A NOVEL METHOD FOR INSERTION OF OVERHEAD INFORMATION INTO SONET FRAMES

By

Michael Y. Ho

B. A. Sc., Simon Fraser University,
Burnaby, British Columbia

PROJECT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREEE OF MASTER OF ENGINEERING IN THE SCHOOL OF ENGINEERING SCIENCE

SIMON FRASER UNIVERSITY
November 2002

APPROVAL

NAME: Michael Y. Ho

DEGREE: Master of Engineering

TITLE OF PROJECT: A Novel Method for Insertion of Overhead Information into SONET Frames

EXAMINING COMMITTEE:

Chairman: Dr. James K. Cavers

Senior Supervisor: Dr. Paul K.M. Ho
Professor of the School of Engineering Science

Supervisor: Dr. Shawn Stapleton
Professor of the School of Engineering Science

External Examiner: Oon-Sim Ang
Engineering Manager, West Bay Semiconductor

External Examiner: Dr. Chong T. Ong
Director of Engineering, West Bay Semiconductor

DATE APPROVED: Nov 28, 2002

PARTIAL COPYRIGHT LICENSE

Title of Thesis/Project/Extended Essay

A NOVEL METHOD FOR INSERTION OF OVERHEAD INFORMATION INTO SONET FRAMES

Author:

(signature) _____

(name) Michael Ho

(date) Nov 28, 2002

ABSTRACT

With telecommunication carriers troubled with debt in today's market, the need to generate more revenue using existing standards has spurred demand for higher density and more feature upgrades for the existing systems in the telecommunication network. Synchronous Optical NETwork (SONET) is one such standard that is widely deployed by telecommunication carriers worldwide. Recent advances in the areas of Very Large Scale Integration (VLSI) semiconductor design and high-speed electronic board design are enabling communication equipment manufacturers to pack more into less space, however, not without some creative engineering.

One specific problem of packing more bandwidth and features into a single integrated devices is the large number of inputs and outputs (I/O's) needed, especially at the physical line interface of SONET equipments. One example of such a feature is the mechanism for insertion of overhead information into a unit of bandwidth. To reduce the number of I/O's, a creative method, different from existing ways, was developed in this project at West Bay Semiconductor, Inc. This report presents the architecture and implementation for this method.

ACKNOWLEDGEMENTS

The author would like to thankfully acknowledge the team at West Bay Semiconductor, Inc for making this project possible. In addition, the author would like to thank SFU professors, Dr. Paul Ho and Dr. Shawn Stapleton for their time and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS & ABBREVIATIONS

ADM	Add Drop Multiplexor
ASIC	Application Specific Integrated Circuit
BLSR	Bi-directional Line Switch Ring
BYTEID	Byte Identification
DS-N	Digital Signal Level N
GOIP	General Overhead Insertion Port
I/F	Interface
I/O	Input and Output
LOH	Line Overhead
OAM&P	Operations, Administration, Maintenance & Provisioning
OC-N	Optical Carrier Level N
OH	Overhead
OHINS	Overhead Insertion
OPCODE	Operation Code
OSI	Open Source Interconnection
POH	Path Overhead
RX	Receive
SDH	Synchronous Digital Hierarchy
SOH	Section Overhead
SONET	Synchronous Optical NETwork
SPE	Synchronous Payload Envelope
STS-N	Synchronous Transport Signal Level N
TX	Transmit
TXOHINS	Transmit Overhead Insertion
UPSR	Unidirectional Path Switch Ring
VT	Virtual Tributary

1 Introduction

This project report introduces the SONET standard highlighting the concepts of rates and formats, the multiplexing procedures, and payload justification. SONET ring networks are introduced before the Add-Drop Multiplexor is examined putting the SONET physical line interface and the problem of overhead insertion into context. The novel method for overhead insertion is discussed. The presented implementation applies to the transmit side of a SONET physical line interface of an OC-48 system.

2 SONET Background

2.1 Introduction to SONET

Synchronous Optical NETwork (SONET) and Synchronous Digital Hierarchy (SDH) are equivalent standards with minor differences. SONET is used widely in North American, and SDH is used in Europe and the rest of the world. For the purpose of this report only SONET terminology is used and discussed. The overhead insertion mechanism, however, can be applied to both.

SONET was originally developed for the telephone network as a long-term solution for a mid-span-meet between vendors. The standard was proposed by Bellcore and was established in 1984. SONET defines the rates and formats, the physical layer, network element architectural features, and network operational criteria for a fiber optic network. The standard soon became an excellent way for data communication as well, because it fits well in the physical layer of the OSI data network model.

To put the topic of overhead insertion into context, SONET rates and formats, multiplexing procedures, and payload justifications are discussed. Also, the Add-Drop Multiplexor, a key network element in SONET networks, is introduced.

2.2 Rates and Formats

2.2.1 Typical End-to-End Connection

Because many existing networks use communication schemes of different digital signal hierarchies, encoding techniques, and multiplexing strategies, the complexity and cost to interconnect these networks are high. To reduce this complexity and cost, SONET was defined to standardized rates and formats for interoperability.

