of the PCI IP block to support PCI-X or act as a PCI to PCI bridge. However, the students did help to increase the confidence in the reliability of the PCI IP block.

5 References

[1] PCI Special Interest Group, *P1386.1/Draft 2.4,* January 12, 2001

[2] PCI Special Interest Group, *PCI Local Bus Specification*, Revision 2.3, March 29, 2002

[3] Silicon Image, SiI3114 PCI to Serial ATA Controller datasheet, August 2003

Appendix X:

Customer project

October, 2004

Overview: This document introduces the work undertaken by the RE during a customer project. This project was undertaken to enable the RE to develop the skills required to specify, design, manufacture and test PCB board designs. This project provided the RE with the opportunity to carry out all these tasks as part of a small design team at the sponsoring company.

Michael Gilroy

EngD 3rd Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

1.1 Overview

This project provided an appropriate learning opportunity in the use of schematic design entry, PCB board layout and manufacture, the processes involved in selecting, purchasing and designing with various circuit components as well as allowing close interaction with the other members of the hardware design team in A2E.

The Accelera SP-3000 series project was carried out by A2E on behalf of one of their customers, Aspex Semiconductor, to upgrade their existing SP-2000 Accelera series.

1.2 Overview of Accelera

Aspex employed A2E to develop the next generation of their Accelera product line. Aspex Semiconductor's current Accelera product consists of a PCI based development system to allow software and system development by their customers of their software programmable Linedancer processor.

The Linedancer is described by Aspex Semiconductor as:

"Aspex' Linedancer™ device is a 100% software programmable processor which offers unmatched performance to deliver unique flexibility and scalability. Linedancer™ delivers stellar performance per dollar compared to existing solutions." [1]

The current generation of Accelera board, the SP-2000 series, was based around a standard PCI card providing:

- Installation into desktop PCs, workstations and servers
- PCI Mezzanine Card (PMC) Interface [2]:
    - High speed external I/O direct to the card eliminates I/O bottleneck on main system PCI bus
    - Allows the addition of customer specific daughter cards for rapid system prototyping
- Local Data Store:

- Synchronous SRAM for maximum performance or synchronous DRAM for maximum density

Applications of the Accelera board include: multimedia processing, signal, image and video processing, and volumetric visualisation and 3D graphics. All of the target applications require high bandwidth data transfer.

1.3 Summary of Accelera Upgrade

The initial upgrade requirements included increasing the Accelera board support from two Linedancer modules to four, providing DMA support to the PCI bridge and add additional, larger, on board DDR SDRAM replacing the current SRAM and SODIMM sockets. The initial design proposal suggested the use of an FPGA to provide the PCI bridging mechanism between the PCI bus and the Linedancer modules.

During further discussions it was decided that a PCI-X based system would be more appropriate and at this stage it was decided that an Intel chipset, the IOP315, should be used [3], [4]. This provided a PCI-X-to-PCI-X bridge with built in DMA, message queuing, an X-Scale processor and additional features not in the original requirements such as Gigabit Ethernet support.

It was also decided to split the design cycle into two phases; the first developing a demo board, the SP-3146 and the second, developing the completely upgraded system for the SP-4000 series. This document details the design and development of the SP-3146 and SP-4000 series.

2 Hardware Design & Implementation

2.1 Overview

The hardware architecture was based upon the SP-2000 series development platform with following enhancements:

- Quad Linedancer capability each with *discrete programme and discrete* data memory
- PCI-X bridge with 4-channel DMA provided by Intel IOP315 processor chipset
- Use of additional functionality available from IOP315 to provide:
- Flash memory
- Additional ARM compliant processor with memory

The platform upgrade was to be implemented on a standard PCI-X form factor PCB.

