ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Implementation of Parity Prediction Scheme in OLS for the Detection of Concurrent Errors

Naresh Yarramsetti
M.Tech Student,
Dept of ECE,
Anurag Engineering College.

Dr. N.Ravi Kumar, M.Tech,Ph.D
Professor,
Dept of ECE,
Anurag Engineering College.

M.Basha, M.Tech
HOD & Associate Professor,
Dept of ECE,
Anurag Engineering College.

Abstract:

An error-correcting code is an algorithm for expressing a sequence of numbers such that any errors which are introduced can be detected and corrected (within certain limitations) based on the remaining numbers. Orthogonal Latin squares of order n over two sets S and T, each consisting of n symbols, is an n×n arrangement of cells, each cell containing an ordered pair (s,t), where s is in S and t is in T, such that every row and every column contains each element of S and each element of T exactly once, and that no two cells contain the same ordered pair. There are number of mitigation techniques proposed to make sure that the errors do not affect the circuit functionality. Among them, to protect the memories and registers in electronic circuits Error Correction Codes (ECC) are commonly used. In this paper, concurrent error detection and correction technique for OLS encoders and syndrome computation is proposed and evaluated. The proposed method efficiently implements a parity prediction scheme that detects all errors that affect a single circuit node using the properties of OLS codes.

Keywords:

ECC, OLS, Error Detection, Error Correction, Memory

Introduction:

Semiconductor memories are more susceptible to transient-faults (the faults which appear and disappear) due to impingement of alpha particles, electrostatic discharges, glitches, etc. among them alpha particles are the most prominent one. In fact, DRAM chips can be exploited as reliable and cheap alpha particle detectors. The intermittentfaults (the faults which appear, disappear and reappear) can occur due to resistance and capacitance variations and coupling effects.

The transient and intermittent faults cause to generate soft memory errors. A soft memory error is defined as a random event that corrupt the value stored in the memory cell without damaging the cell itself. Such errors are being successfully dealt with ECC, which has gained a tremendous impetus during the recent past.A memory can be made fault-tolerant with the application of an error-correcting code, such as Hamming code, Reed-Muller code, the proposed OLS codes, etc. the mean time between "failures" of a properly designed memory system can be significantly increased with an error-correcting code. In this respect, a system "fails" only when the errors exceed the error-correcting capability of the code. Also, in order to optimize data integrity, the error correcting code should have the capability of detecting the most likely of the errors that are uncorrectable.

This paper presents the OLS scheme to deal with the memory errors, mostly with specific to the critical systems. The techniques have been developed independently and present unique characteristics and advantages. This technique makes use of the hardware and information redundancies simultaneously in the design, which might not present the cost effective solution, but since such techniques are mainly targeted for critical systems, the cost becomes the secondary factor. Employing ECC with the memories, generally increases the cost between 10% to 20%, typically increases the die-area around 20% and may also slow down the speed around 3-4%, but the advantages obtained in terms of reliability and accuracy are incomparable.

OLS codes are based on the concept of Latin squares. A Latin square of size m is an m × m matrix that has permutations of the digits 0, 1,… , m – 1 in both its rows and columns. Two Latin squares are orthogonal if when they are superimposed every ordered pair of elements appears only once.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

These codes have k = m2 data bits and 2tm check bits, where t is the number of errors that the code can correct. For a double error correction code t = 2, and, therefore, 4m check bits, are used. As mentioned in the introduction, one advantage of OLS codes is that their construction is modular. This means that to obtain a code that can correct t +1 errors, simply 2m check bits are added to the code that can correct t errors. The modular property also enables the selection of the error correction capability for a given word size. As mentioned before, OLS codes can be decoded using OS-MLD as each data bit participates in exactly 2t check bits and each other bit participates in at most one of those check bits.This enables a simple correction when the number of bits in error is t or less. The 2t check bits are recomputed and a majority vote is taken. If a value of one is obtained, the bit is in error and must be corrected. Otherwise the bit is correct. As long as the number of errors is t or less, the remaining t –1 errors can, in the worst ase, affect t –1 check bits. Therefore, still a majority of t + 1 triggers the correction of an erroneous bit. In any case, the decoding starts by re computing the parity check bits and checking against the stored parity check bits.

$$H = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} \begin{bmatrix} 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\ 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0 \\ 0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0 \\ 0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0 \\ 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \end{bmatrix} \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} I_{2m}$$

Fig. 1. Parity check matrix for OLS code with $k = 16$ and $t = 1$.

