Error Mitigation of Point-to-Point Communication for Fault-Tolerant Computing

Robert L. Akamine, Robert F. Hodson
NASA Langley Research Center
Hampton, VA 23681-2199
(757) 864-5005
robert.l.akamine@nasa.gov
robert.f.hodson@nasa.gov

Brock J. LaMeres
Montana State University
Bozeman, MT 59717
(406) 994-5987
lameres@ece.montana.edu

Robert E. Ray
Jacobs/ESTS
Marshall Space Flight Center
(256) 544-0604
robert.e.ray@nasa.gov

Abstract -- Fault tolerant systems require the ability to detect and recover from physical damage caused by the hardware's environment, faulty connectors, and system degradation over time. This ability applies to military, space, and industrial computing applications. The integrity of Point-to-Point (P2P) communication, between two microcontrollers for example, is an essential part of fault tolerant computing systems. In this paper, different methods of fault detection and recovery are presented and analyzed.

TABLE OF CONTENTS

1. INTRODUCTION... 1
2. FORWARD ERROR CORRECTION.......................... 2
3. TRIPLE MODULAR REDUNDANCY........................... 5
4. PARALLEL ARQ PARITY.................................... 5
5. ETHERNET.. 8
6. CONCLUSIONS.. 12
REFERENCES.. 12
BIOGRAPHY.. 12

1. INTRODUCTION

Detecting and recovering from physical damage or 'glitches' that occur in communication between two points is a critical capability for fault tolerance systems. In military applications, mitigation of physical damage of wires caused by combat operations is critical to give troops the confidence in their offensive and defensive systems. Manned space operations face similar dangers. Recovering from 'glitches' caused by radiation induced Single Event Upsets (SEU) and Single Event Transients (SET) on communication wires is essential for space systems. Radiation events are mainly caused by ionized radiation which deposit charge in memory devices such as flip-flops. This charge can trigger state changes. Considerable research, development, and application of Consumer-Off-The-Shelf (COTS) products, due to the lack of cost-effective high performance radiation hard parts, in fault tolerance computing systems has produced radiation tolerant processor platforms [1][2][3]. However, the integrity of data passing between two processing or storage systems needs analysis to ensure the fault tolerant ability of an entire system.

Generically speaking, two strategies of error detection and recovery exist; masking and non-masking. Masking techniques are defined as detecting, isolating, and correcting errors in real time. Forward Error Correction (FEC) codes, such as Hamming codes, are an example of a masking method. In non-masking techniques, error detection occurs in real time, but fault isolation and correction occur in separate and later processing. Automatic Repeat Query (ARQ) methods that use a command and response algorithm to verify proper data communication is a non-masking technique. Overall the different strategies of communication integrity can be illustrated in a hierarchical tree as shown in Figure 1.

Two masking techniques are studied further:

1. Using Hamming codes to detect, isolate, and correct for single bit errors in a parallel FEC example.

2. Implementing a Triple Modular Redundancy (TMR) system to detect, isolate, and correct for data errors in a parallel example

Two non-masking techniques are also studied:

1. ARQ implementation with simple even-parity to detect errors in a discrete example

2. ARQ implementation using Ethernet as the physical communication medium in an asynchronous serial technology example

Each example discusses the theory of operation, and analysis. Analysis presents the advantages and disadvantages of each technique as a function of data bandwidth, complexity, wire mass, and latency.

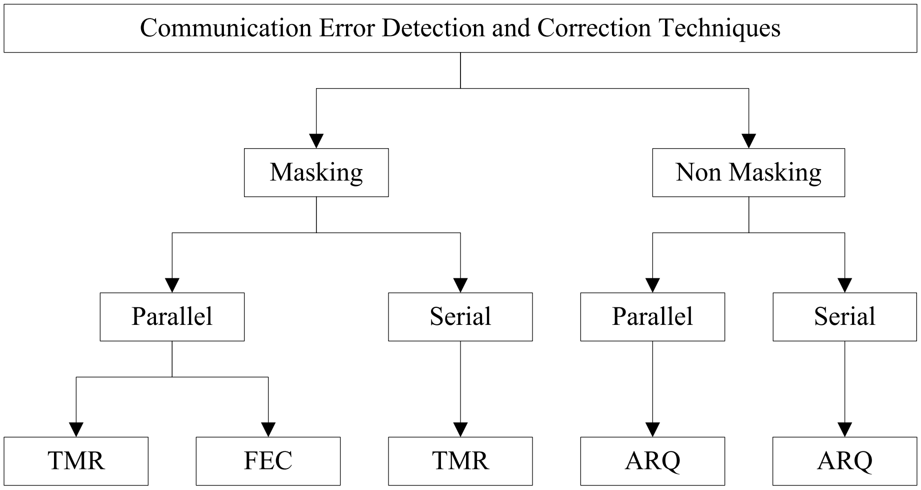


Figure 1 – Error Detection and Correction Techniques

2. Forward Error Correction Example

Research and development conducted by Montana State University successfully demonstrated a masking, parallel, FEC design. The demonstration used Hamming Codes, a block code suitable for a parallel data communication, due to its ability to detect single bit errors and maintain channel rate efficiency [4].

Richard Hamming introduced his FEC in 1950 and the most common use of his code is in memory and storage electronics. A Hamming code operates by overlapping n parity bits for $2^n - n - 1$ data bits. Each parity bit is the exclusive-or (XOR) operation of the data bits where the binary AND of the Hamming parity position and the bit position is non-zero [4]. For example, a (7,4) Hamming code, seven total bits with four data bits, has the data bit position and parity position table defined in Table 1.

Table 1. A (7,4) Hamming Code Table

(7,4) Hamming Code								
Bit Position		1	2	3	4	5	6	7
Parity/Data		p1	p2	d1	p4	d2	d3	d4
Parity Position	1	X		X		X		X
	2		X	X			X	X
	4				X	X	X	X

The overlapped parity property of Hamming codes allow for Single Error Correction (SEC). Double Error Detection (DED) can be achieved by adding an additional parity bit which is the XOR of the four data bits in a (7,4) Hamming code [4]. The resulting Hamming encoding circuit combining SEC and DED (SECDED) technologies is shown in Figure 2.

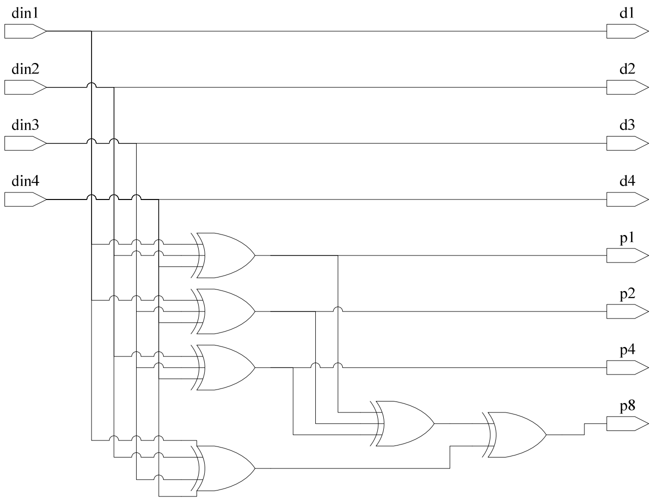


Figure 2 - (7,4) SECDED Hamming Encoder

Decoding Hamming codes is best described in a pictorial circle diagram similar to a union and intersection circle diagram [5]. For a (7,4) Hamming code the circle diagram can be defined as shown in Figure 3.