SONET systems are synchronous because all system elements use similar clocks rated at a grade of Stratum 3 or higher. The Optical Carrier (OC) level and the electrical equivalent, Synchronous Transport Signal (STS), are the building blocks used in SONET. A STS consists of two parts: the STS payload and the STS overhead. The STS payload carries the communicated information, while the STS overhead carries signaling and protocol information.

For two user networks to communicate, the signals are converted to a STS, carried through various SONET networks before the SONET terminating equipment converts the STS back to the user network format. As illustrated in Figure 1, four layers exist for the typical SONET end-to-end connection: the path layer, line layer, section layer and photonic layer.

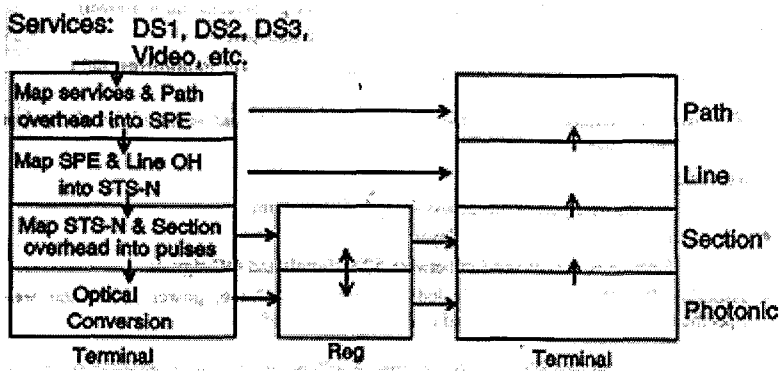


Figure 1 Optical Interface Layers.

Information from each layer is communicated to and processed by the same layer in the terminating equipment and this processed information is passed up and down the layers. Each layer is responsible for specified aspects of the physical interface. The path is responsible for monitoring; the line is responsible for synchronization, multiplexing, and protection switching; the section, for framing, scrambling, and error monitoring; and the photonic layer, for setting the pulse shape, power level, and wavelength.

An example can be illustrated when transporting DS-1 signals. The DS-1 signals are put into Virtual Tributary 1.5 (VT1.5) Containers. The path layer multiplexes 28 VT1.5 containers and inserts path overhead to form a STS-1 payload. The line layer multiplexes 3 STS-1 payload signals and inserts line overhead. The section layer takes these combined signals and performs framing and scrambling and inserts section overhead to form 3 STS-1 signals. The phontonic layer then converts the 3 electrical STS-1 signals to 3 OC-1 optical signals for final transmission.

The process of putting the DS-1 signals into STS signals is called mapping. The clocks of the DS-1 signals are asynchronous to the STS signals, so to accommodate, the STS-1 payload is shifted along using a mechanism called pointer justification, which is described later.

2.2.2 Frame Structure

To understand the problem of overhead insertion, the STS-1 frame is examined. Figure 2 illustrates a STS payload straddling two frames.

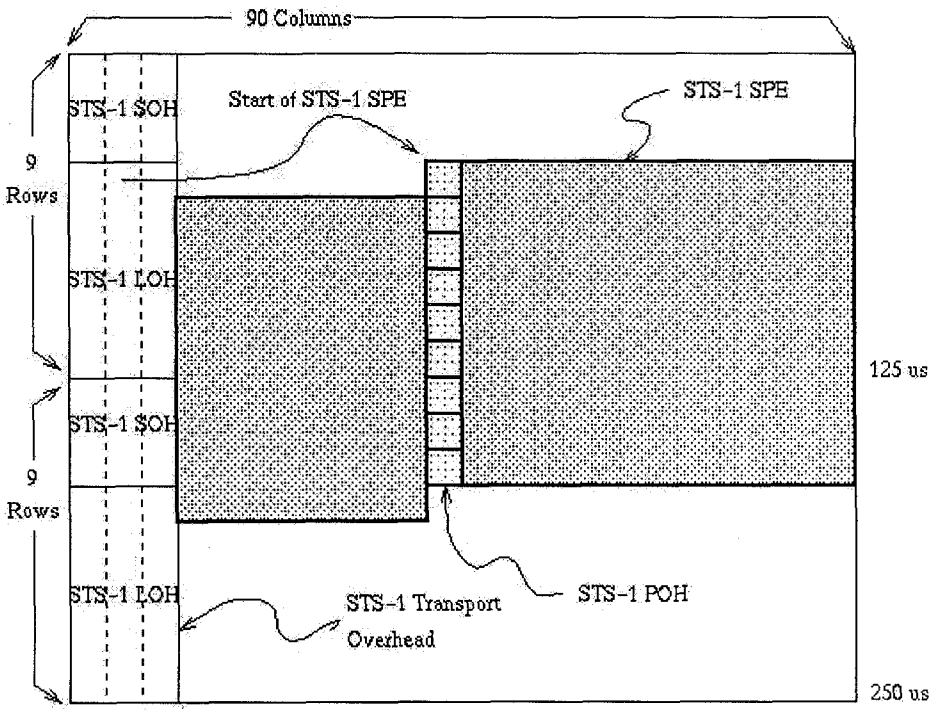


Figure 2 STS-1 Frame.