Before describing the proposed error detection techniques, the standard definition of self-checking circuits that are used in this section is presented. During normal, or fault-free, operation, a circuit receives only a subset of the input space, called the input code space, and produces a subset of the output space, called the output code space. The outputs that are not members of the output code space form the output error space. In general, a circuit may be designed to be self-checking only for an assumed fault set. In this brief, we consider the fault set F corresponding to the single stuck-at fault model.A circuit is self-checking if and only if it satisfies the following properties: 1) it is self-testing, and 2) fault-secure. A circuit is self-testing if, for each fault f in the fault set F, there is at least one input belonging to the input code space, for which the circuit provides an output belonging to the output error space.

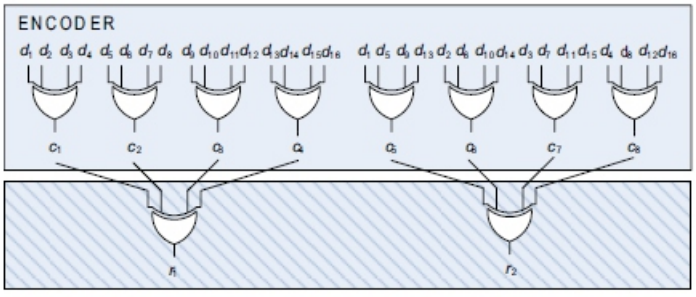


Fig. 2. Proposed self-checking encoder for OLS code with $k = 16$ and $t = 1$.

Proposed self-checking encoder for OLS code with k = 16 and t = 1. A circuit is fault-secure if, for each fault f in the fault set F and for each input belonging to the input code space, the circuit provides the correct output, or an output belonging to the output error space. The fault-secure property guarantees that the circuit gives the correct response, or signals the presence of a fault that provides an output in the error space. Faults are always detected, since there is an input that produces an output that identifies the presence of the fault.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

This property is related to the assumption that the interval between the occurrences of two faults is enough to permit to all the elements belonging to the input code space to appear as circuit inputs before the occurrence of the second fault. Thus, an output belonging to the output error space appears at the circuit output before the occurrence of the second fault.The technique that we propose is based on the use of parity prediction, which is one of the techniques commonly used to detect error in general logic circuits. In our case, the problem is substantially simpler, given the structure of the OLS codes. For the encoder, it is proposed that the parity of the computed check bits (ci) is compared against the parity of all the check equations. The parity of all the check equations is simply the equation obtained by computing the parity of the columns in G. For OLS codes, since each column in G has exactly 2t ones, the null equation is obtained (see, for example, Fig. 1). Therefore, the concurrent error detection (CED) scheme is simply to check.

$$c1 \oplus c2 \oplus c3 \oplus \cdots \oplus c2tm = 0. \quad (4)$$

This enables an efficient implementation that is not possible in other codes. For example, in a Hamming code a significant part of the columns in G has an odd weight and for some codes the number is even larger as they are designed to have odd weights [23]. The input code space of the OLS encoder corresponds to the input space, since the encoder can receive all the possible 2 input configurations. The output code space of the OLS encoder is composed by the outputs satisfying (4), while the output error space is the complement of the output code space.

A fault that occurs in one of the gates composing the OLS encoder can change at most one of the ci check bits. When this change occurs, the OLS encoder provides an output that does not satisfy (4), i.e., an output belonging to the output error space. Hence, this guarantees the fault-secure property for this circuit. Additionally, since the encoder is composed only by XOR gates, no logic masking is performed in the circuit. Therefore, when a fault is activated the error is propagated to the output. This ensures the self-testing property of the circuit. In order to check if the output of the OLS encoder belongs to the output code space or the output error space, a self-checking implementation of a parity checker is used. The checker controls the parity of its inputs and is realized with a repetition code.