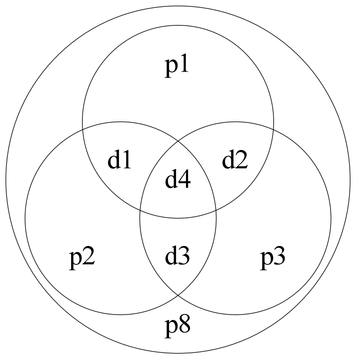


Figure 3 - (7,4) SECDED Hamming Diagram

A few decoding examples using the Hamming diagram follow; when there are no errors, one data bit error, one parity bit error, and more than one error.

If no errors occurred during transmission between two points then no parity flags will exist. If one data bit error at ‘d2’ occurred during transmission then two parity errors will at ‘p1’ and ‘p2’. Two parity flags occur if a single error occurs at ‘d1’ and ‘d3’. All three parity errors will occur if ‘d4’ is erroneous. If one parity error is received only that one parity bit will flag and can be ignored. In the case of multiple errors Hamming codes are not adequate and the overall parity check must be used to determine a double error condition [5].

In general, Hamming codes become more efficient when the number of data wires is increased in one major way; channel efficiency η. As the number of data bits for encoding increases the ratio of parity bits decreases as shown in Figure 4 given the equation [5]

$$\eta = \frac{\#\, of\ Data\ Bits}{\#\, of\ Total\ Wires}$$

The Montana State University technology demonstration implemented a (31, 26) Hamming code on a Virtex-5 evaluation board to provide the SECDED ability. The combined Hamming and final parity bit provides a 32 bit parallel bus, or block, which encodes 26 data bits [4].

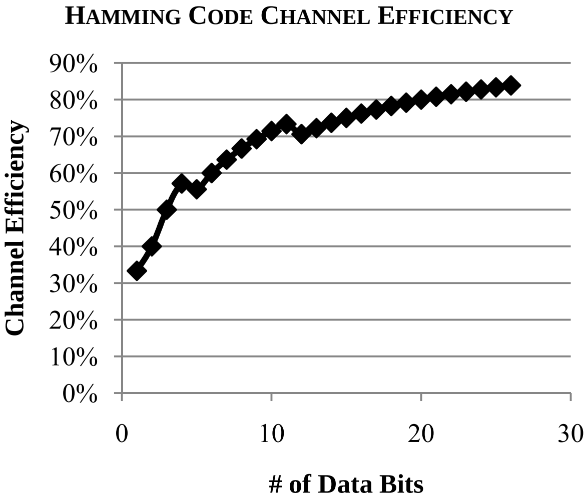


Figure 4 - Hamming Code Channel Efficiency

The demonstration was intended to emulate communication between instruction ROM and a Xilinx Pico-Blaze core. An 8 and 18 bit counter provides the address and instruction respectively, and the system diagram is shown in Figure 5. It is the intention that the Hamming encoding, FPGA routing fabric and Hamming decoding remain transparent to the ROM and Pico-Blaze. Additional logic was added to switch from known bad wires to spare wires. The state flow diagrams of the ROM and microcontroller designs, in Figure 6 and 7 respectively, display the basic switching method to avoid damaged wires [4].

The implementation results of the demonstration proved the (31,26) Hamming code was able to detect and recover from experimentally induced hard-faults. With the addition of spare wires and switching logic the design was able to reroute to spare wire and continue operation without channel performance degradation [4].

The Hamming code system, being a parallel bus, has numerous disadvantages. Although a larger Hamming code has higher channel efficiency, the total bandwidth of any parallel bus is limited due to inter-symbol interference, noise, and transmission line properties. This parallel system does not have the ability to communicate over longer distances. Compared to serialized technology, the parallel Hamming system will have less bandwidth. Wire mass is high in this system, which may make it unsuitable for avionics and space applications.

There are some advantages of the Hamming code system, without the added logic to switch to spare wires when damage is detected, is a simple way to detect and correct single bit errors and operate with one damaged wire. With a moderate amount of complexity additional spare wire switching can be added, as demonstrated. Hamming codes by their virtue are easy to encode and decode giving a low latency between two communication points.