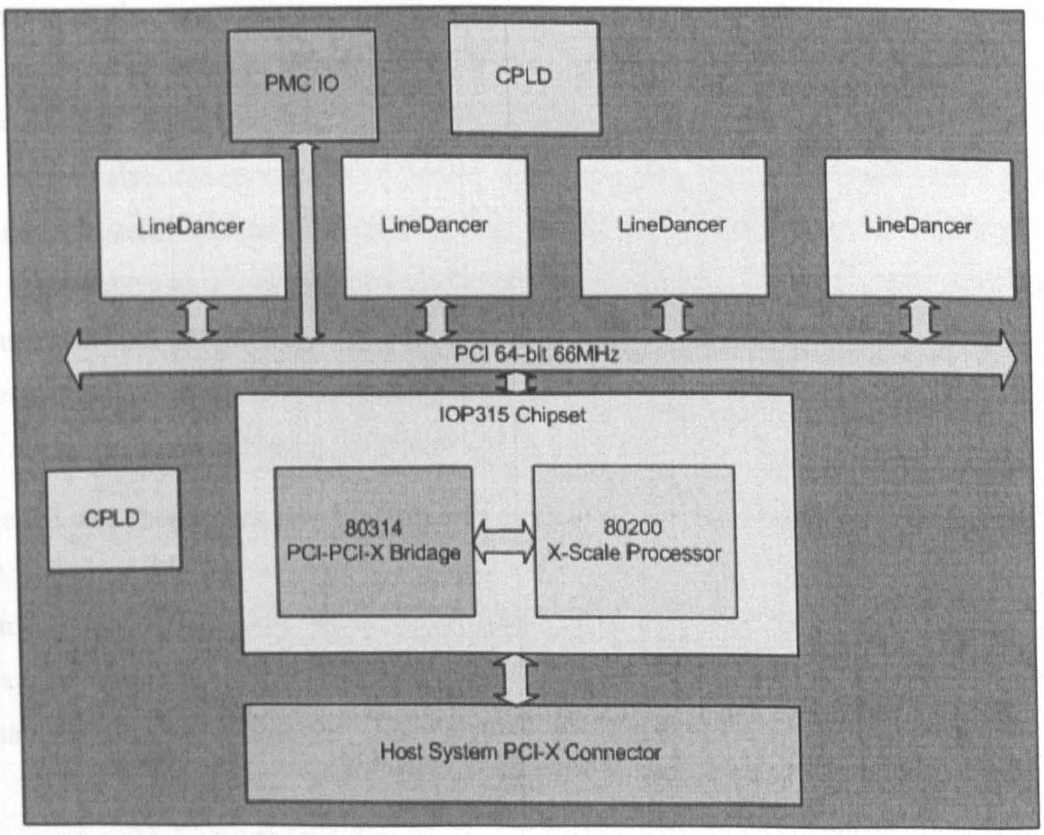


Figure 1. Shows the proposed Aspex PCB architecture.

2.2 Design Process

The design progressed through the following stages:

- Specification
- Schematic entry
- Layout
- Manufacture
- Test

The initial specification was provided by the customer. Working with the other members of the design team, the RE was responsible for component specification and design of large areas of the final board. The RE was also responsible for initial contact with the layout engineers and developing an estimate of the final board layout.

The system design was based on the utilisation of off-the-shelf components and re-using as much of the existing board design as possible to allow a rapid and easily verifiable system to be developed. The two Intel devices were used as the starting point for new design work. The schematics for the 80314 were not available from Intel at the start of the design, also, the specification for this chip changed as the project developed as Intel found new issues and tried to resolve them. This caused a number of difficulties during the design entry as the chip specification was changed from allowing the PCI-X bridge to connect to both 5V and 3.3V PCI slots to only 3.3V. This necessitated a redesign of the boards PCI connector to ensure the keying for 5V support was removed. It also limited the use of the board for the customer.

The RE was responsible for the schematic entry of the entire design with the exception of the power supplies. All work was subject to on-going review. Minimal changes were made to the design of the board when compared with the previous revision. The major changes were the new Intel chipset, upgrading to DDR-SDRAM, and utilisation of smaller passive components.

Once the major components from the design had been entered estimated layouts were produced to guide the board layout engineer. A selection of these are shown below. At this stage the design was still utilising SODIMM RAM modules.