The two outputs (r1, r2) are each equal to the parity of one of two disjoint subsets of the checker inputs (ci).When a set of inputs with the correct parity is provided, the output code {r1, r2} takes the values 00 or 11. When the checker receives an erroneous set of inputs, the checker provides the output codes 01 or 10. Also, if a fault occurs in the checker, the outputs are 01 or 10. This guarantees the self-checking property of the parity checker. The proposed encoder is illustrated in Fig. 2 for the code with k = 16 and t = 1.

The proposed circuit can detect any error that affects an odd number of ci bits. For a general code, in most cases there is logic sharing among the computations of the ci bits. This means that an error may propagate to more than one ci bit, and if the number of bits affected is even, then the error is not detected by the proposed scheme. To avoid this issue, the computation of each ci bit can be done separately. This, however, increases the circuit area of the encoder as no logic sharing is allowed. Another option is to control the logic in such a way that errors can only propagate to an odd number of outputs. This would also increase the cost compared to an unrestricted implementation. Additionally, even if the error propagates to an odd number of outputs, the delay of each path can be different. This may cause registering of only some of the output errors at the clock edge.

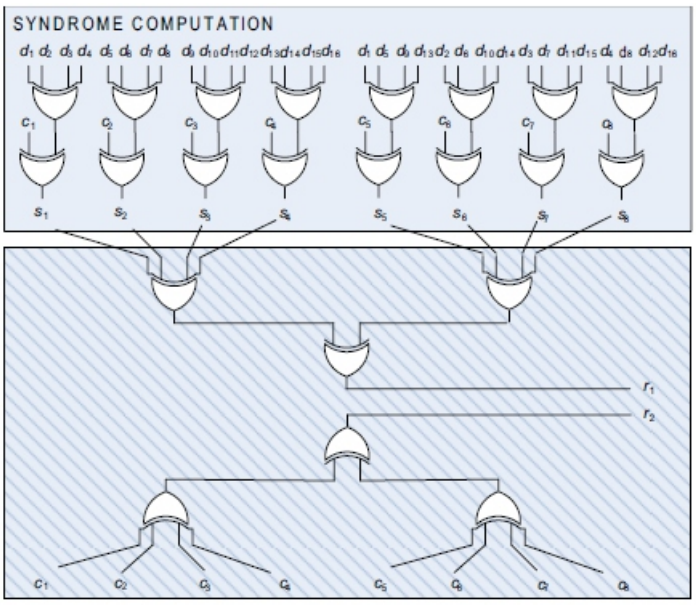


Fig. 3. Proposed self-checking syndrome computation for OLS code with $k = 16$ and $t = 1$.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

For OLS codes, as discussed in the previous section a pair of data bits shares at most one parity check. This guarantees that there is no logic sharing among the computation of the ci bits. Therefore, the proposed technique detects all errors that affect a single circuit node.For the syndrome computation, the parity prediction can be implemented by checking that the following two equations take the same value

$$r1 = s1 \oplus s2 \oplus s3 \oplus \cdots \oplus s2tm \quad (5)$$

$$r2 = c1 \oplus c2 \oplus c3 \oplus \cdots \oplus c2tm \quad (6)$$

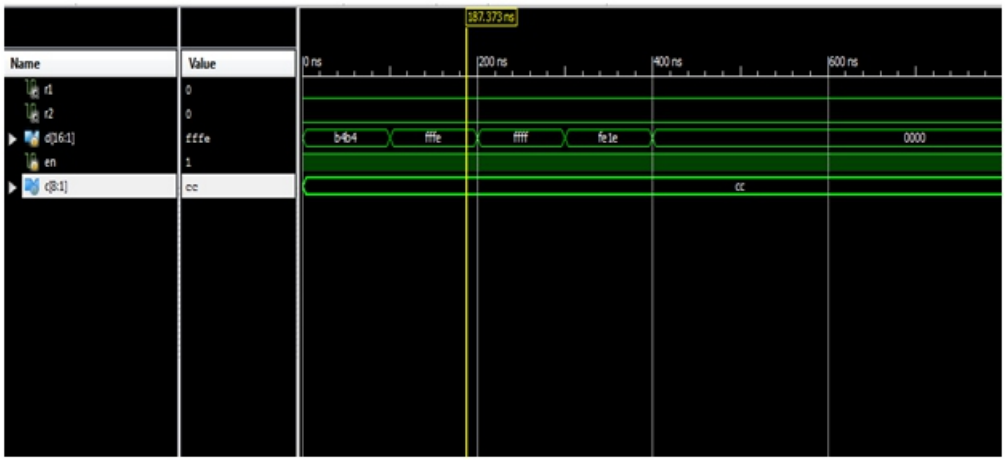
where si are the computed syndrome bits. The proposed circuit is shown in Fig. 3 for the code with k = 16 and t = 1. For syndrome computation, the input code space is only a subset of the possible 2k+2tm input configurations as only up to t errors are considered. This subset is given by the valid OLS codewords and the non-valid OLS code words that are at a Hamming distance of t or less from a valid codeword. Those correspond to the input configurations in which there are no errors or at most t errors on the di inputs such that the errors can be corrected.