A STS-1 frame travels at a rate of 8000 frames per second based on the Nyquist frequency of voice (or 2 times 4 KHz). Each frame is made of 810 bytes, which can be organized as 9 rows, and 90 columns. The frame is transmitted serially along a

fiber optic with the most significant bit of each byte transmitted first. The serial stream of the STS-1 signal is therefore sent at a rate of 51.84 Mbps (9 rows X 90 columns X 8000 frames/sec X 8 bits/byte).

The first 3 columns and the first column of the STS payload or Synchronous Payload Envelope (SPE) contain the 9 section, 18 line and 9 path overhead bytes, which together take up 4.44% (4 columns / 90 column in frame X 100 %) of an STS-1 frame. These overhead bytes are responsible for the operations, administration, maintenance and provisioning (OAM&P) of the SONET network.

The section overhead (SOH) bytes are responsible for framing, STS identification, section error checking, order wire voice communication, user communication channel, and section data communication channel. The line overhead bytes (LOH) contain information needed for line error checking, automatic protection switching, line data communication, line order wire, and the location of the start of the STS-1 payload. The first column of the STS-1 payload contains the path overhead (POH) bytes used for STS path identification, path error checking, path signal label, path status, path user channel, and VT multi-frame indicator. Together the SOH, LOH, and POH make up the SONET overhead bytes. Figure 3 shows the SONET overhead bytes arranged in row and column format.

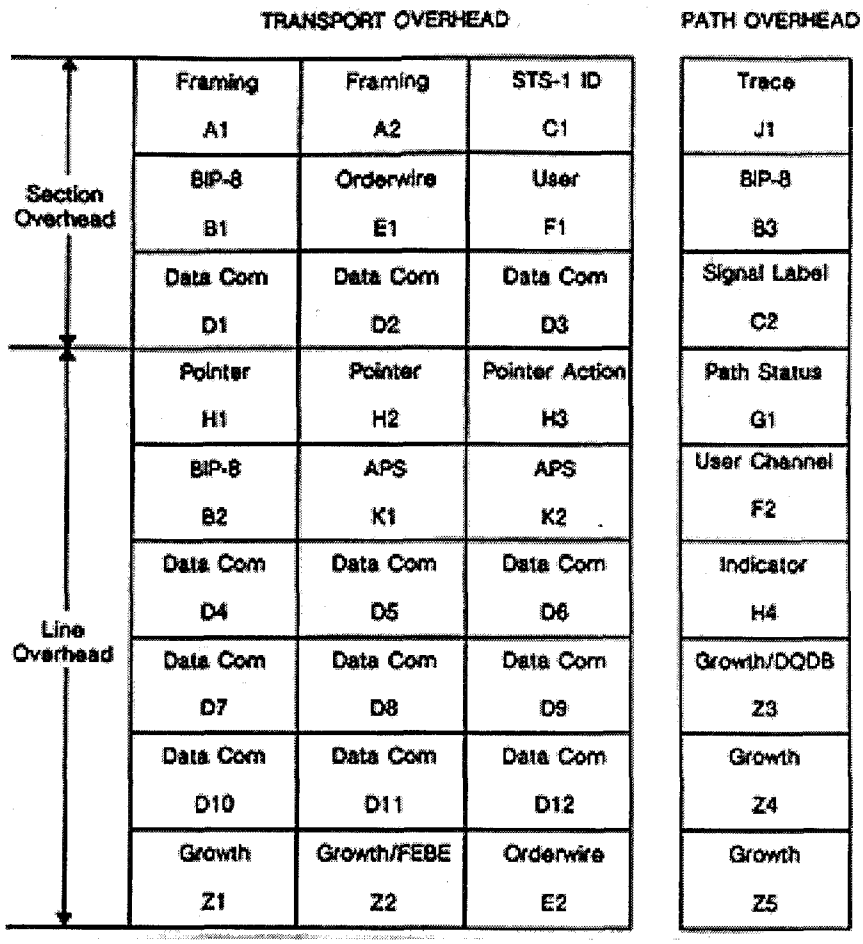


Figure 3 Section and Path Overhead

The overhead insertion mechanism, described later, is a way to insert information into bytes as the STS-1 frame is constructed. Another SONET concept that needs to be discussed to understand the mechanism of overhead insertion is SONET multiplexing procedures.

2.3 Multiplexing Procedures

Because fiber optics are capable of carrying very high rates of signaling, SONET standardizes multiplexing procedures for combining STS-1 signals to larger signals to

maximize utilization. STS-1 signals can be multiplex into larger STS-N containers (commonly N=3, 12, 48, or STS-192). Multiplexing involves interleaving the STS-1 bytes together to form a larger frame that is still carried at a rate of 8000 frames per second; hence a STS-3 is three times the size of a STS-1 so its rate is three times as fast.

