Dual Channel Architecture for Reliable FPGA High Speed Serial Links

Kevin Ellsworth, Travis Haroldsen, Brent Nelson, and Michael Wirthlin
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering
Brigham Young University, Provo, UT. 84602
kellsworth1010@gmail.com, trharoldsen@yahoo.com, nelson@ee.byu.edu, wirthlin@ee.byu.edu

Abstract— Point-to-point connectivity through FPGA high-speed serial I/O is an important component for many space-based applications. The susceptibility of high-speed transceivers to soft errors is still under investigation and is a matter of concern in reliable point-to-point communications. This work develops a technique to provide full-bandwidth, lossless data transmission across high-speed transceivers in the presence of soft errors. The technique utilizes a redundant channel which can be used if the first channel fails and needs to be repaired. This allows for uninterrupted transmission even in the face of soft errors. The design has been implemented and tested through manual injection of various faults. No system failures occurred under the injection of any combination of tested faults on a single channel thus demonstrating the potential of such a design to protect against soft errors while maintaining high-bandwidth transmission. The cost of this technique is that it consumes $4.7\times$ the area and has a throughput that is 98% that of an unmitigated single-channel design.

TABLE OF CONTENTS

1 INTRODUCTION 1
2 MOTIVATION 2
3 PRELIMINARY WORK 2
4 ARCHITECTURE 2
5 IMPLEMENTATION AND TESTING 5
6 FUTURE WORK 6
7 CONCLUSION 7
ACKNOWLEDGEMENTS 7
REFERENCES 7
BIOGRAPHY 7

This work was supported in part by the I/UCRC Program of the National Science Foundation within the NSF Center for High-Performance Reconfigurable Computing (CHREC), Grant No. 0801876.
978-1-4244-7351-9/11/$26.00 ©2011 IEEE

1. INTRODUCTION

Many space-based applications require high-bandwidth point-to-point connectivity such as for the collection and processing of large amounts of sensor data or for communication between chips. Field programmable gate arrays (FPGAs) provide an effective platform for space-based applications due to their flexibility, reprogrammability, and low development cost. The increased availability of Multi-Gigabit Transceivers (MGTs) on FPGAs are providing the high speed communication links necessary to meet the demands of many of these high-bandwidth applications.

Concerns arise, however, over the susceptibility of MGTs to single event upsets (SEUs) in a space environment, especially for applications intolerant to data loss. Preliminary research on MGT failure modes indicates most failures are completely recoverable with proper stimulus, but the time necessary to recover may be on the order of microseconds [1]. With data transmission speeds reaching 28 Gbps over a single channel, a soft error which causes a loss of link lasting only microseconds can result in several kilobytes of data loss.

To date, very little work has been done on developing reliable architectures for high speed serial at the data link layer. Many protocols exist that are designed to provide reliable transmission in the face of signal noise but few are designed to handle any failure of the actual transmit or receive mechanism. Those protocols that could conceivably be used for such reliability are generally expensive in terms of resource utilization and spatial and temporal overhead. Furthermore, these protocols are overly complex and inappropriate for point-to-point applications. The goal of this work is to investigate soft error mitigation techniques at the data link layer which provide full-bandwidth, lossless data transmission in the presence of SEUs.

2. MOTIVATION

Failure mechanisms in MGTs caused by SEUs is an area of ongoing research by various groups such as the Xilinx XRTC. There are two main areas of concerns from possible effects of SEU failures - 1) corruption of transmitted data (bit errors), and 2) failure of the transmit or receive mechanisms (resulting in a loss of link). The first area of concern can be modeled as signal noise and conventional protocol techniques such as CRCs can be utilized to detect errors [2]. The failures resulting in loss of link are thus of primary concern in the development of mitigation techniques.

Research described in [1] on the failure mechanisms of MGTs suggests that loss of link failures due to high-energy radiation can almost always be recovered by applying appropriate local stimulus such as a reset of the transceiver or one of its components. The amount of time required to recover, however, can be of great concern to systems that require continuous high-bandwidth transmission. Such systems may not be able to tolerate a communication link that is down for even microseconds. Thus it is necessary to provide a mechanism that will allow for continuous transmission in spite of MGT failures.

One solution to provide reliable uninterrupted data transmission is to provide a redundant channel. The redundant channel allows for data to be transmitted across the working channel while the failed channel is recovering from a failure. This work describes an implementation of such a dual-channel architecture and details its architecture, design, and fault testing.