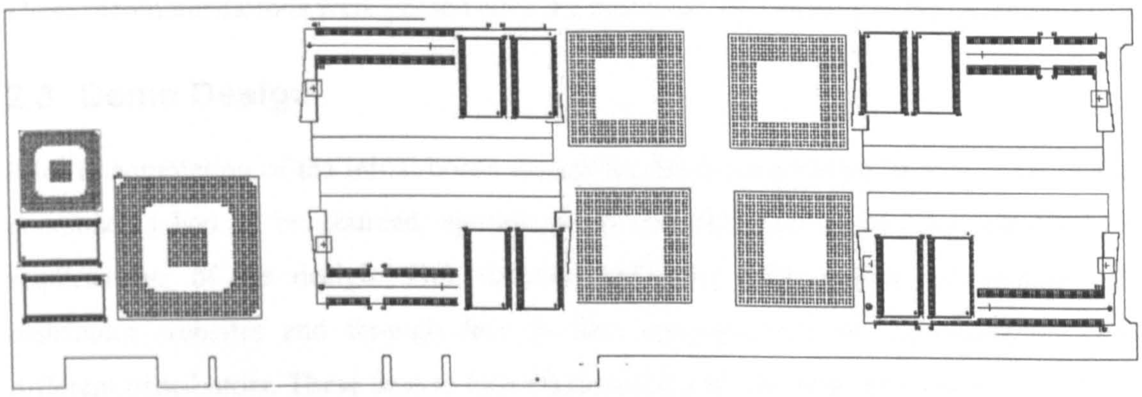


Figure 2. Diagram 2. Layout proposal 1

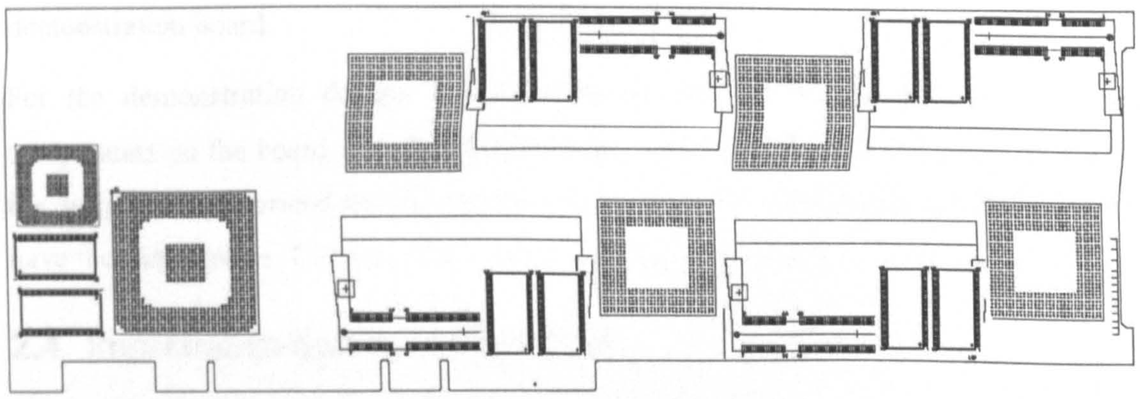


Figure 3. Layout proposal 2

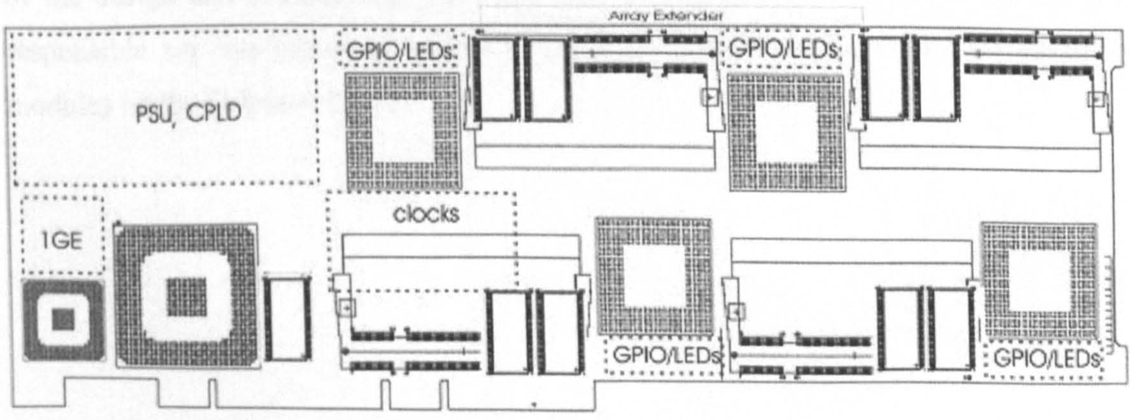


Figure 4. Initially preferred layout for design

Through discussions with the layout engineer *and after an initial* placement of the major components it was found that the design would not fit on a standard *PCI-X PCB. It* was therefore necessary to reduce component size for passive components and RAM modules. These recommendations were passed onto the customer who agreed to the changes.

2.3 Demo Design

Prior to completion of the initial board design for the demonstration boards, a number of components had to be sourced, specifications checked and verified as matching the requirements of the design. This involved searching both online via supplier and distributor websites and through face to face discussions with representatives from different distributors. These face to face discussions with the Intel representative allowed us to acquire the Intel IOP315, Flash and RAM modules in time to build the demonstration board.

For the demonstration design, it was necessary for the boards to have all of the components on the board though not necessarily connected. To this end an auto-route of the design was performed and the boards built on time. The demonstration boards did not have the Linedancers, CPLDs, JTAG, or memory modules routed.

2.4 Implementation of the SP-4000

The implementation of the final SP-4000 board required minimal changes from the demonstration board. The majority of the work carried out at this stage was verification of the design and constraining the board layout. In addition to this work the RE was responsible for implementing various setup and control functions for the Linedancer modules on the on board CPLDs.

3 Testing

A small number of test boards were manufactured to allow *initial testing* of the design to be undertaken. These PCBs were populated with only one fully functioning Linedancer. During testing a number of issues with the design were found. A number of the PCI signals should have had pull up resistors for those instances when a Linedancer was not populated. This was found by analysis of the PCI bus and observing incorrect bus sequences. The power-up sequence was also an issue and was resolved through a modification to the populated resistors on the board.

The customer provided a number the software programmes to debug the devices. This testing with software located highlighted an issue with the JTAG programming line. The reset line was constantly on due to a missing pull-up resistor. Resolving this issue allowed the design to be verified as operating correctly for all the supplied tests.