A simple example is the multiplexing of three STS-1 signals to form a STS-3. Not all of the transport overhead, namely the section and line overhead bytes, of the three STS-1 is needed in this combined signal; for the most part only one set is needed (See Figure 4; (NA in Figure 4 are Not Applicable)

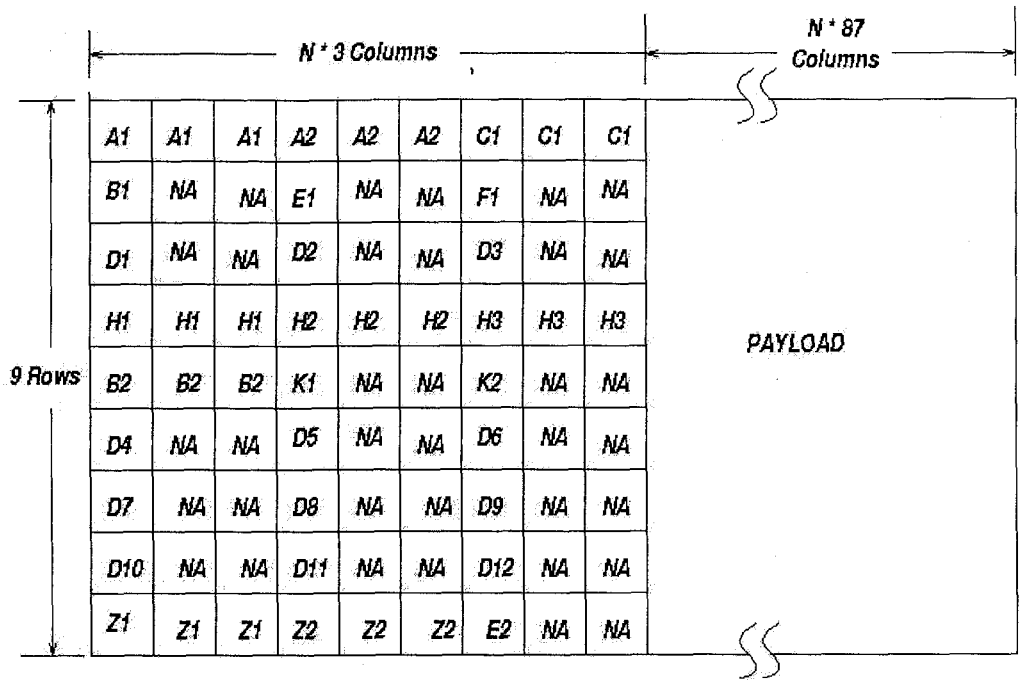


Figure 4 STS-3 Frame.

The bytes from each of the three STS-1 signals are interleaved one at a time. The first byte of the STS-3 frame is from the first STS-1, the second from the second STS-1, the third from the third STS-1, the fourth from the first STS-1, and so on.

However, for larger STS-N signals the "3-to-1" pattern no longer holds. For a STS-12 and upwards, the interleaving is done 4 at a time. Figure 5 illustrates a two-stage multiplexing procedure used to form a STS-12.

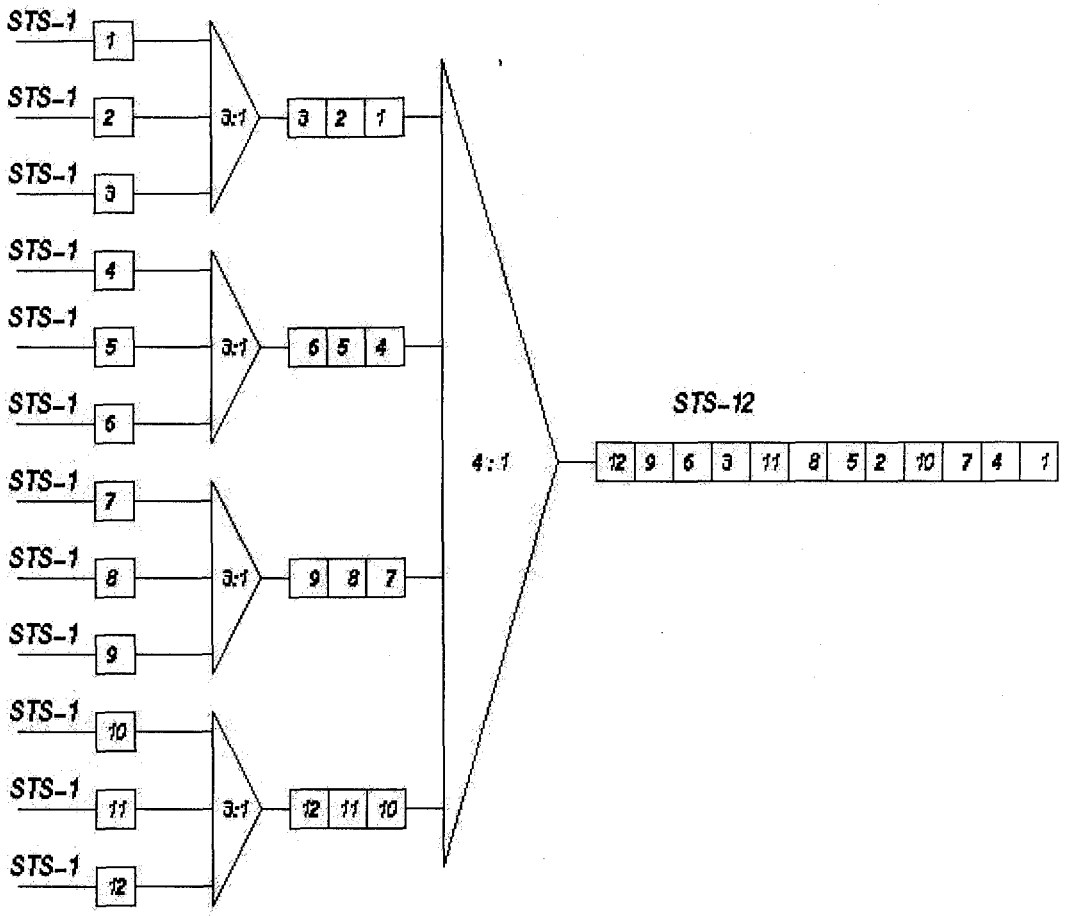


Figure 5 Two-Stage Interleaving.