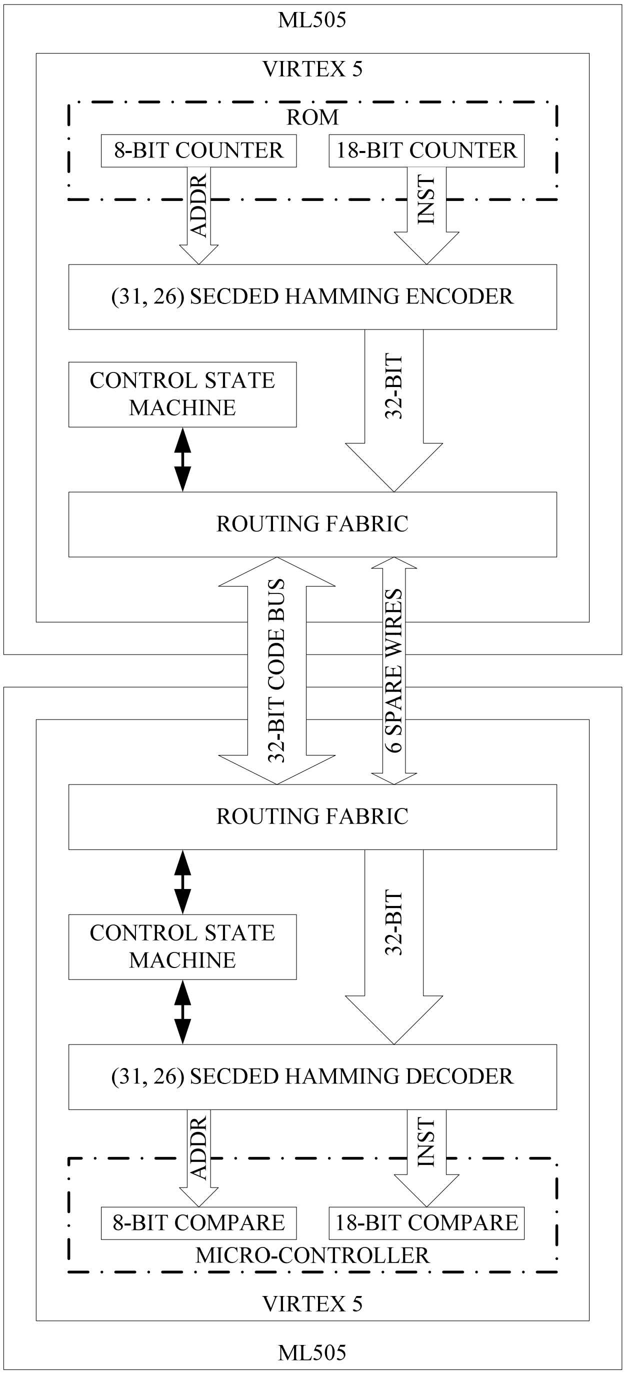


Figure 5 – Hamming Reconfiguration System Diagram

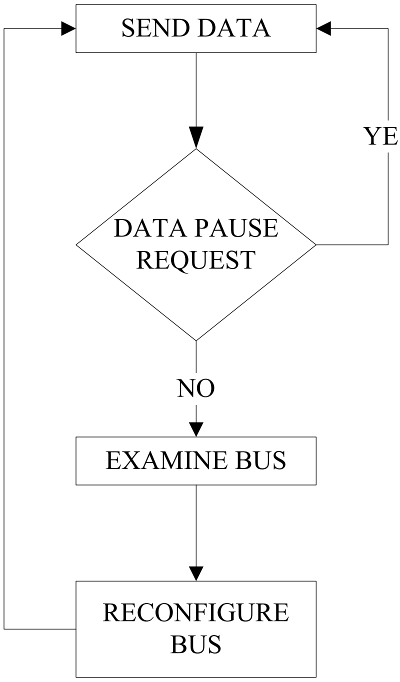


Figure 6 – ROM Hamming Reconfiguration Switching State Diagram

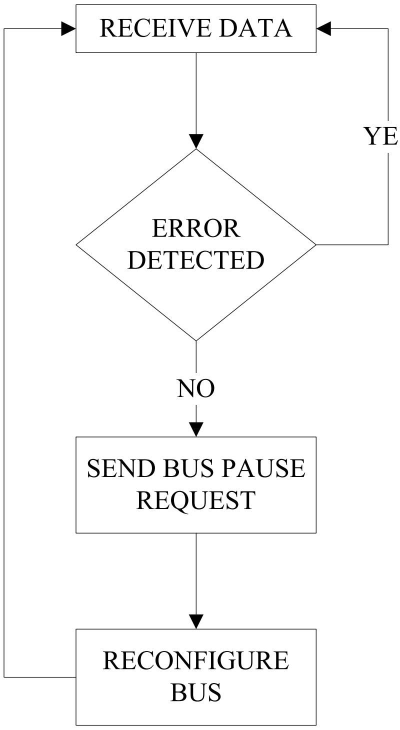


Figure 7 – Microcontroller Hamming Reconfiguration Switching State Diagram

3. Triple Modular Redundancy

TMR is a simple masking technique. In a parallel bus system the data is replicated, or split, three times at the data source. The sink, or receiver, votes on the three channels bit by bit and determines the correct data by majority rule. In theory, up to one wire can be damaged per bit in TMR and still maintain data integrity. The channel efficiency of TMR is low when compared to Hamming codes and is a constant

$$\eta = \frac{1}{3}$$

The demonstration system involved two rapid prototyping FPGA boards. One board serves as the constant data source or ROM for example. The other serves as the data receiver or micro-controller. A one byte up-count pattern is copied between three separate one-byte parallel buses in the ROM prototyping board and is connected to the receiving end. The data in the receiver, or micro-controller, determines errors by voting on the three inputs bit by bit. An example of a one-bit TMR voter circuit is shown in Figure 8. The resulting data is compared to the expected up-count pattern to verify proper operation. The complete TMR system diagram is shown in Figure 9.

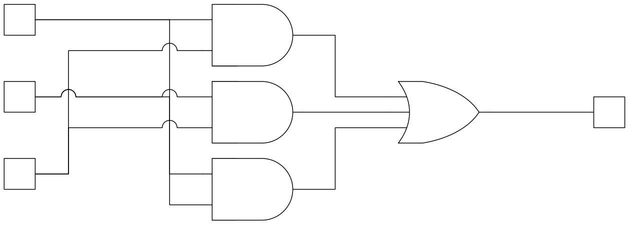


Figure 8 – One-bit TMR Voter Circuit

The TMR demonstration was successfully able to detect and recover from induced bit errors within the scope of TMR. There are major disadvantages to TMR. As with the Hamming code example, a parallel bus system has a relatively slow data rate. The low channel efficiency compounds the problem by adding in three times more wire mass per bit. TMR does provide a very simple way to guarantee data communication between two points. Given its low complexity, TMR provides a low latency technology.

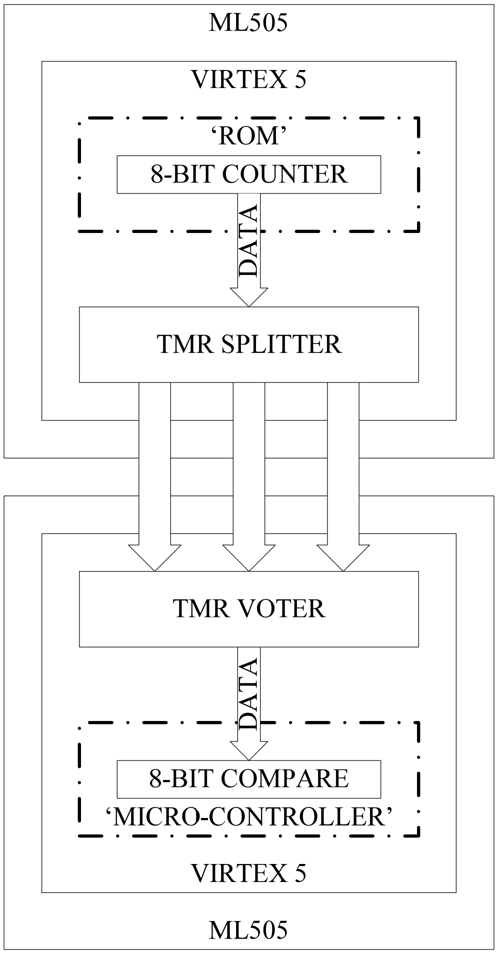


Figure 9 – TMR System Diagram

4. ARQ Parallel Parity Example

The last parallel bus method of detecting and correcting errors in communication between two points is using parity and ARQ. In a simple ARQ system, the transmitter sends data to the receiver and then waits for an acknowledge flag before sending new data. If an acknowledge flag is not received in a certain amount of time the transmitter resends the data. The receiver can determine valid data by any error detecting code or redundancy check. A simple even or odd parity can detect errors under certain conditions. If there are an odd number of bit errors in transmission then parity can detect the error, an even number of errors cannot be detected. Despite this limitation, parity is an extremely simple method of error detection that has little effect on the communication channel efficiency.

This simple ARQ does not guard against a certain condition. If data wires are damaged then the receiver has high probability accepting data incorrectly based off the characteristics of parity. In addition, if the acknowledge line is damaged then the transmitter will assume the receiver did not accept the data and continue to resend. To guard against these conditions, a method to discover damaged wires and reroute to spares needs investigation.

To overcome the fore mentioned error conditions changes must be made to how the transmitter responds to acknowledgement timeouts. If an acknowledgement is not received then the same data should be retransmitted at least once to determine if a SET occurred. A sequence bit should be reserved, or added, to the bus to inform the receiver the data repeat was intended and not associated with a missed acknowledgment. New data changes the sequence and a retransmit does not. If another timeout is encountered then the transmitter must assume wire damage has occurred somewhere on the bus. A logic high is asserted on the bus, all bits including spare wires, followed by a logic low by the transmitter. This allows the receiver to determine which wires are damaged and reroutes to the next available spare. The receiver in turn sends back a logic high and logic low to allow the transmitter to identify the damaged wire and switch to a spare. Once the wires have been rerouted then normal operation resumes. The transmitter follows the state flow visualized in Figure 10 and the receiver in Figure 11.

The channel efficiency of a parity driven ARQ with an additional sequence bit increases as the number of data bits increases (as shown in Figure 12), excluding spare wire from the calculation. With spare wires included in the calculation the channel efficiency is defined as

$$\eta = \frac{\#\,data\ bits}{\#data\ bits + 3 + \#spare\ bits}$$

It should be noted since the transmitter must wait for an acknowledgement from the receiver the maximum possible data rate is half the total bandwidth of the bus. Practically speaking the latency of receiving end will not be instantaneous; therefore the data rate will be much less than half the bandwidth.