3. PRELIMINARY WORK

A significant concern when creating a dual-channel architecture is properly aligning the received data so that both channels will be synchronized. Prior to the development of the dual-channel architecture described in this paper, we conducted an experiment to determine the maximum skew between data transmitted across independent channels. We discovered that under normal conditions the skew between channels was very limited, especially if they shared the same reference clock, with data on the two channels arriving only a couple of clock cycles apart from each other if there was any skew at all.

There are two options which could be used for clocking a dual-channel design: (1) a shared reference clock for both channels or (2) two independent reference clocks, one for each channel. A shared reference clock is a reasonable architectural decision because each channel has its own PLL to generate its own clock from the reference clock. Thus, the reference clock routing going to the PLL in each channel forms a very small target for SEUs compared to any other component in the system and limits the probability of a common mode failure. Furthermore, although the logic for each channel provides for a unique reference clock, the high I/O costs and difficulty of generating low jitter differential clocks may deem such an option not worth the cost. Our initial testing used such a shared reference clock. In our final testing of the design, however, we routed the single reference clock into the FPGA over two different paths. Thus, each channel received the same frequency clock but with different phase. Both designs performed equally well. Understanding the limited skew between independent channels gave us confidence in being able to design a system which could properly align received data from dual channels with a minimal amount of buffering.

4. ARCHITECTURE

A block diagram of our dual-channel architecture is shown in Figure 1. As each half of the figure is a transmit/receive pair, we have highlighted in black a transmit block on the left and a receive block on the right to show data transmission from left to right. The implementation is built on top of the Xilinx Aurora protocol [3], although the concept is not dependent on a specific protocol.

In a Xilinx FPGA, each high speed serial I/O tile contains two MGT cells. To avoid a single point of failure that could bring down both channels at the same time, this design places the two redundant channels into separate tiles.

Aurora Protocol

The Aurora protocol is an open IP core which is designed to provide a minimal amount of logic and protocol overhead to interface with the physical layer of the MGT serial links on Xilinx FPGAs. It provides a mechanism for the framing of data across a serial link, 8B/10B encoding, and basic error checking for encoding errors, physical layer errors, and framing errors. It does not however, provide any frame tracking or data verification beyond 8B/10B encoding. Aurora accepts data packets from the user circuitry, surrounds each data packet with start-of-frame and end-of-frame characters, and transmits them.

We utilized Aurora to provide the basic interface to the serial links and built added reliability circuitry around it. Our design (called RAPS for Reliable Architecture for