Three STS-1 signals are combined to form STS-3 signals. Four STS-3 signals are grouped to form a single STS-12 signal. To extend this to larger STS-N signals, four STS-12 is group and interleaved to form a STS-48 signal. Four STS-48 signals are interleaved to form a STS-192 signal, and so on.

The multiplexing procedure and the order in which the bytes are arranged to form a STS-12 signal are important to understand the overhead insertion mechanism discussed later. The next concept that needs to be covered is payload justification,

which introduces the concept of a shifting. This results in a moving path overhead location.

2.4 Payload Justification

Because local networks operate asynchronously to the synchronous SONET network, the location of the STS payload, into which the local network payload is mapped, must be allowed to shift within the STS frame structure. Figure 6 shows how the STS frame facilitates this by allowing payload justifications.

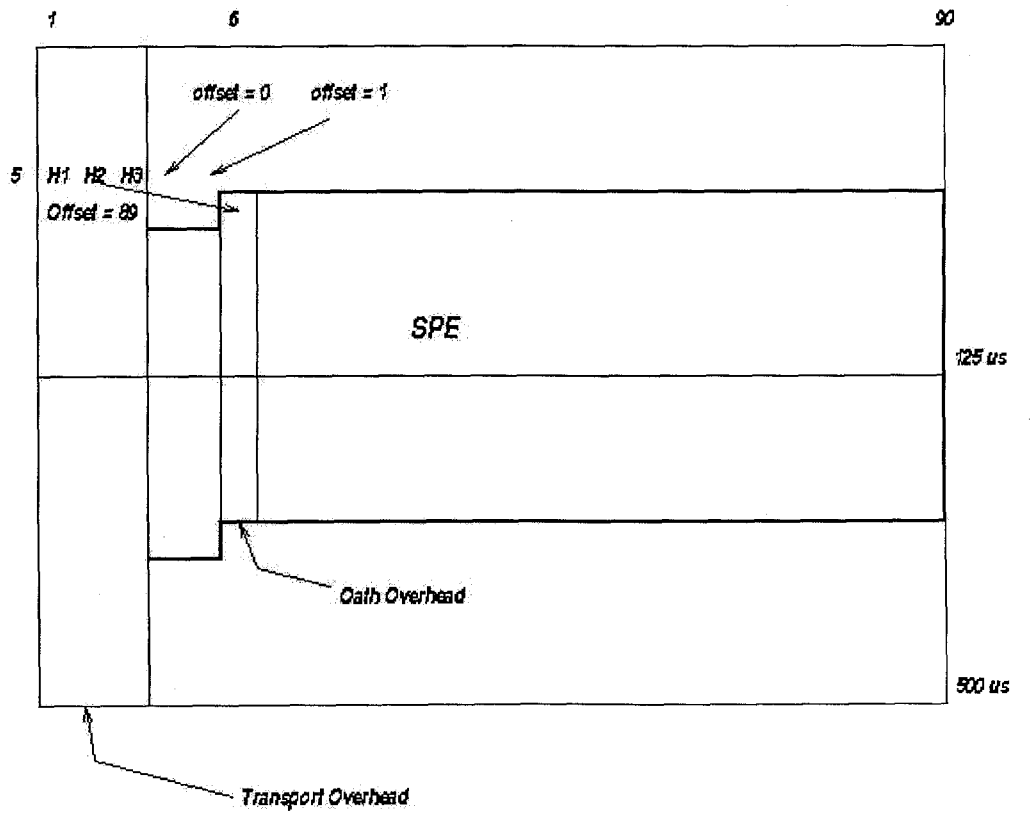


Figure 6 Payload SPE in STS-1 Frame.

In the line overhead of the STS frame, 3 overhead bytes (H1, H2, H3) are used to facilitate payload justifications. H1 and H2 are pointers to indicate the location of the

start of the STS payload. The H3 byte is used as extra space for carrying the extra "overflow" bandwidth of a faster STS payload. The byte following the H3 byte (in column 4 of the STS frame) has fixed stuff inserted when the payload is slow. The justification of this payload is strictly defined in the SONET standards, but for the purpose of this report, we only want to note that the STS payload envelope (SPE) moves with respect to the STS frame. Hence, the path overhead (POH) location moves with respect to the STS frame. Figure 6 shows what this process looks like. This variable location of the POH affects the architecture and implementation of the overhead insertion mechanism.

2.5 Add - Drop Multiplexor

An Add-Drop Multiplexor (ADM) is a key building block in a SONET network. The ADM is responsible for adding and dropping STS signals in and out of a SONET network. A common SONET network topology is the ring as exemplify in Figure 7.