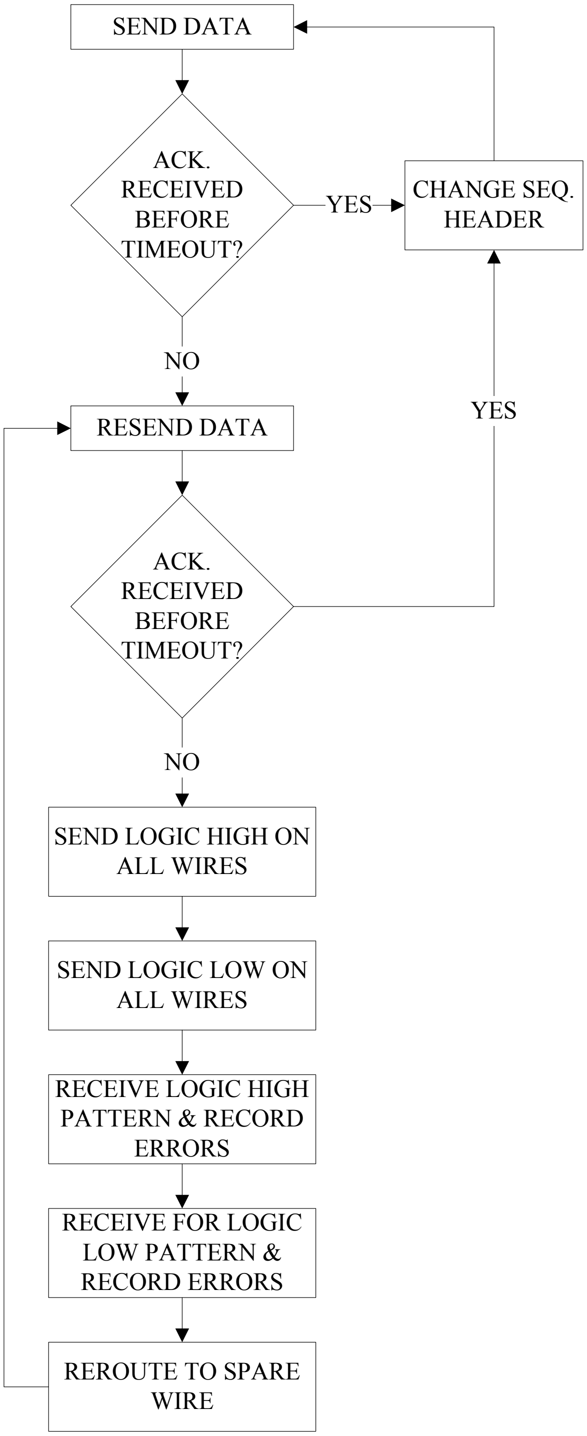
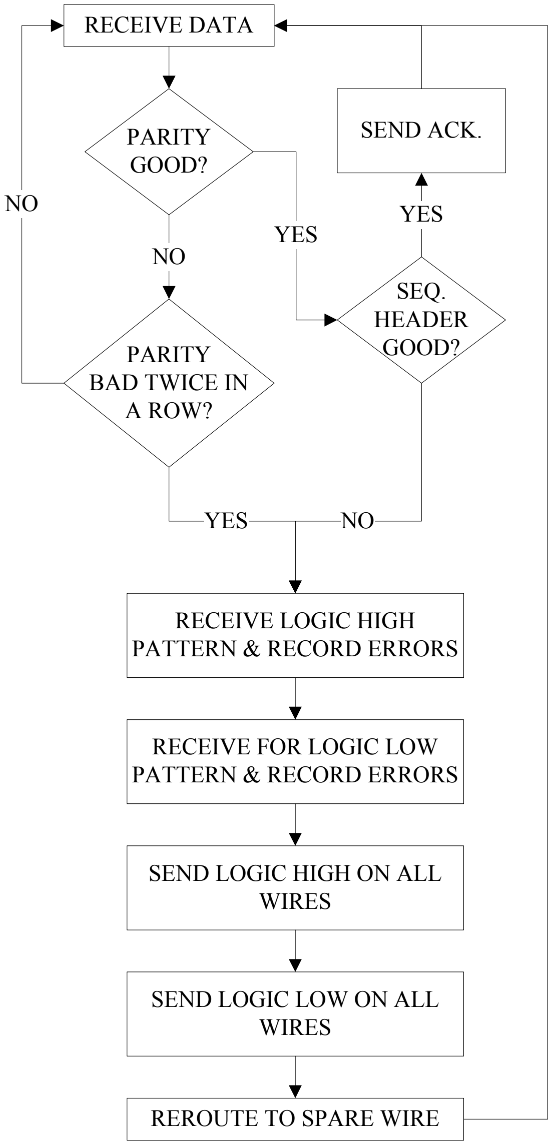

SEND DATA
ACK. RECEIVED BEFORE TIMEOUT?
YES
CHANGE SEQ. HEADER
NO
RESEND DATA
YES
ACK. RECEIVED BEFORE TIMEOUT?
NO
SEND LOGIC HIGH ON ALL WIRES
SEND LOGIC LOW ON ALL WIRES
RECEIVE LOGIC HIGH PATTERN & RECORD ERRORS
RECEIVE FOR LOGIC LOW PATTERN & RECORD ERRORS
REROUTE TO SPARE WIRE

Figure 10 - ARQ Parity Transmitter State Flow



RECEIVE DATA
PARITY GOOD?
NO
YES
SEND ACK.
YES
SEQ. HEADER GOOD?
NO
PARITY BAD TWICE IN A ROW?
NO
YES
NO
RECEIVE LOGIC HIGH PATTERN & RECORD ERRORS
RECEIVE FOR LOGIC LOW PATTERN & RECORD ERRORS
SEND LOGIC HIGH ON ALL WIRES
SEND LOGIC LOW ON ALL WIRES
REROUTE TO SPARE WIRE

Figure 11 - ARQ Parity Receiver State Flow

The demonstration of this technology took place on FPGA development boards. A one byte data field was selected for the sake of brevity and to directly compare to the TMR demonstration. The design is scalable to the desired amount of data bits required. The system implementation took the form as shown in Figure 13. The results of the demonstration when subjected to single fault conditions at a time proved the ARQ system can detect, recover, and fix the parallel bus giving it immunity to SET's and hard faults. Figure 14 illustrates the timing of transferring data under normal, SET, and hard fault conditions. The required time to recover from an SET and hard fault decreases or increases with the timeout interval.