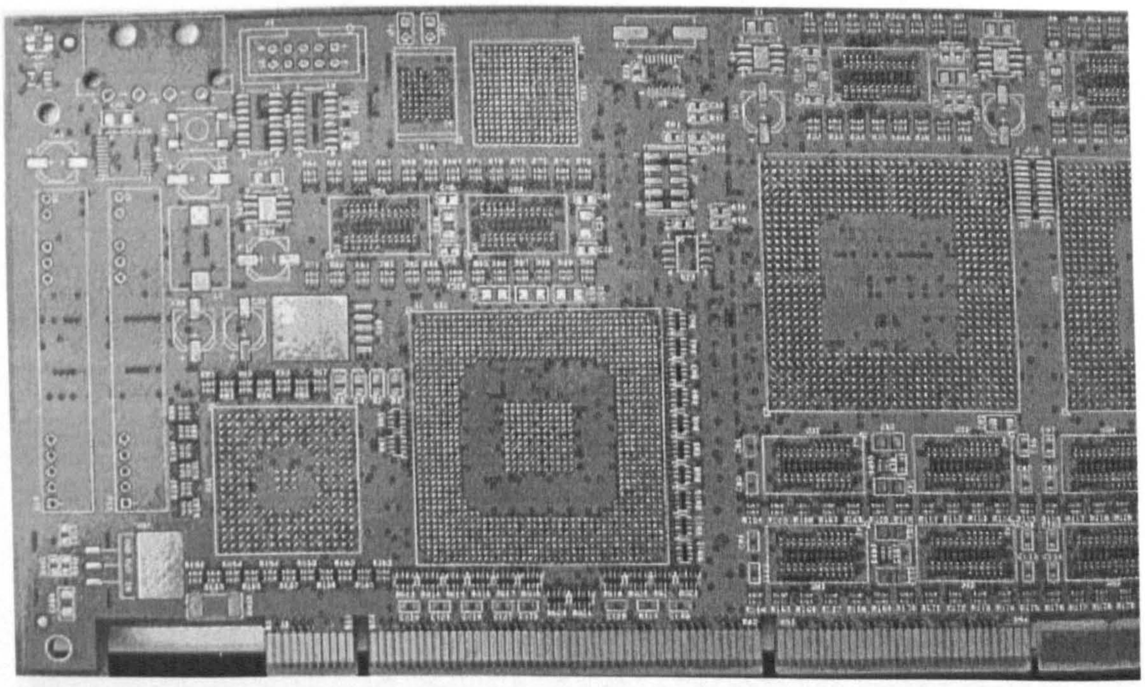


Figure 5. Showing the top side of the final board layout

4 Analysis Of Project

This project provided the opportunity to learn how to develop a hardware design from an initial specification through to manufacture and test. The RE was developed the skills required to use Orcad Schematic capture, provide design constraints, and follow naming conventions and structure that enable a design to be carried forward to the layout stage rapidly and successfully. The RE was also gained an insight of the communication necessary between those involved in each stage of the project to ensure a smooth and efficient transition from stage to stage in a projects life cycle.

The project was successful in that the board was built and delivered on time. The design could have been improved if the issues such as missing pull-up resistors had been spotted prior to manufacture. It would also have been easier to complete the design successfully had the specification updates from Intel been available prior to the project starting.

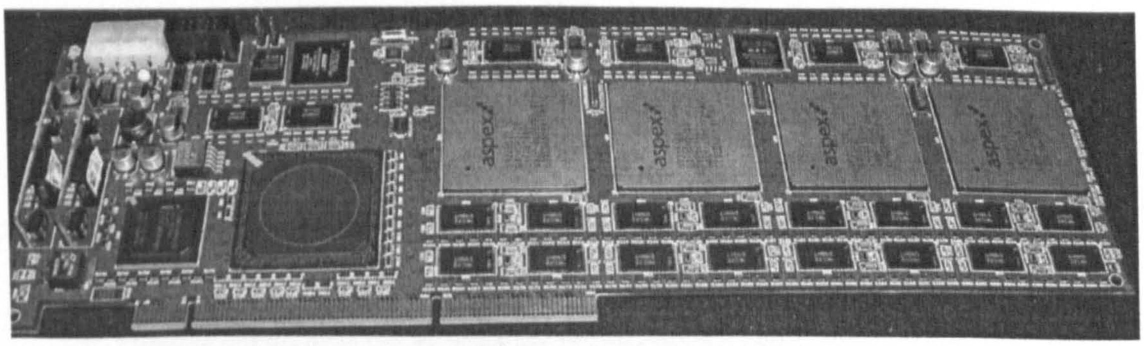


Figure 6. Diagram. Showing the top side of the populated design

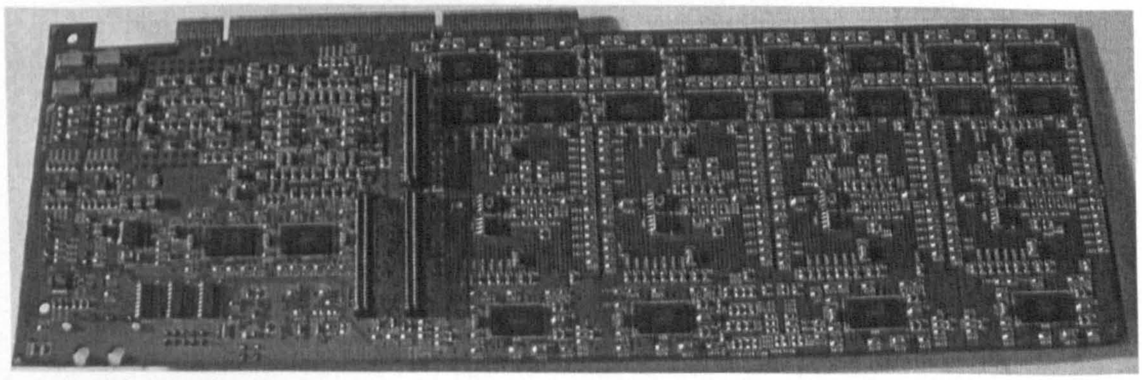


Figure 7. Showing the bottom side of the populated design

5 References

[1] Aspex Semiconductors, *Linedancer processor* product brief, June 2004

[2] PCI Special Interest Group, *P1386.1/Draft 2.4*, January 12, 2001

[3] Intel, GW80314 I/O Companion Chip datasheet, July 2004

[4] Intel, Intel 80200 Processor based on Interl XScale Microarchitecture Datasheet, January 2003.