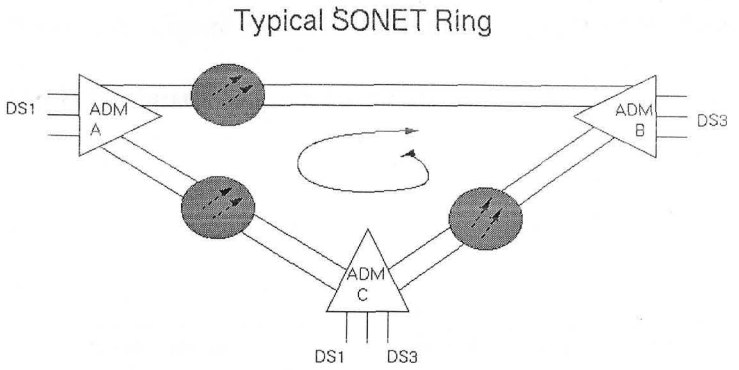


Figure 7 ADM in a SONET Ring.

In the example above, the connections between the three ADMs make out the SONET ring. The ring abides to the SONET standard so it carries STS signals, but connected to the ring are local user networks that may carry a variety of user signals. In Figure 7, ADM A maps DS1 signals to STS signals from the local networks to the SONET ring, and de-maps STS signals back to DS1 signals. ADM B handles DS3 and ADM C handles DS1 and DS3.

The inherent properties of the ring are optimal for traffic protection. For example, traffic can travel into both directions of the ring, so each destination can be reach in two ways allowing redundancy. This redundancy is key to the many automatic protection schemes defined for SONET like Bi-directional Line Switch Ring (BLSR) and Unidirectional Path Switch Ring (UPSR). But for the purpose of this report, the details of these schemes will not be discussed. The function of the ADM, however, is key to understand where the overhead insertion takes place.

To isolate where the overhead insertion mechanism is applied, the block diagram of a basic ADM is examined. Figure 8 shows a block diagram of an ADM that is configured to map DS3 signals into a STS-48 signal.

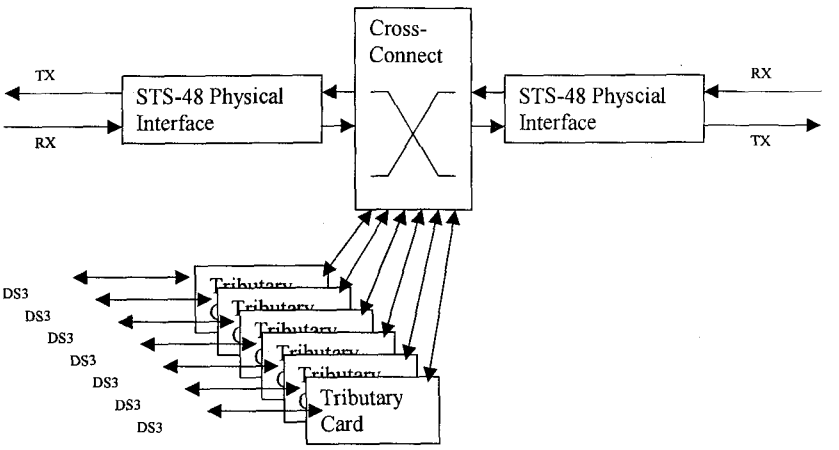


Figure 8 Block Diagram of an ADM.

In the STS-48 ADM example, the overhead insertion mechanism is located in the line card. The physical interface is responsible for generating and processing the STS-48 frames, and it aligns these STS-48 frames to the system clock before it is forwarded to the cross-connect. The tributary cards map the DS3 signals into STS-1 frames, which are fed into the cross-connect. The cross-connect acts as the traffic controller directing STS-1 frames to their destinations. Together, these components make up the function of the ADM.

The overhead insertion mechanism of interest is located on the transmit side of the physical interface. A block diagram for the transmit side of the physical interface is illustrated in Figure 9.

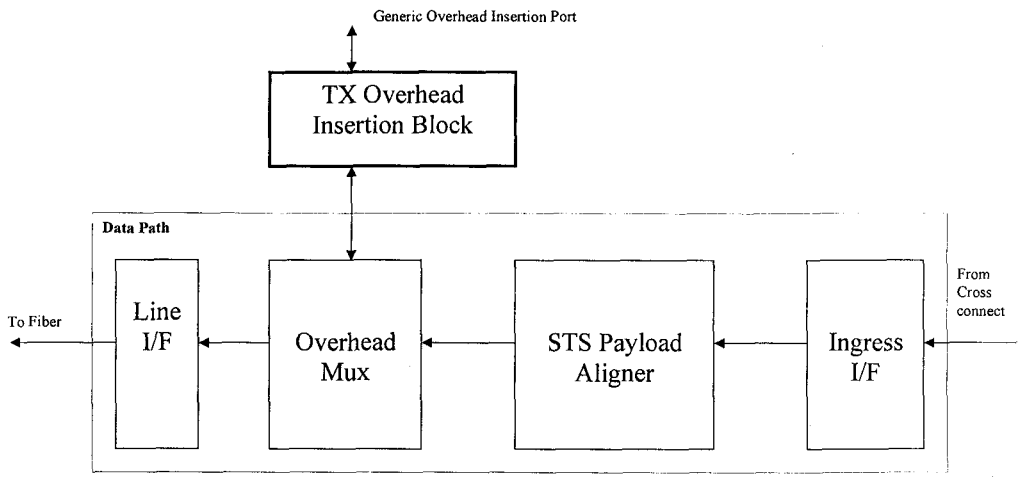


Figure 9 Block Diagram of the Physical Interface Component.