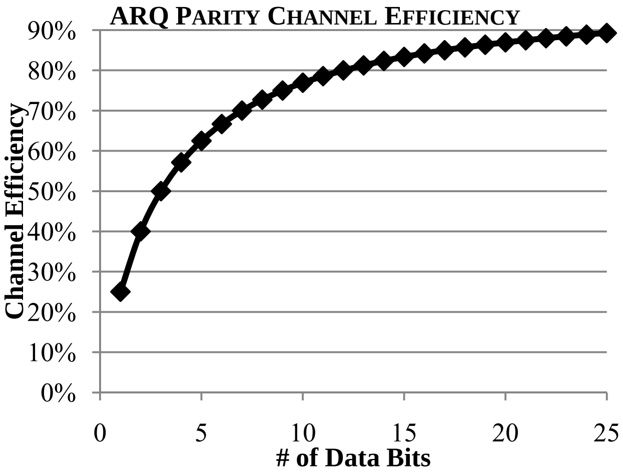


Figure 12 – ARQ Parity Channel Efficiency

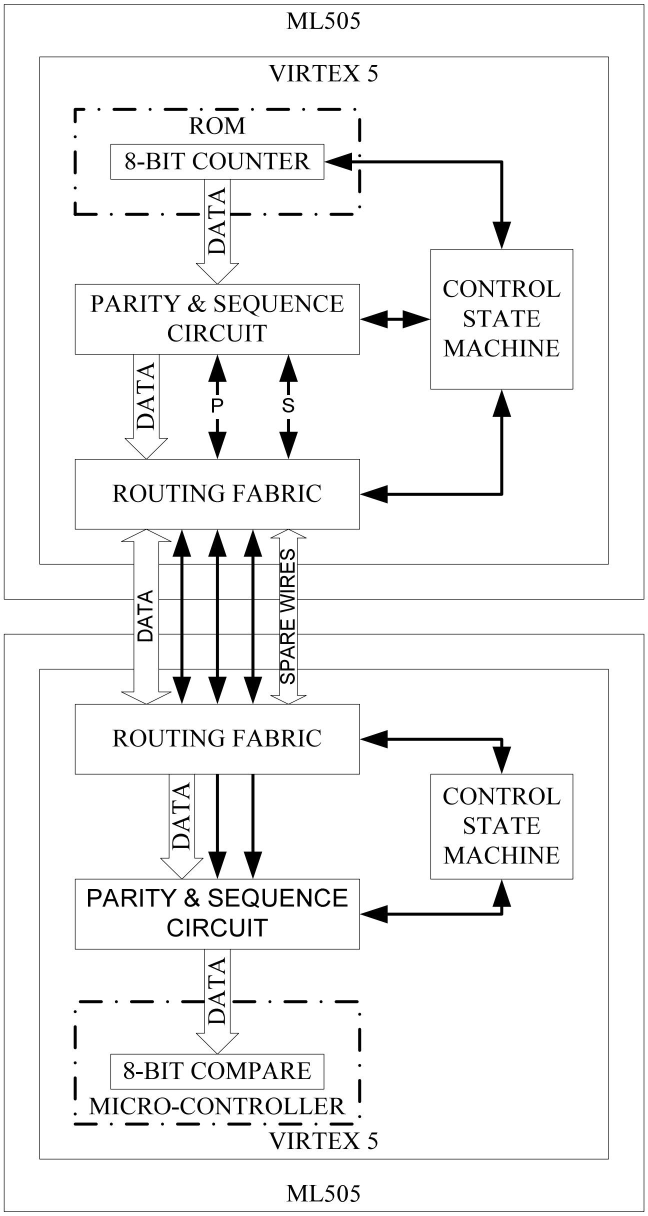


Figure 13 - ARQ Parity System Diagram

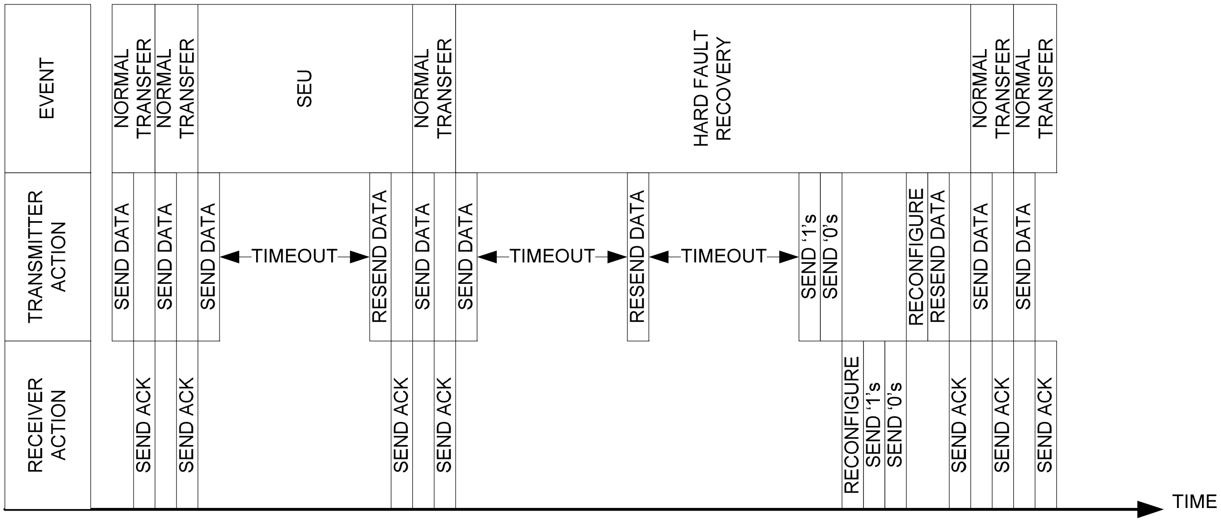


Figure 14 - ARQ Parity Timing Diagram

Despite the ability to detect, correct, and recover from errors the parallel ARQ parity example suffers from serious disadvantages. The main disadvantage is the low data rate. The complexity of this method is also extreme when compared to other forms of error detection and correction. The bandwidth of this system is half the bus total bandwidth since an acknowledgement is required for each data transfer. The latency of this system is higher than the TMR and Hamming example. In addition, latency increases given an error condition.

There is one advantage to this method of communication. This system, in larger parallel bus systems, requires fewer wires than the Hamming code and TMR example. If wire mass is a concern and a parallel bus is required per requirement or availability, then an ARQ system that uses parity is acceptable.

5. Ethernet

The first serial technology application uses 10Base-T or 100Base-T Ethernet, as the physical layer, to detect and correct communications errors. 10Mb/s and 100Mb/s Ethernet both use differential pairs as the electrical standard. Considering an 802.3 MAC frame, as shown in Figure 15, the preamble exists to allow the receiver to lock onto the bit clock and recover the remainder of the frame. The destination and source address is the Media Access Control (MAC) address of the two end-points. The frame type identifies the type of embedded protocol in the data frame, or the size of the data field transmitted. The Frame Check Sequence (FCS) is capable of detecting multiple bit transmission errors. The FCS is a 32-bit Cyclic Redundancy Check with a $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$ polynomial. The inter-frame gap is a length of time required between packets. Considering a maximum length frame of 1538 bytes an Ethernet frame is very efficient.

$$\eta = \frac{Payload\ size}{Payload\ size + 38}$$

$$\eta_{\max size} = \frac{1500}{1500 + 38} = 97.5\%$$

A communication link with multiple Ethernet ports between the two points can be used to detect and recover from SET's and hard faults such as wire damage. The FCS embedded in the Ethernet frame is the method of detecting errors. A command and response state flow can be used to respond to errors and switch away from damaged wires.

When a data packet, ideally a maximum length packet, is transmitted the receiver checks the FCS and responds with a minimum length packet to inform the transmitter if the frame was received properly or not. Since Ethernet is serial technology if a packet is received then the wire integrity is good. The FCS check guards against SET's. If a bad packet is received a response is sent to request the same packet again; otherwise a new packet is requested.