The output code space of the OLS syndrome computation is composed by the outputs given by (5) and (6) satisfying r1 = r2, while the output error space is the complement of the output code space. The fault-secure property for the syndrome computation is easily demonstrated for the faults in F by observing that the circuits that compute r1 and r2 do not share any gate and both circuits are only composed of XOR gates. Therefore, a single fault could propagate to only one of the outputs, producing an output on the output error space. To prove the self-testing property for the syndrome computation, suppose that a fault occurs in one of the gates computing (5). If the input configuration is a valid OLS codeword, all the syndrome bits are 0, detecting all stuck-at-1 faults in the XOR gates computing (5). Instead, if the input is a non-OLS codeword that is affected by a t or less errors, some syndrome bits are 1, allowing the detection of a stuck-at-0 faults in the XOR gates computing (5). Finally, suppose that a fault occurs in one of the gates computing (6). Since any combination of the 2tm check bits is allowed, any fault can be activated and the error propagated to the output r2.

SIMULATION RESULTS:

In this diagram, the erroneous data has been compiled and the syndrome has been computed in such a way that the values of R1,R2 are unequal for some of the data inputs which gives conclusion that the syndrome would gives the wrong notation when the data is mismatched or erroneous one.

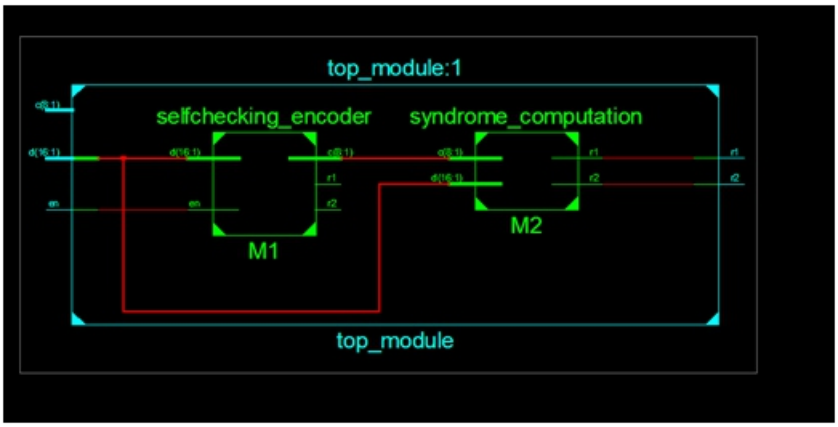
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal



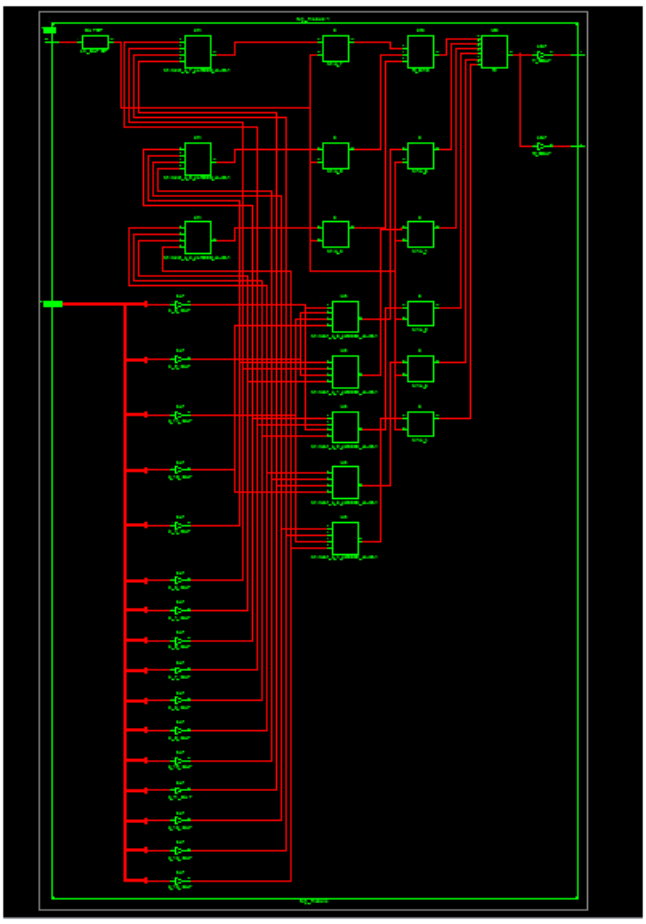
In this diagram, the syndrome has been computed for the error less data and hence the values of R1,R2 are always equal in nature by which we can conclude that the data has not been corrupted or the data is of error free one.

This diagram illustrates the syndrome computation for same data with different computational data by which we get the error notation. Here orange lines indicate us the values of r1, r2 from which we can conclude the error detection.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal



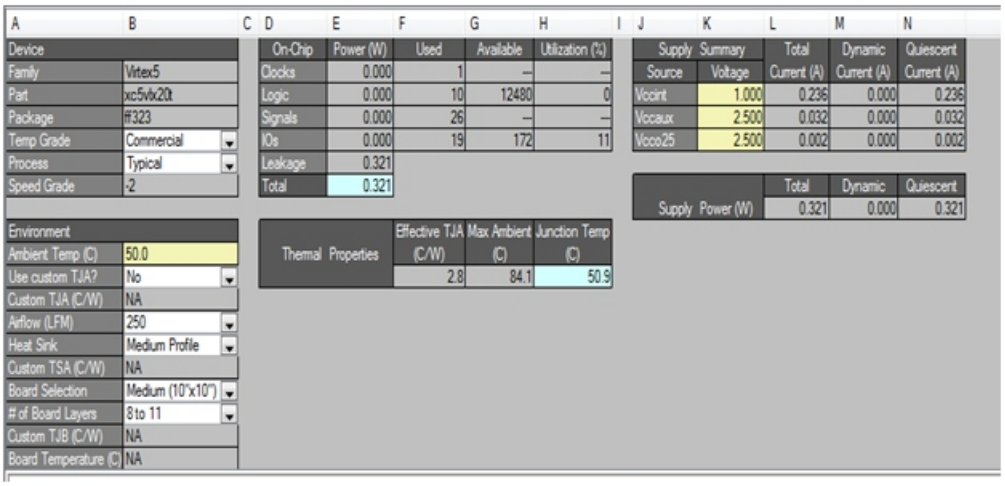
RTL Schematic Diagram



Technology Schematic Diagram

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Power optimization results:



Device	
Family	Virtex5
Part	xc5vlx20t
Package	ff323
Temp Grade	Commercial
Process	Typical
Speed Grade	-2

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	–	–
Logic	0.000	10	12480	0
Signals	0.000	26	–	–
IOs	0.000	19	172	11
Leakage	0.321			
Total	0.321			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.8	84.1	50.9

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.236	0.000	0.236
Vccaux	2.500	0.032	0.000	0.032
Vcco25	2.500	0.002	0.000	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.321	0.000	0.321

Conclusion:

In this brief, a CED technique for OLS codes encoders and syndrome computation was proposed. The proposed technique took advantage of the properties of OLS codes to design a parity prediction scheme that could be efficiently implemented and detects all errors that affect a single circuit node. The technique was evaluated for different word sizes, which showed that for large words the overhead is small. This is interesting as large word sizes are used, for example, in caches for which OLS codes have been recently proposed. The proposed error checking scheme required a significant delay; however, its impact on access time could be minimized. This was achieved by performing the checking in parallel with the writing of the data in the case of the encoder and in parallel with the majority voting and error correction in the case of the decoder.