The block diagram shown above is typically implemented in one half of an Application Specific Integrated Circuit (ASIC); the other half is for the receive direction.

Information to be transmitted is carried into the transmit side of the physical interface through the ingress interface. The information is typically encapsulated in a STS frame without overhead information. The STS payload aligner transfers this STS frame from the system clock to the SONET network clock reference. In the overhead multiplexor, the overhead information is inserted into the STS frame. The frame is finally serialized and converted to an optical signal for transmission.

The TX overhead insertion block provides a mechanism for external devices to insert overhead information into the SONET frame, by way of a port on the ASIC. In the past, this mechanism uses a set of three I/Os for synchronization, clock and data for each type of overhead in each STS-1. For the case of STS-48, there would be 48 I/O sets for section, 48 I/O sets for line, and 48 sets for path, for a grand total of 432 I/Os ((48+48+48) X 3 = 432). The reasons for these sets of synchronization, clock and data are the simplicity and flexibility of the interface especially for lower SONET bandwidths. The clocks can be provided at low frequencies thereby simplifying the design of the external interface. Because each set of I/O is responsible for a particular STS-1, the external device does not need decoding logic. Furthermore, many external devices can interface to each set independently. However, as bandwidth increases, this scheme breaks down as shown in the STS-48 case where 432 I/Os are needed. The novel method discussed in this report implements this overhead insertion mechanism using much fewer I/O per unit of bandwidth.

The concept of the novel method is as follows. The method uses timing information from the data path to request for overhead information. Before the need to provide overhead to the data path, the overhead information is already requested from the external device via the overhead insertion (GOIP) port. Therefore, the overhead insertion block buffers up enough overhead information so that this information can be promptly transferred to the overhead multiplexor for insertion into the SONET frame. The implementation of this concept is discussed next.

3 Transmit Overhead Insertion Requirements

The overhead insertion mechanism serves many needs. It provides is a way for an external device to insert proprietary overhead into the SONET frame. It also provides the equipment to manually override the SONET overhead automatically generated upstream in the SONET physical interface ASIC. To satisfy these equipment needs, the requirements of the overhead insertion mechanism is as follows:

- To provide a general external interface to insert overhead information into SONET frames using fewer input / output ports to the SONET physical interface ASIC.
- To provide insertion access to all section overhead (SOH), line overhead (LOH) and path overhead bytes of a SONET frame.
- To provide additional features like an XOR function and replacement masks for the insertion of certain overhead bytes.
- To provide parity checking for the incoming data from the external port.
- To provide all the necessary signals for interfacing with the external device.

4 Transmit Overhead Insertion Implementation

4.1 Overview

The implementation of the transmit overhead insertion (TXOHINS) mechanism was developed at West Bay Semiconductor. This implementation interfaces between four STS-12 data path cores and the general external overhead interface port (GOIP). Figure 10 is a connector diagram depicting the input and output connectors for the transmit overhead insertion (TXOHINS) mechanism.

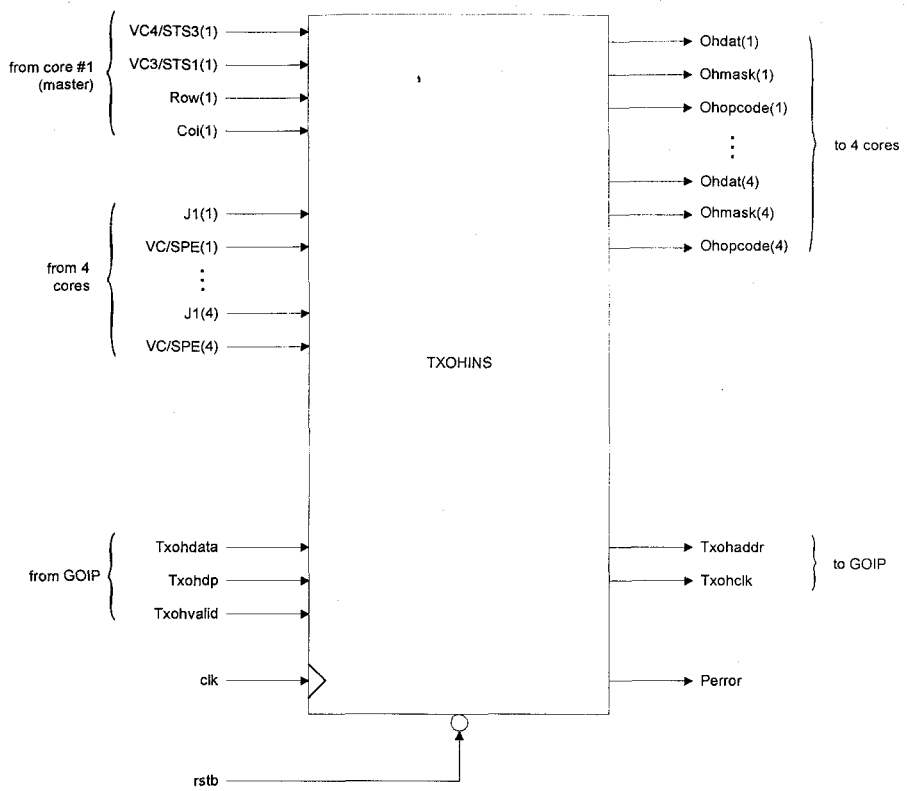


Figure 10 TXOHINS Connector Diagram.