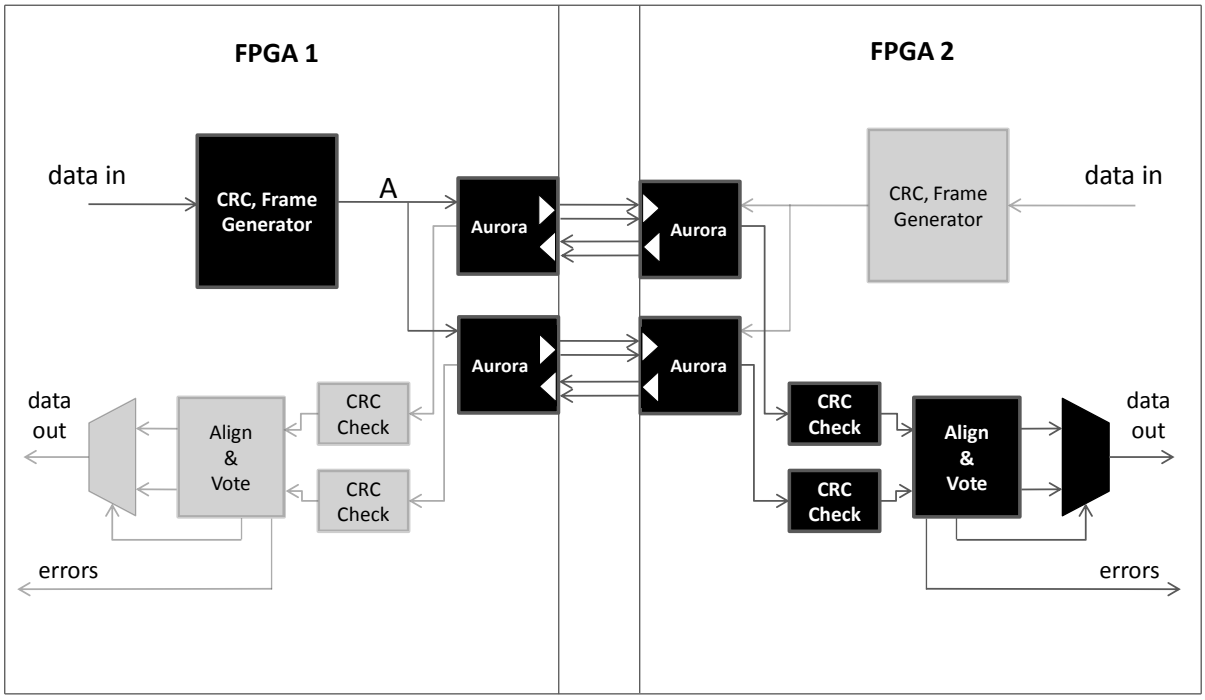

Figure 1. Block diagram of RAPS.

Point-to-Point Serial I/O) uses the Aurora Protocol implementation version 5.1 generated using Xilinx Coregen. Our logic presents the same interface to the rest of the FPGA design that the original Aurora core presented. Thus, our logic can easily be inserted into an existing design.

Transmission

Figure 2. RAPS packet structure

In our design the transmit block receives data from the user. A sequential packet number is prepended to the data. Additionally, a 32 bit standard ethernet CRC is generated from the user data and appended. The complete packet structure is illustrated in Figure 2. The new packet is then passed to the two independent Aurora cores for transmission across both channels. For verification, a data corruption block was inserted at location A in Figure 1. More details on this block are provided in the testing section.

Reception

On the receive side, the packets are checked for any errors that may have occurred during transmission as reported by error signals from the Aurora protocol and CRC check failures. The receiver then aligns the data according to the packet number and a voting mechanism tests the twin data packets for errors and selects between the two if an error is found in either packet. A further description of the blocks is given below.

CRC Check—The trailing CRC on the packet is checked immediately following the Aurora core. The CRC checker is a Xilinx Coregen block which verifies the data against the CRC 32 bits at a time and strips the CRC from the data stream.

Align and Vote—The alignment of the packets proved to be a complicated procedure. The packets must be aligned to compare the received data and appropriately pass it to the end user. Prior testing revealed that skew between channels is minimal, usually only consisting of a few bytes. This means that extensive buffering of data is not required.