Appendix XI:

Disk Storage Market Analysis as at June 2004

June, 2004

Overview: This document discusses the disk storage market as of June 2004 based upon freely available data. A short analysis of the market and the potential for a RAID 6 hardware controller is then made.

Michael Gilroy

EngD 2nd Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

The disk storage market suffered a steep decline between 2000 and 2001 from which it is still recovering [1]. The disk storage market is defined by IDC as systems including three or more hard disk drives. It is our belief that this market is the prime target market for a RAID 6 based storage system.

2 Market Leaders

The major players in the storage market place based on revenue as of Q3 2003 are detailed below:

Table 1. Disk storage systems market shares based upon worldwide revenue at Q3 2003. Reults from IDC.

Company	Market share by revenue (%)
EMC	28.9%
HP	25.6%
IBM	11.5%
Network Appliances	8.6%
Dell	6.9%
Hitachi	6.5%
Sun Microsystems	3.8%
Others	8.2%

3 Trends

3.1 Market Growth

The disk storage market is recovering from a decline of approximately 20% from 2000-2001 [1].

- Revenue for Q3 2003 is down approximately 0.3% when compared with revenue in Q2 2002 [1].
- Total disk capacity sold Q3 2003 increased 36% with a price drop per megabyte of approximately 30% [2].

3.2 Market value

The disk storage market is worth approximately $19.5 billion a year [2]. The external RAID market generated revenue of $2.7 billion from Q3 2002 to Q3 2003 [3].

4 Analysis

The storage market has a large number of major players vying for dominance within the market. Nonetheless, smaller companies continue to control just over 8% of the market, or just under $2 billion dollars a year in potential revenue. Entry to the market is therefore expected to be viable as the market offers opportunities for both large and small vendors.

Whilst growth has recently been on the decline this has been a direct consequence of a decrease in the cost of hard disk drives, rather than a decrease in storage requirements. Based upon the fact that storage requirements continue to increase, it is reasonable to assume that the need for reliable storage solutions will increase also.

RAID storage solutions have shown a noticeable growth from 2003 to 2004, however, there are not as yet widely adopted RAID 6 storage solutions from the major vendors. It is assumed that this will not remain the case indefinitely. It is therefore important to aim to deliver a solution over the coming year to provide a sufficient lead over the major players in developing a hardware solution. This will permit the hardware solution to be sold to the major players whilst their competitors develop their own solutions.

5 References

[1] Coleman, L, InfoStor, *Storage market shows signs of turnaround,* February 2004, URL Available:
"http://www.infostor.com/Articles/Article_Display.cfm?Section=ARCHI&ARTICLE_ID=199341&VERSION_NUM=1&p=23"
Last Access: June 2004

[2] Elliot King, *Storage market remains flat, competition fierce*, December 15th 2003. URL Available:
http://www.winnetmag.com/Article/ArticleID/41180/41180.html
Last Accessed: June 2004

[3] Jaring Internet Magazine, *Network Appliances gains external market share in external storage market*, 18th June 2004
URL Available:
http://www.magazine.jaring.my/2004/storage.html

Appendix : XII

Disk Storage Market Review

April, 2006

Overview: This document discusses the disk storage market as of April 2006 and examines the trends and potential for RAID 6 based products.

Michael Gilroy

EngD 4th Year

Industrial Sponsor: A2E Limited

Table of Contents

1 Introduction

This document discusses the current trends and performance of disk storage systems, to provide analysis for the marketing potential of A2E's RAID 6 storage solutions. The proposed solutions are for the development of a PCI based RAID 6 hardware controller with on board SATA controller, or RAID 6 implemented over the AMD HyperTransport bus. We define the market and market segments of interest and focus upon the market trends of the major vendors to determine a set of recommendations of market segment to target and to identify a plausible marketing strategy.

2 Executive summary

The disk storage systems market continues to grow year on year. A large section of the market is currently held by a small number of large vendors, however up to 23% of the market is held by smaller vendors [1]. The largest of the disk storage system providers have in the recent past seen profits grow rapidly year on year with IDC predicting a 50% growth in storage capacity demand annually until 2010 [2] with an estimated market value of around $5 billion per quarter, as at Q1 2006.

RAID 6 controller cards currently demand a premium as do hardware based RAID 5 controller cards.

AMD has recently achieved a 20% share (~1.2 million units for 2006) of the server processor market place worldwide.

3 Storage Market

3.1 Key players

The key players within the storage market are: HP, IBM, EMC, Sun Microsystems, Dell, Hitachi. The storage market place is heavily segmented into hardware and software solutions and technologies. Based upon storage disk systems and management software revenue figures for 2005 and 2006, an estimated value of $44 billion may be attached to this market [3]. EMC has estimated a figure of approximately $55 billion for the coming year [4].