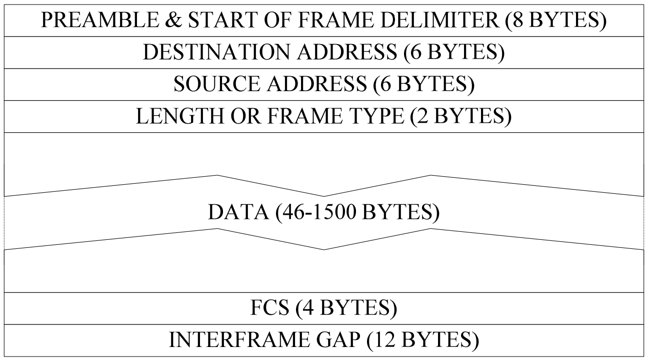

PREAMBLE & START OF FRAME DELIMITER (8 BYTES)
DESTINATION ADDRESS (6 BYTES)
SOURCE ADDRESS (6 BYTES)
LENGTH OR FRAME TYPE (2 BYTES)
DATA (46-1500 BYTES)
FCS (4 BYTES)
INTERFRAME GAP (12 BYTES)

Figure 15 - 802.3 MAC Ethernet Frame

To account for the instance when the response line is damaged, both physical layers send the response packet. If the transmitter does not receive a response packet from both Ethernet links then the receiver did not receive the packet and the current transmit line is damaged. In this case the transmitter switches to the backup and resends the packet. If the transmitter receives a response packet on one or the other links then that corresponding response line is damaged. The transmitter mitigates SET's in the response packets since both links receive the identical packet. If one FCS fails and the other does not then the valid packet is used. However, if one link is already known bad and a FCS fails in a response packet, then the transmitter must request the response packet again. Following the state flow diagrams in Figure 16 and 17, for the transmitter and receiver respectively, illustrates the process required to detect and switch from wire damage. The theoretical efficiency of this error detection and recovery scheme would be

$$\eta = \frac{t_d}{t_{dp} + t_{rx} + t_{ap} + t_{tx}}$$

$$t_d = time\ to\ transmit\ data\ bits\ only$$

$$t_{dp} = time\ to\ tx\ data\ packet$$

$$t_{rx} = receiver\ processing\ delay$$

$$t_{ap} = time\ to\ tx\ ack.packet$$

$$t_{tx} = transmitter\ processing\ delay$$

Ideally with t_{rx} and t_{tx} zero the maximum efficiency of this system is 92.5%

The system was implemented on an FPGA development board using Xilinx's CORE generator to provide the base line dual Ethernet physical layer. To provide a simple and quick demonstration a 10Mb/s link was selected. The simple frame generator provided data to the Ethernet interface and was controlled by a Finite State Machine (FSM). The FSM controlled the operation and response of the Ethernet physical layers as described in Figure 15 and 16. The system diagram is shown in Figure 18. In the best observed situation the channel efficiency of this Ethernet system is 89.6%. The timing diagrams, which show experimental results for normal, SET, and wire damage operation are illustrated in Figure 19, 20 and 21 respectively.

This dual Ethernet system offers many advantages over the parallel demonstrations. This system offers a low wire mass solution that can still communicate over a long cable length. It is less complex than the ARQ parity example, yet more complex than the Hamming code and TMR demonstration. The sustained channel efficiency is higher in the Ethernet system compared to the parallel systems. The only observed disadvantage to this scheme is higher data latency than the Hamming and TMR example, but less latency than the ARQ Parity example.

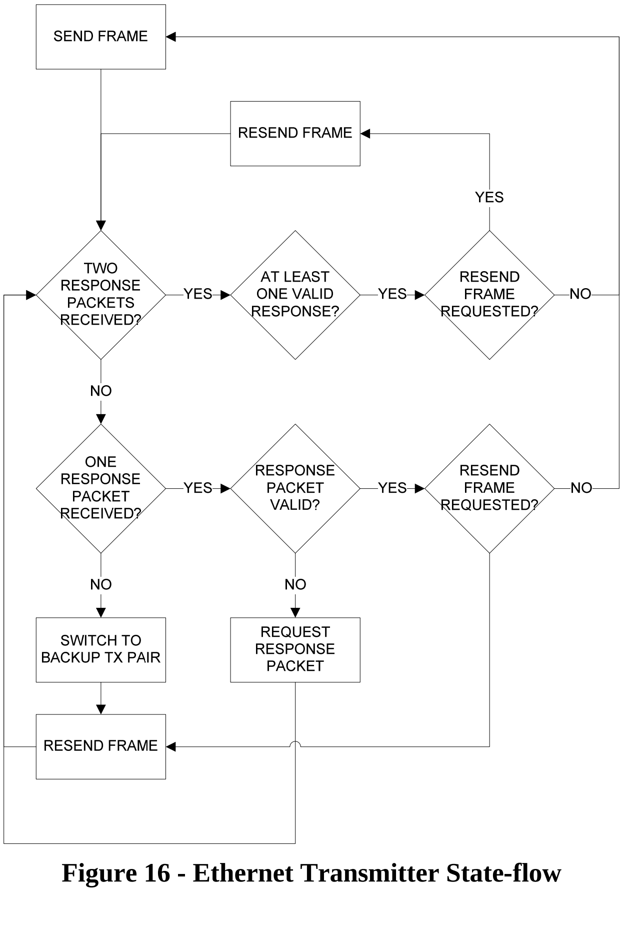


Figure 16 - Ethernet Transmitter State-flow

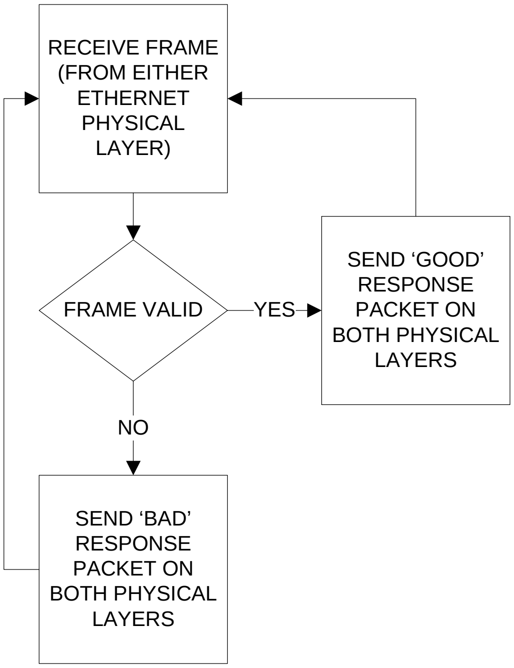

RECEIVE FRAME (FROM EITHER ETHERNET PHYSICAL LAYER)
FRAME VALID
YES
SEND 'GOOD' RESPONSE PACKET ON BOTH PHYSICAL LAYERS
NO
SEND 'BAD' RESPONSE PACKET ON BOTH PHYSICAL LAYERS