To prepare the data to be aligned, the data is buffered

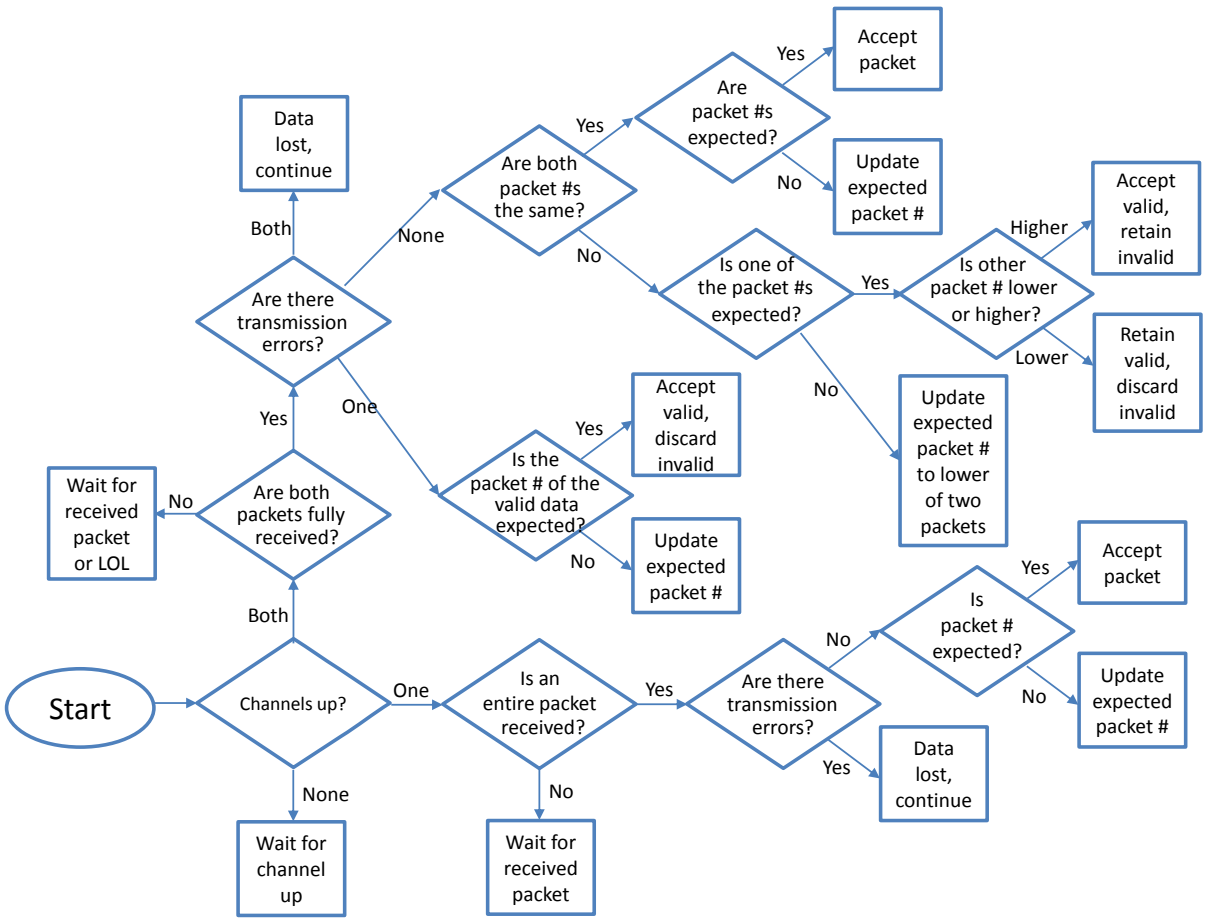

Figure 3. Flow Diagram of the Receivers Alignment Procedure

in a FIFO sufficiently large to hold at least 2 packets. All information about the packet is also stored in a separate FIFO. This information includes the packet number, packet length, and errors (from the Aurora core and the CRC checker) relating to the packet. Once this information is available on both lanes, the two transmitted packets are ready to be aligned.

The alignment and voting decision procedure is diagrammed in Figure 3. In the diagram, a decision tree is traversed until a decision is made for the current state of the receiver. Decisions include delaying until more information is present, accepting and passing the data to the end user, discarding and retaining packets for the iteration for alignment purposes, matching the expected packet number to what actually appears from the transmitter, and possibly acknowledging that packet data is unrecoverable and continuing on.

The alignment circuitry stores the packet until either both lanes have fully received a packet or one of the Aurora cores reports a loss-of-link event. A steady stream of data is necessary as a lost packet can only be detected by the reception of the following packet. Once either of these states occurs, the system attempts to align the data based off of the packet number, and then passes the packet containing correctly transmitted data along to the user.

Under normal operation packets are received on both channels, and the packets are checked for transmission errors. Both packets should contain a packet number one greater than the previously received packet. Either of these packets is then certified to be valid and passed along to the user. If one or more bit errors occurs on only one of the channels, the packet from the other channel is accepted. If one of the channels appears to have gotten ahead or behind the other, mechanism are in place to dis-

card lagging packets while maintaining current packets or retaining leading packets while passing along current packets to realign the data.

To allow for the continuous transmission of data in the event of a loss of link on a single channel, the receiver will operate using the single active channel until the faulty channel is repaired. This allows for the broken link to be repaired without suffering a loss in throughput.

Finally, mechanisms are in place to update the expected packet number should the transmitter and receiver break in packet numbering (usually resulting from a transmitter reset). The voter is also able to acknowledge that a packet's data is unrecoverable in some events and continue operation while alerting the user to the lost data.

Upset Recovery

The architecture is designed to provide as much information as possible on the types of failures which have occurred through various error signals. These error signals include information on each channel such as connection status, framing errors, packet alignment, loss of packet, etc. This information allows an external repair mechanism to determine a probable cause of the failure and to perform the lowest cost repair necessary to correct the system. Such recovery techniques of differing costs are known and discussed in [1].

Performance

The architecture allows for minimal overhead. The added data transmission latency is small and is relative to the size of the transmitted packet as the packet must be buffered to allow the accompanying CRC to be verified. The additional overhead for each frame is only 4 bytes, allowing an efficiency of 98% of the throughput of an unmitigated Aurora core at a packet size of 256 bytes.

5. Implementation and Testing

To verify the architecture's reliability, the design of Figure 1 was implemented onto a Xilinx Virtex5 LX110T FPGA. The design was completed in roughly two months by a team of two graduate students. The implementation focused on the testing of the architecture and did not include sophisticated repair mechanisms for the MGTs.