3.2 Market segmentation

The world wide storage market place is segmented into:

1. **Software:** backup and archive, content management, storage management, network management, etc
2. **Hardware**: External storage, SAN, NAS
3. **Services**: Storage services, IT management services

Three types of consumer must be considered.

1. **Large corporate customers**: Utilising large scale high storage capacity systems with requirements for always on always available data solutions. Price of less concern than performance and matching requirements.
2. **SMEs**: Utilising large to small scale storage systems and requiring minimum IT skills and support services. More price sensitive though willing to pay for more expensive pre configured solutions.
3. **Home Users**: Typically requiring plug and play solution low cost and ease of use.

Consumers can be further segmented by their willingness to adopt technological advances. The early adopters whose outlook is 'I must have the latest thing; if it's new technology or the latest fashion I want it', the majority, who require encouragement to use the new technology; and the laggards, who come late to the technology once it is mature. The RAID 6 technologies being discussed are most likely to be of interest to large corporate customers and SMEs and early adopters in the consumer market.

3.3 Trends

The storage market continues to increase in sales volume and growth year on year. The largest of the disk storage system providers have in the recent past seen profits grow rapidly year on year with IDC predicting a 50% growth in storage capacity demand annually until 2010 [2] with an estimated market value of around $5 billion per quarter, as at Q1 2006.

4 Disk Storage Market

4.1 Key players

This study concentrates on the key players in the disk storage market[1] by operator, these are: HP, IBM, EMC, Sun Microsystems, Dell, Hitachi. The disk storage market place is estimated to be worth approximately $20 billion in 2006 based upon the most recent figures [4], [5].

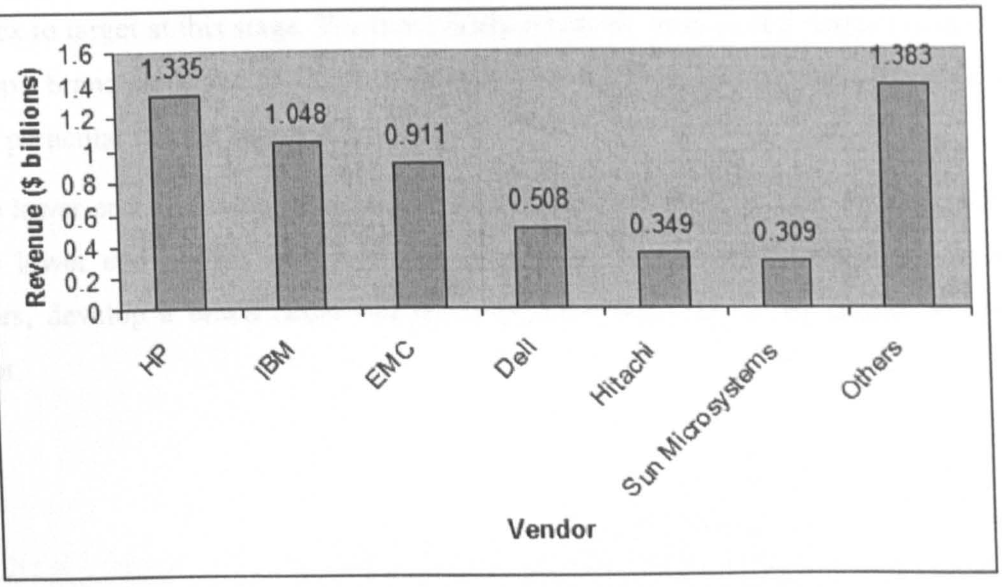


Figure 1. Worldwide revenue for in the disk storage systems market Q1 2006 [4], [5]

4.2 Market segmentation

Market segmentation may be made by characterising the devices as

1. **Enterprise level:** Devices of this class offer many terabytes of storage capacity and very high data throughput rates, with prices starting from around $40,000. i.e. the IBM DS6000 series offering support for 34TB of storage and costing $97,000

[1] Disk storage system is defined by IDC as a set of storage elements including controllers, cables, and host adaptors, associated with three or more disks. A system may be located inside or outside a server cabinet.

2. **Mid-range:** Devices of this class offer storage capacities from around 1-10TB with pricing ranging from \$5,000-\$40,000. i.e. the IBM DR550 series offering storage capacities from 1TB-7TB and costing ~\$40,000
3. **Low end:** Largest sub class ranging with a wide range of products from 0.1TB-3TB of storage space and costing from \$100-\$3,000.

Enterprise and Mid-range products will tend to be offered with support contracts and high levels of software and technical support. Also at this higher end of the market a complete storage solution will normally be sold, rather than the underlying technology. These market segments, whilst more tolerant of higher cost solutions, appear to be overly complex to target at this stage. The most likely routes to entry in this market would be to develop a brand name and products over time, or to sell on solutions to the current leaders in this particular market segment.

At the lower end of the market however there is greater scope for new entrants. Success in the lower end of this market this may be used to prove the technology to larger vendors, develop a brand name and promote the technology to the higher end of the market.

5 RAID Controller Solution

The RAID controller market is split into low price, low performance, PCI based RAID accelerator cards, and PCI-X/PCI-express based mid to high range products. The low end market is highly cost sensitive with prices starting as little as £10. This price point and market segment offers little return on investment and is currently unsuited to the hardware RAID 6 accelerator market.