References:

[1] Pedro Reviriego, Salvatore Pontarelli, and Juan Antonio Maestro, Concurrent Error Detection for Orthogonal Latin Squares Encoders and Syndrome Computation, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 21, NO. 12, DECEMBER 2013

[2] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," IBM J. Res. Develop., vol. 28, no.2, pp. 124–134, Mar. 1984.

[3] E. Fujiwara, Code Design for Dependable Systems: Theory and Practical Application. New York: Wiley, 2006.

[4] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. IEEE VLSI Test Symp., May 2007, pp. 349–354.

[5] H.M. Shao, T.K. Truong, L.J. Deutsch, J. Yuen and LS. Reed, "A VLSI Design of a pipeline Reed-Solomon Decoder", IEEE Trans. Comput., vol. C-34, no.5, pp 393-403, May 1985.

[6] Wicker, Stephen B., Bhargava, Vijay K, "Reed-Solomon Codes and the Compact Disc", IEEE Press ISBN 978-0-7803-1025-4.

[7] J.L. Massey, "Deep Space Communications and Coding: A Match Made in Heaven," in Advanced Methods for Satellite and Deep Space Communications, Lecture Notes in Control and Information Sciences, Volume 182, Berlin: Springer-Verlag, 1992.

[8] Sanjeev Kumar, Rajni Gupta, "Bit Error Analysis of Reed-Solomon Code for Efficient Communication System," International Journal of computer Applications (0975 – 8887) Volume 30- No. 12, September 2011.

[9] V.K Agrawal, Pankaj Goel, Gaurav Mittal, "Review of Reed-Solomon Code for Error Detection and correction," International Journal of Research in IT, Management and Engineering IJRIME Volume2, Issue6 (june-2012) ISSN: 2249-1619.

[10] Sung-woo Choi, sang-Sung Choi, Han-ho Lee, "RS decoder architecture for UWB", wireless home Network Research Team, ETRI ICACT, Feb 2006

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Detecting Functional and Performance Issues in a Network Using Auto Test Packet Generation

Avinash Manne
M.Tech Student,
Department of Computer Science Engineering,
Chilukuri Balaji Institute of Technology.

P. Dharshan
Associate Professor & HOD,
Department of Computer Science Engineering,
Chilukuri Balaji Institute of Technology.

Abstract:

Network monitoring is the use of a system that constantly monitors a computer network for slow or failing components and that notifies the network administrator (via email, SMS or other alarms) in case of outages. It is part of network management. Commonly measured metrics are response time, availability and uptime, although both consistency and reliability metrics are starting to gain popularity. The widespread addition of WAN optimization devices is having an adverse effect on most network monitoring tools -- especially when it comes to measuring accurate end-to-end response time because they limit round trip visibility. Network tomography is an important area of network measurement, which deals with monitoring the health of various links in a network using end-to-end probes sent by agents located at vantage points in the network/Internet. In this paper we examine and implement an Automatic Test Packet Generation (ATPG) method. This approach gets router configurations and generates a device-independent model. ATPG generate a few set of test packets to find every link in the network. Test packets are forwarded frequently and it detect failures to localize the fault. ATPG can detect both functional and performance (throughput, latency) problems.

Keyword: Automatic Test packet, Ping, Network, Performance.

Introduction:

In network management terms, network monitoring is the phrase used to describe a system that continuously monitors a network and notifies a network administrator though messaging systems (usually e-mail) when a device fails or an outage occurs. Network monitoring is usually performed through the use of software applications and tools.

At the most basic level, ping is a type of network monitoring tool. Other commercial software packages may include a network monitoring system that is designed to monitor an entire business or enterprise network. Some applications are used to monitor traffic on your network, such as VoIP monitoring, video stream monitoring, mail server (POP3 server) monitoring, and others. While an intrusion detection system monitors a network for threats from the outside, a network monitoring system monitors the network for problems caused by overloaded and/or crashed servers, network connections or other devices. For example, to determine the status of a webserver, monitoring software may periodically send an HTTP request to fetch a page. For email servers, a test message might be sent through SMTP and retrieved by IMAP or POP3.