The resource utilization for the design, as well as the utilization for a single channel design using the Aurora protocol is given in Table 1. The dual channel design utilizes two Aurora cores as well as the additional reliable protocol logic. The cost of the additional protocol logic makes the reliable design more than four times that of an unmitigated Aurora protocol design but is still less than a TMR design. A TMR design would require even more logic for aligning the received data and would utilize three rather than two times the logic for the Aurora cores.

Approximate Resource Utilization				
Component	Slices	BRAM	CRC32	MGTs
Dual Channel	1,389	12	3	2
Aurora	296	0	0	1
Frame Gen	189	4	1	0
CRC Check	71	0	1	0
Alignment	672	7	0	0
Available on FPGA (LX110T)	17,280	148	32	8

Table 1. FPGA resource utilization for dual channel design.

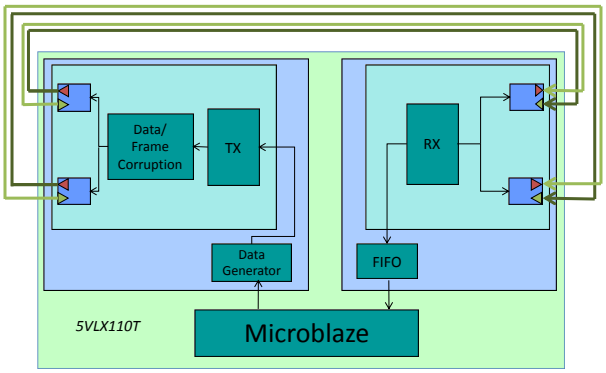

Figure 4. Testing and verification setup.

The test setup on the FPGA is detailed in Figure 4. It was created by generating two instances of the design on a single FPGA. The MGTs of each instance were connected via SMA cables. To better represent a multi-FPGA environment, each MGT in the design was run from a different phase reference clock. The reference clock rate was 156.25 Mhz which resulted in a transmission rate of 3.125 Gbs across the MGTs. Attached to one instance was a data generator which created randomly sized packets separated by random wait intervals. The receiving instance logged errors on the redundant channels of received data.

The architecture was verified by injecting a variety of errors. The faults injected consisted of the following: (1) errors in the data stream, (2) errors in the Aurora frame headers and footers, and (3) catastrophic failures

(eg. loss of link). Error types (1) and (2) were injected at location A in Figure 1 with a VHDL data corruption block. These errors were meant to simulate the different types of errors that are typical of MGTs.

Data Stream Errors

When an error on the data stream was requested, the block would pseudo-randomly flip a bit or series of bits in the data payload entering the Aurora core. This is approximately equivalent to a bit error. This type of error was easily detected by the CRC blocks on the receive channel.

Testing with insertion of more than 200 million data stream errors in a single test run resulted in no failures in the system. Additionally, data stream errors never required a reset of the channel. Errors were inserted in both channels, but only in a single channel at any given time.

Framing Errors

When a framing error was requested, the corruption block would corrupt the signal which tells the Aurora core to create an end-of-frame signal for the packet. This would thus remove or insert an erroneous end-of-frame character in the transmitted data stream. These errors were potentially more hazardous to the system because they caused alignment mismatches on the receive side. Because received packet lengths were different sizes, or frame signatures were missing entirely, these faults were designed to stress the alignment and frame tracking of the receive logic.

Similar to data stream errors, testing included the injection of more than 200 million framing errors in a single test run with no system failures. The insertion of framing errors had the potential to affect the system more than data stream errors due to the effect of improperly filling data FIFOs in the align logic. However, successive framing errors were inserted to stress the alignment logic and never resulted in failure. Again, framing errors were inserted on both channels but not simultaneously.

Catastrophic Failures

Catastrophic failures were simulated by resetting the MGT or physically removing the cables connecting the transmit/receive pair. These faults were designed to test the complete failure of the transmit or receive mechanism. Any reset of the core or loss of link from removing a cable completely disrupts the connection between MGTs and forces the channel connection to reinitialize after the fault is removed. We note that such catastrophic failures due to radiation events are rare as described in [1].

Following bit errors or framing errors, the faulty channel was fully operational for the next packet of data. Following a catastrophic failure, the faulty channel was unusable until it was reinitialized by the Aurora core. Such a reinitialization of the channel required time on the order of a few microseconds. During this time, however, the other redundant channel was able to continue transmitting data. After reinitialization the previously faulty channel was again fully operational.