Mid range to high end RAID controller cards again vary widely in price and performance. There are currently limited suppliers for hardware RAID 6 solutions that have been identified. These and the high performance RAID 5 controller cards are the market segments of greatest interest.

Focusing on the RAID 6 controller market there are three providers of RAID 6 solutions.

- Intel
- HP
- Adaptec

Of these three, only Intel provides RAID 6 utilising a Reed-Solomon coding scheme, offering support for array striping and the best performance when compared with competing solutions. Intel is promoting its XScale architecture (recently purchased by Marvell) with Promise, Ario, and Areca.

HP has had, for a number of years, a RAID 6 product based on a hardware and software combination to provide double disk redundancy on its high end systems. This appears not to be a true hardware RAID 6 solution.

Adaptec also has a RAID 6 solution which again is based on a hardware software combination to build and regenerate the array.

Purchase prices vary greatly for RAID controller cards as can be seen from the table below. The general trend is that the more disk ports and higher speed the interface board the higher the purchase price.

Table 1. Examples of current RAID controller card pricing from PC World Business 30th June 2006

Manufacturer	Device	Interface	RAID support	Disk connections	Cost
HP	Storage controller (RAID)	PCI	RAID 1	2-port SATA	£9.78
Adaptec	RAID 1210SA Storage controller	PCI	RAID 0, 1, JBOD	2-port SATA	£30.95
Adaptec	RAID 1420SA Storage controller	PCI-X	RAID 0, 1, 10, JBOD	4-port SATA II	£67.18
Promise	FastTrak SX4100 Storage controller	PCI 66 MHz	RAID 0, 1, 5, 10, JBOD	4-port SATA	£151.39
Intel	RAID Controller SRCS16 - Storage controller	PCI 64-bit	RAID 0, 1, 5, 10, 50	6-port SATA	£197.73
Adaptec	2820SA Storage controller	PCI-X	RAID 0, 1, 5, 10, 50	8-port SATA II	£320.74

Whilst Promise has reportedly released its initial RAID 6 controller cards with support from 8 to 16 hard disks, accurate pricing for these devices was not readily available. Prices quoted in press releases suggest a retail price of around $850 for the 16 port model [11]. HP's RAID 6 solution is integrated into their Proliant storage range, however from their website the 6-port SATA RAID controller may be purchased for $399 and the 8-port SATA RAID controller for $729.

6 Opteron Processor Solution

AMD has released its HyperTransport (HT) bus standard allowing interconnection directly to the Opteron processors via spare CPU sockets on motherboards. The AMD Opteron processor currently accounts for 22.1% of the x86 server processor market worldwide for Q1 2006 taking total sales over the $1billion dollar mark for the first time [8]. IDC estimates that global world wide server market will grow from 7.7 million units in 2006 to 9.6 million units in 2008 [10].Almost all of these Opteron processors will be utilised in systems with high data storage requirements which will require some form of RAID system be implemented. Using the current estimates for 2006 we could predict a potential market of 50% of Opteron sales which could benefit from hardware acceleration for RAID storage solutions. However, from this potential market it is more likely that between 1-5% would be willing to pay the premium to have the *storage algorithms* carried out on the HT bus. This would give an estimated annual volume of between 42,500-8,500 units.

To enable development for systems on the HyperTransport bus will require significant investment in new and unproven technology in a new and untested market. With Xilinx now providing a RAID 6 IP core it may only be a matter of time before a competing solution is released for this platform, potentially from the co-processor hardware manufacturers themselves.

7 Market Trends

The following trends have been noted during the market research.

The larger vendors for disk storage systems build their systems around third party hardware and software. There is a current trend to move towards Serial Attached SCSI and Serial ATA disks drives replacing the previous parallel standards.

The introduction of RAID 6 solutions for larger storage arrays containing more than 8 disks is increasing thanks to a number of new RAID 6 controller cards.

RAID controller cards increase in purchase cost in relation to the number of drives supported and the use of hardware control and acceleration. There is also a premium on 64-bit and PCI-X based controller cards.

RAID 6 controllers command a premium in price which rapidly increases dependant upon the number of disk drives supported by a given device. There appears to be a market for 4-port RAID 6 controller cards, though there is a clear trend aiming towards support 8 or more drives as the standard.

RAID controller cards are currently being manufactured for both PCI-X and PCI-express busses. Server class motherboards are currently being manufactured with both PCI-X and PCI-express bus support. There are relatively few server class motherboards which do not have PCI-X slots available. Therefore there would appear to be no immediate advantage to developing for the PCI-express bus.

8 Marketing recommendations

8.1 Strategic positioning

A decision has to be made as to the markets which A2E desires to target.

The strategic options available include:

- Manufacture and sell A2E branded RAID 6 controller cards
- Sell RAID 6 controller hardware to third party vendors
- Sell RAID 6 IP
- Implement HyperTransport based RAID 6 solutions

With Xilinx providing a RAID 6 solution for its FPGAs it has to be assumed that there is a market for RAID 6 IP blocks. Selling directly to Xilinx customers may now prove to be difficult as the have the option of using Xilinx's own IP and this brand name will most likely carry more weight than that of A2E. However, Altera as yet has no comparable product. A2E currently has a RAID 6 IP block which has been demonstrated to operate successfully in hardware along with supporting device drivers and test platform. This may be enough to convince Altera to market the IP block on their own website and enable them to compete with Xilinx.