Figure 17 - Ethernet Receiver State-flow

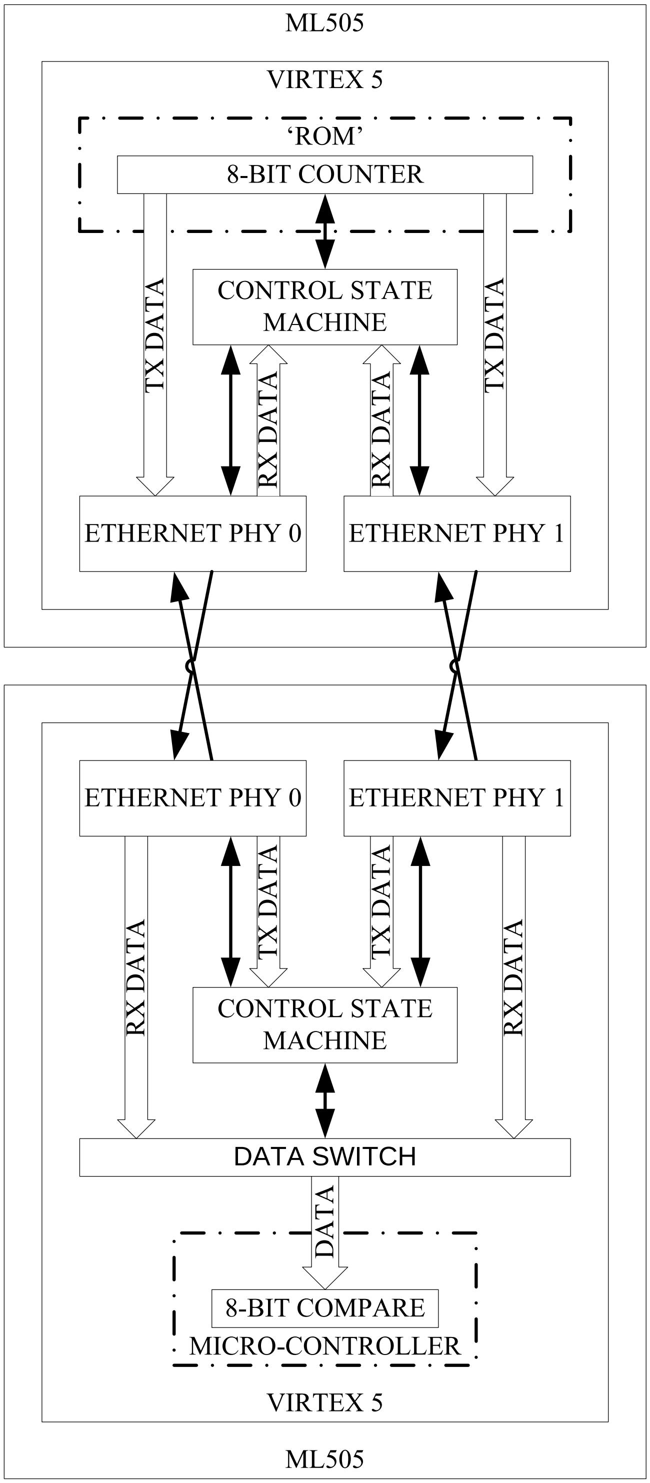


Figure 18 - Ethernet System Block Diagram

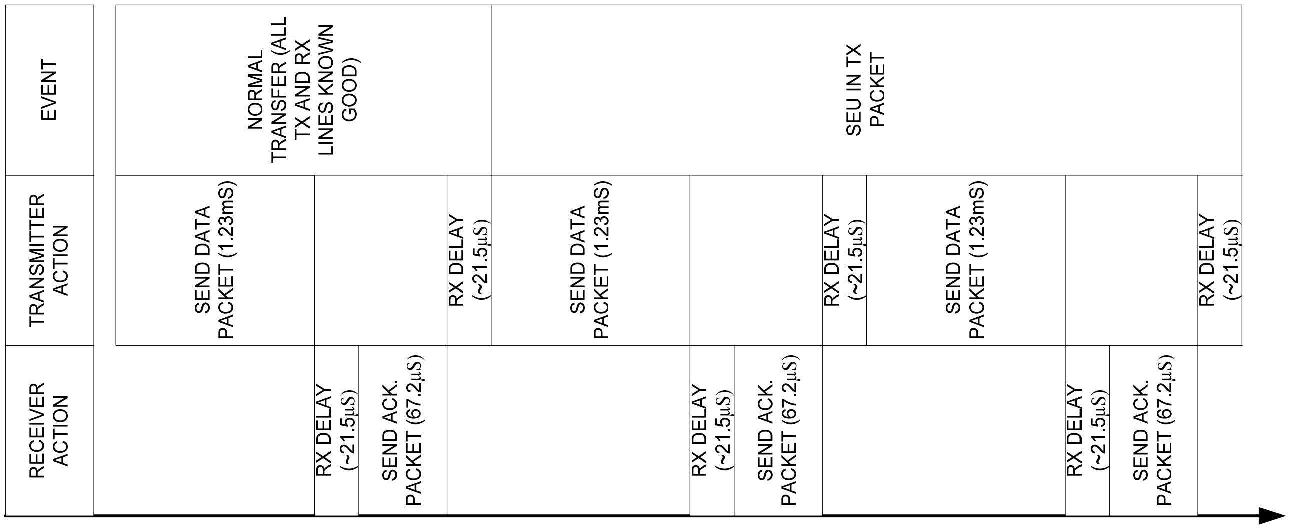

EVENT
TRANSMITTER ACTION
RECEIVER ACTION
NORMAL TRANSFER (ALL TX AND RX LINES KNOWN GOOD)
SEU IN TX PACKET
SEND DATA PACKET (1.23mS)
RX DELAY (~21.5µS)
SEND DATA PACKET (1.23mS)
RX DELAY (~21.5µS)
SEND DATA PACKET (1.23mS)
RX DELAY (~21.5µS)
RX DELAY (~21.5µS)
SEND ACK. PACKET (67.2µS)
RX DELAY (~21.5µS)
SEND ACK. PACKET (67.2µS)
RX DELAY (~21.5µS)
SEND ACK. PACKET (67.2µS)

Figure 19 – Normal Operation Ethernet Timing Diagram

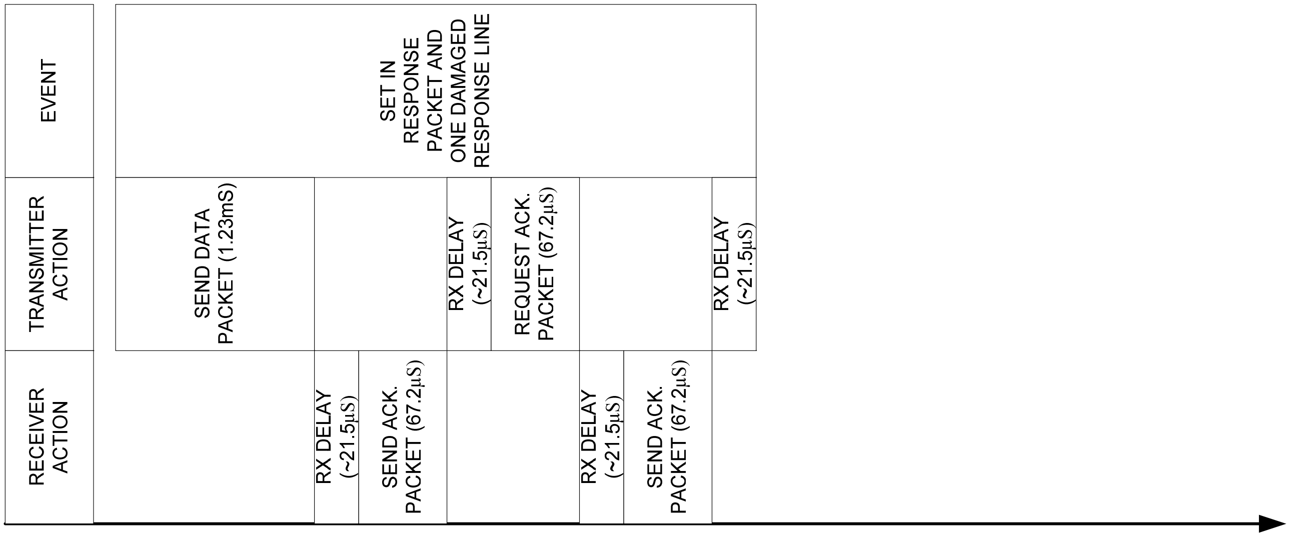

EVENT
TRANSMITTER ACTION
RECEIVER ACTION
SET IN RESPONSE PACKET AND ONE DAMAGED RESPONSE LINE
SEND DATA PACKET (1.23mS)
RX DELAY (~21.5µS)
REQUEST ACK. PACKET (67.2µS)
RX DELAY (~21.5µS)
RX DELAY (~21.5µS)
SEND ACK. PACKET (67.2µS)
RX DELAY (~21.5µS)
SEND ACK. PACKET (67.2µS)