The nature of inserting catastrophic failures and the recovery time necessary made it more difficult for a test run to contain as many fault insertions. However, testing with the insertion of more than 100 catastrophic failures resulted in no system errors.

The design also experienced no failures with any combination of multiple types of errors injected into a single channel. Additionally, the affected channel always recovered after the injected fault was removed. Obviously, when errors were injected on both channels data transmission was interrupted and data lost. However, a reset of the system then always caused the system to recover and data transmission was able to resume.

6. FUTURE WORK

In order to further verify the operation of the dual-channel architecture fault injection of the FPGA configuration bitstream should be performed. Additionally, we are interested in testing the design in a high ion beam. We are interested to see the effects of SEUs on the Aurora core, particularly to see if the design is capable of mitigating errors to the Aurora core logic. To perform beam testing, a recovery mechanism for the architecture will need to be implemented. An area for future investigation is to utilize the error signals from the architecture to provide an efficient, high-level recovery mechanism. Also, The current implementation of the architecture focuses on mitigation of MGT failures. Further work needs to be performed to investigate protection against failures for the surrounding protocol logic and memories.

Additionally, while the architecture requires fewer resources than a full TMR implementation, the use of two transceivers for a single lane of data is still too costly for many applications. Future work may include a multi-channel design with only one redundant channel. Such a system could be modeled after a RAID system and would provide for higher throughput with less overhead.

7. CONCLUSION

The dual-channel architecture for serial IO presented in this work provides a data link layer mitigation technique for reliable MGT data transmission in the face of SEUs. A main focus of this work has been to provide a highly available system in order to have continuous high-bandwidth transmission despite faults. The dual-channel design provides for reliability and availability with less overhead than a TMR system. Our implementation of the design demonstrated its effectiveness in protecting against SEU failures, and lays the foundation for the development of more efficient architectures in the future.

ACKNOWLEDGMENTS

We would like to acknowledge Los Alamos National Labratory and specifically Keith Morgan for the initial suggestion of the described dual channel design and for providing valuable feedback on our work.

REFERENCES

[1] R. Monreal, G. Swift, C. Khuc, C. Carmichael, C. Tseng, S. A. Anderson, M. Coe, and J. Price, "Investigation of the single event effects and subsequent recovery mechanism induced by multi giga-bit transceivers (mgt)," in *NSREC*, Apr 2010.

[2] K. Morgan, M. Caffrey, M. Dunham, P. Graham, H. Quinn, C. Carmichael, T. Duong, A. Lesea, G. Miller, G. Swift, C. W. Tseng, Y. Wu, R. Monreal, and G. Allen, "Upset-induced failure signatures, recovery methods, and mitigation techniques in a high-speed serial data link for apace applications," in *NSREC*, 2008.

[3] Xilinx Corporation, "Aurora 8b/10b protocol specification, sp002," Jun. 2009.

BIOGRAPHY

Kevin Ellsworth is a M.S.E.E. student at Brigham Young University with a research emphasis in reliability for FPGAs. He received his B.S. degree in Computer Engineering from Brigham Young University in 2009.

Travis Haroldsen is a M.S.E.E. student at Brigham Young University with a research emphasis in reliability for FPGAs. He received his B.S. degree in Computer Engineering from Brigham Young University in 2010.

Brent Nelson is a professor in the Department of Electrical and Computer Engineering at Brigham Young University and program head for the Computer Engineering program there. He received his PhD in computer science in 1984 from the University of Utah in the area of VLSI CAD. His current research interests focus on high-end computing applications of FPGAs and on CAD for the design of FPGA-based applications. He currently serves as co-director for the NSF Center for Reconfigurable High Performance Computing (known as CHREC) and as director of the BYU site within that center.

Michael Wirthlin is currently an Associate Professor in the Department of Electrical and Computer Engineering at Brigham Young University in Provo, Utah. He has been actively involved in FPGA design for over 20 years and is active in the FPGA design, architecture, and tool research communities. He is currently a leading researcher in FPGA reliability modeling and fault tolerant design techniques. He and his students have developed tools for automatically inserting fault tolerant structures into FPGA designs and have tested these techniques on the orbiting Cibola Flight Experiment (CFE) satellite launched by Los Alamos National Laboratory. His research interests include FPGA reliability modeling, FPGA fault tolerant design techniques, Configurable Computing Systems, high-level synthesis, and computer-aided design for application-specific computing.