Based upon the retail value attached to the mid to high end RAID controller market this would appear to be an obvious point of entry to this market. There are a number of routes that would be possible to take to enter the market.

- Manufacture and sell A2E branded RAID 6 controller cards
- Sell RAID 6 controller hardware to third party vendors

The production of A2E branded hardware would be necessary in the short term as product demonstrators and to support future development. Once developed the design could be sold on or licensed to third parties to re-brand. More complete performance metrics will be necessary to convince third parties of the performance gains of the A2E design over the Intel approach.

The market to develop co-processor/hardware acceleration for AMD Opteron servers offers a new niche market willing to pay for added performance. The potential of this market has yet to be realised and A2E has the opportunity to be a leader in this area. However, a number of caveats apply here. A2E would be reliant upon DRC for the provision of suitable hard solutions and to some extent access to their customer base. Also with Xilinx releasing a RAID 6 IP core for their devices it has to be assumed that there is a risk of DRC or another company utilising the Xilinx solution over the HyperTransport bus.

The recommended position for the short term is to approach Altera with regards to marketing RAID 6 IP and continuing development of a RAID 6 controller card whilst targeting potential customers interested in the technology.

8.2 Product

The product requirements for selling a RAID 6 IP block include:

- RAID 6 IP block
- Simulation file
- Datasheets
- Software drivers
- Test platform
- Customer support

Currently A2E have met all of these requirements, however it would be recommended to manufacture and test custom FPGA development boards prior to release to customers. Documentation and supporting materials would also require further review prior to customer release.

The product requirements for a RAID 6 hardware controller based upon equivalent products would be:

- PCI-X interface
- Minimum of 6 hard disk drives (4 data, 2 checksum)
- Support for RAID 0,1, 5, 6, JBOD and various combinations
- Windows Device Drivers
- Linux Device Drivers

A2E is currently working on completing all of the above requirements for a RAID controller card.

8.3 Price

The following general principles can be recommended for a RAID controller card:

- The market is willing to pay a premium for RAID 6 technology at the moment.
- Retail prices for a RAID 6 controller in the coming year should be approximately:
    - £150- £200 – for 4-port SATA RAID 6 controller
    - £200-£400 for a 8 port SATA RAID 6 controller
    - £400+ for a 16 port SATA RAID 6 controller

9 Conclusions

There is an increased interest in RAID 6 for storage solutions as can be seen by the number of new products currently or soon to be released supporting RAID 6 and the recent release of a RAID 6 IP core by Xilinx. The market for such technologies looks set to increase rapidly in the coming year as storage requirements are expected to increase for the foreseeable future and RAID based solutions continue to offer the best performance/cost results. With an increase in the use of lower cost and lower reliability SATA drives RAID 6 will allow storage array capacities to increase to meet the rising demand. Marketing of the various product offerings should begin immediately to locate interested parties and to enable early realisation of market potential.

10 References

[1] P Sayer, "IDC: Storage market revenue grows; HP's share shrinks", Available URL: http://www.computerworld.com/hardwaretopics/storage/story/0,10801,95682,00.html Last accessed 29th June 2006

[2] D Reinsel, et al., *Worldwide Disk Storage Systems 2006-2010 Forecast and Analysis: Expansion, Efficiency, and Economic growth*, May 2006, IDC Document Number 201596

[3] D Goulden, "*The Next Wave of Innovation: Broadening the Impact of ILM*", 2006, EMC Corporation, Available URL: http://library.corporate-ir.net/library/10/106/106202/items/182568/Goulden.pdf

[4] Online News, published by DMReview.com, June 9th 2006 URL available: http://www.drmreview.com/article_sub.cfm?articleId=1057332
Last accessed: 29th June 2006

[5] C. Martens, http://www.infoworld.com/article/05/09/02/HNdiskstoragemarket_1.html?DISK-BASED%20BACNKUP%20APPLIANCES
Last Accessed: 29th June 2006

[6] SNS Europe, 19th June 2006, http://www.snseurope.com/snslink/news/generalnews-full.php?newid=3883
Last accessed 29th June 2006

[7] IDC, 2nd June 2006, URL Available: www.datastorex.com/content/archived-new.asp?smonth=6&year=2006&recid=7959&type=3
Last Accessed: 9th June 2006

[8] T Shifrin, ComputerWeekly.com, URL Available:
http://www.computerweekly.com/Articles/2006/04/26/215660/AMD+gains+gound+on+Intel+in+server+chip+market.htm
Last Accessed: 29th June 2006

[9] Xilinx Corps, "*Hardware accelerator for RAID6 Parity generation / data recovery controller*", Application Note: Virtex-4 Family, April 19th 2006.

[10] IDC, "Worldwide and U.S. Server 2006-2010 Forecast", IDC, April 2005.

[11] ByteEnable, Promise introduces a new 16-port SATA RAID 6 controller, 7th March 2006, URL Available:
http://www.linuxelectrons.com/article.php/20060307111529286
Last Accessed 30th June 2006