Status request failures - such as when a connection cannot be established, it times-out, or the document or message cannot be retrieved - usually produce an action from the monitoring system. These actions vary -- an alarm may be sent (via SMS, email, etc.) to the resident sysadmin, automatic failover systems may be activated to remove the troubled server from duty until it can be repaired, etc. Monitoring the performance of a network uplink is also known as network traffic measurement, and more software is listed there.Route analytics is another important area of network measurement. It includes the methods, systems, algorithms and tools to monitor the routing posture of networks. Incorrect routing or routing issues cause undesirable performance degradation or downtime. Website monitoring service can check HTTP pages, HTTPS, SNMP, FTP, SMTP, POP3, IMAP, DNS, SSH, TELNET, SSL, TCP, ICMP, SIP, UDP, Media Streaming and a range of other ports with a variety of check intervals ranging from every four hours to every one minute. Typically, most network monitoring services test your server anywhere between once-per-hour to once-per-minute.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Monitoring an internet server means that the server owner always knows if one or all of his services go down. Server monitoring may be internal, i.e. web server software checks its status and notifies the owner if some services go down, and external, i.e. some web server monitoring companies check the services status with a certain frequency. Server monitoring can encompass a check of system metrics, such as CPU usage, memory usage, network performance and disk space. It can also include application monitoring, such as checking the processes of programs such as Apache, MySQL, Nginx, Postgres and others.External monitoring is more reliable, as it keeps on working when the server completely goes down. Good server monitoring tools also have performance benchmarking, alerting capabilities and the ability to link certain thresholds with automated server jobs such as provisioning more memory or performing a backup.

Network monitoring services usually have a number of servers around the globe - for example in America, Europe, Asia, Australia and other locations. By having multiple servers in different geographic locations, a monitoring service can determine if a Web server is available across different networks worldwide. The more the locations used, the more complete is the picture on network availability. When monitoring a web server for potential problems, an external web monitoring service checks a number of parameters. First of all, it monitors for a proper HTTP return code. By HTTP specifications RFC 2616, any web server returns several HTTP codes. Analysis of the HTTP codes is the fastest way to determine the current status of the monitored web server. Third-party application performance monitoring tools provide additional web server monitoring, alerting and reporting capabilities.

NETWORK DESIGN and Key Terminology:

As mentioned in the last section, the automatic test packet generation (ATPG) system makes use of geometric model of header space analysis. This section explains some of the key terms associated with geometric framework of header space analysis.

Packet:

Packet in a network can be described as a tuple of the form (port, header) in such a way that, it is the job of port to show position of packet in a network at instantaneous time. Each one of the port is allotted with one and only one unique number.

Switch:

Another term used in geometric model of header space analysis is switches. It is the job of switch transfer Function T, to model devices in a network. Example of devices can be switches or routers. There is a set of forwarding rules contained in each device, which decides how the packets should be processed. When a packet comes at a switch, a switch transfer function comperes it with each rule in descending order of priority. If packet does not match withany of the rule then it is dropped. Each incoming packet is coupled with exactly single rule.

Rules:

Piece of work for rules is generation of list of one or more output packets associated with those output ports to which the packet is transferred, and explain how fields of port are modified. In other words, rules explains how the region of header space at entrance in changed into region of header space at exit.

Rule History:

At any moment, every packet has its own rule history, which can be described as ordered list of rules packet have matched up to that point as it covers the network. Rule history provides necessary and important unprocessed material for automatic test packet generation (ATPG). That is the reason why it is fundamental to ATPG.

Topology:

The network topology is modeled by topology transfer function. The topology transfer function gives the specification about which two ports are joined by links. Links are nothing but rules that forwards a packet from source to destination with no modification. If there is not a single topology rule matching an input port, the port is situated at edge of a network and packet has come to its desired destination.

Life of a Packet:

One can see life of a packet as carrying out or executing switch transfer function and topology transfer function at length. When a particular packet comes in a network port p, firstly a switch function is applied to that packet. Switch transfer function also contains input port pk.p of that packet. The result of applying switch function is list of new packets [pk1, pk2, pk3,].