Figure 20 – SET Operation Ethernet Timing Diagram

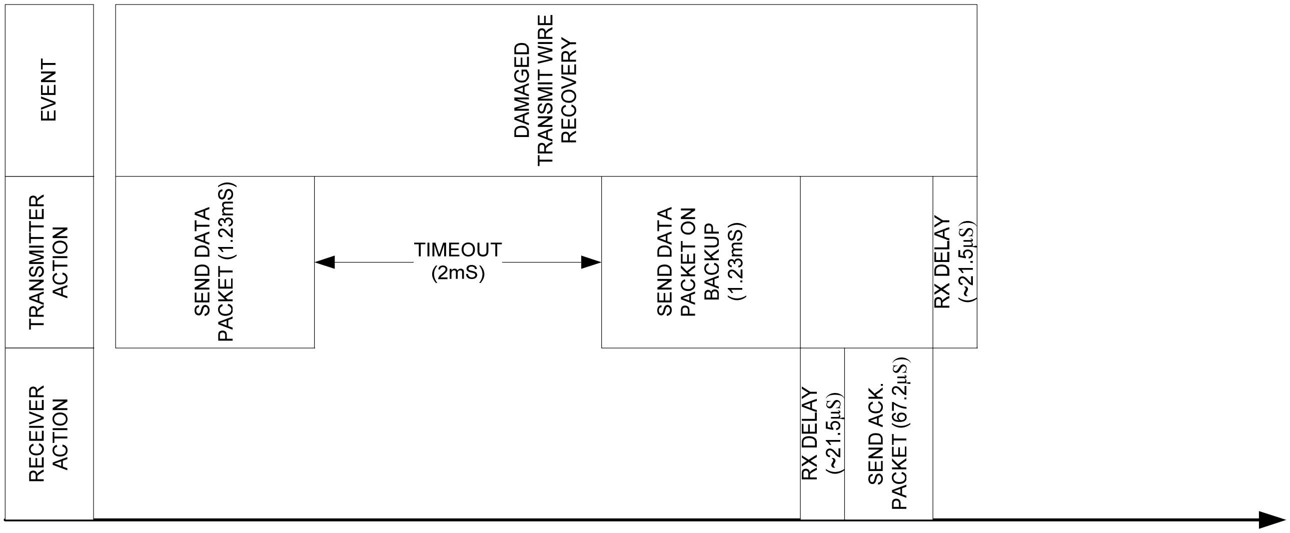

EVENT
TRANSMITTER ACTION
RECEIVER ACTION
DAMAGED TRANSMIT WIRE RECOVERY
SEND DATA PACKET (1.23mS)
TIMEOUT (2mS)
SEND DATA PACKET ON BACKUP (1.23mS)
RX DELAY (~21.5µS)
RX DELAY (~21.5µS)
SEND ACK. PACKET (67.2µS)

Figure 21 – Damaged Transmit Wire Operation Ethernet Timing Diagram

6. Conclusion

Non-masking and masking methods of error detection and correction have been overviewed and analyzed. Masking techniques in general provide a low latency communication link since errors can be detected and corrected, within limits, at the receiving endpoint. Masking methods by the same virtue of detecting and correcting errors can also detect and recover data from damaged wires. The disadvantage of masking techniques is they take away from the total bandwidth of the applicable communication link. However, the bandwidth hit is negligible when compared to non-masking techniques.

Non-masking methods are just a capable of detecting errors. However, in general they increase communication latency since they are not capable of correcting data without retransmitting the data. To recover from hard faults, non-masking systems must switch from damaged links to backups which increases the systems complexity and adding to the latency problems. The bandwidth of non-masking techniques is severely affected since command and response methods state flow is required to verify proper reception of data.

Regardless of non masking or masking techniques, wire mass concerns need to be addressed. Parallel systems require massive amounts of mass when compared to serial technology. The problem is compounded when spare wires are added to provide redundancy.

References

[1] J. Wall, "The past, present and future of EEE Components for Space Application", IEEE International, 1998.

[2] R. Hillman, "Space Processor Radiation Mitigation and Validation Techniques for an 1800 MIPS Processor Board".

[3] J. Bahr, "Design of a Processor-to-Memory I/O Interface with Automatic recovery of Line Failures".

[4] D. Mackay, "Information Theory, Inference, and Learning Algorithms", Cambridge University Press, 2003.

Biographies

Robert L. Akamine *is an Electronic Instrumentation Systems Engineer for NASA's Langley Research Center (LaRC) in Hampton, VA. He received his B.S. in Electrical Engineering from Capitol College in Laurel, MD in 2007. After graduation, Mr. Akamine began his career at NASA's Goddard Space Flight Center researching and developing digital X-band TDRSS waveform transponders. His current contributions at LaRC include a data recorder design for STORRM, and hardware image processing for ALHAT.*

Robert F. Hodson *is the Chief Engineer for Electronic Systems Branch at NASA Langley and is the current Avionics Lead for Constellation's Software and Avionics Integration Office (SAVIO). In this role he has participated and led multiple review and risk assessment activities and has been the lead of the Common Avionics and Software Team. He is presently a core member of the NESC's Avionics team. Dr. Hodson has over 28 years experience as an engineer, of which 20 have been since completion of his Ph.D. in Computer and Information Science from the Florida State University. Dr. Hodson has authored or co-authored over 35 publications including technical papers, reports, and one book. In addition to his Ph.D., Dr. Hodson holds a Master's degree in Computer Engineering from the University of Central Florida and a dual Bachelor's degree in Electrical Engineering and Computer Science from the University of Connecticut where he graduated Magna Cum Laude.*

Brock J. LaMeres *(M'98-SM'09) received the B.S. degree in electrical engineering from Montana State Univ., Bozeman in 1998, and the M.S. degree in electrical engineering from the Univ. of Colorado, Colorado Springs in 2001, and the Ph.D. degree in electrical engineering from the Univ. of Colorado, Boulder in 2005. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Montana State University (MSU), Bozeman. LaMeres teaches and conducts research in the area of digital systems.*

Robert E. Ray, Jr. *received the B.S. degree in computer science from Grand Valley State College in 1984, and the M.S. and Ph.D. degrees in computer science from the University of Alabama in Huntsville, in 1989 and 1996 respectively. He is currently the reconfigurable computing task leader for NASA's (Jacobs Engineering Group) Advanced Avionics and Processors Systems project at the Marshall Space Flight Center in Huntsville, AL.*