In Figure 10, the connectors on the bottom left and right side of TXOHINS constitutes the GOIP, while the connectors on the top left and right, as well as, the CLK and RST connect to the four STS-12 data path cores.

The Overhead Insertion Port (GOIP) port is an external, user-accessible port on the ASIC. It is used to control and provide information for the insertion of all section overhead (SOH), line overhead (LOH), and path overhead (POH) bytes. The port allows the external device to insert into (or replace) all SOH and POH bytes. The port also provides the ability to replace an overhead byte with a mask and to execute a byte-wide XOR function for some of the overhead bytes.

4.2 Function Description

This section describes the functional sub-blocks of the TXOHINS mechanism. The TXOHINS mechanism consists of 4 unique, major sub-blocks (Figure 11). They are the Core Interfaces, Overhead Address Queues, Overhead Byte Tables (Data Tables), and the Port Interface. The block diagram below shows the instances and their connectivity.

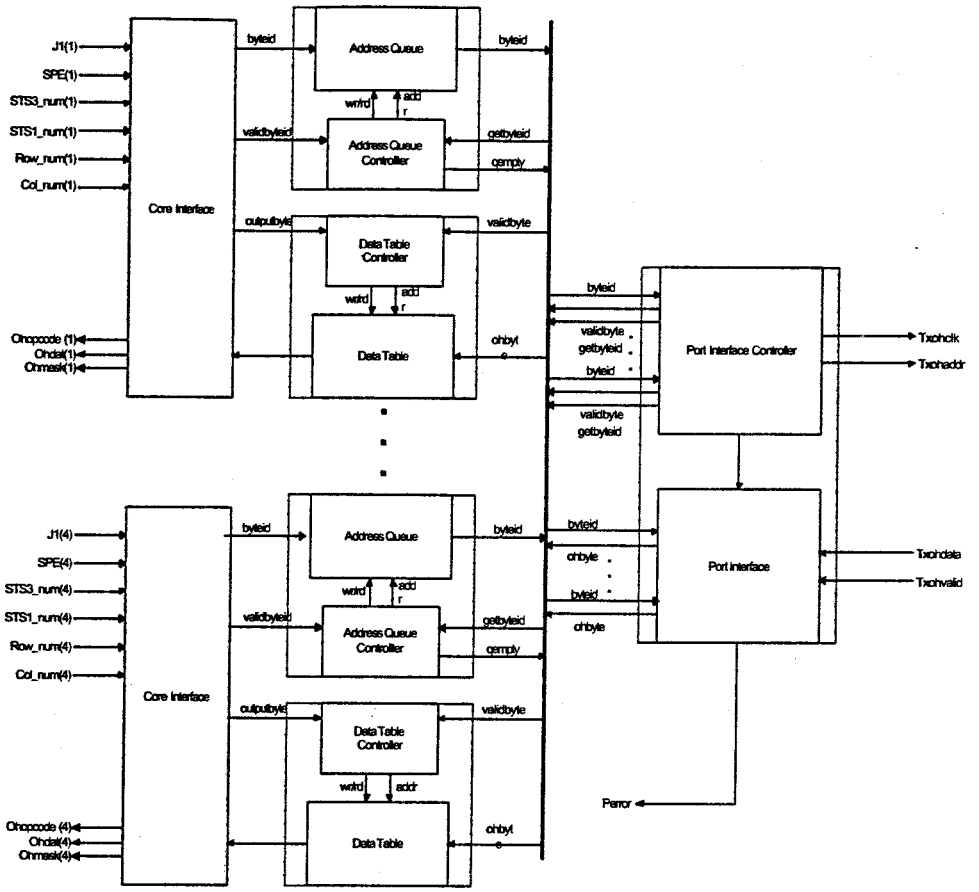


Figure 11 TXOHINS Block Diagram

(Note that the dotted lines surround the "sub-blocks" within the major sub-blocks, namely Address Queue and Data Table.) There are 4 instances of the Core Interface,

4 instances of the Address Queue, 4 instances of the Data Table and 1 instance of the Port Interface.

The Core Interfaces interpret the timing information provided by the data path cores. The timing information consists of the J1 and SPE indicators, STS3_NUM, STS1_NUM, ROW_NUM and COL_NUM labeled in Figure 11. If the timing indicates that the current byte is a SOH or POH byte, the core interface generates a start-of-row indicator (if the current byte is SOH) or a BYTEID (if POH) to be queued up in the Address Queues. The Address Queues store up these indicators from each core interface, so that the GOIP can access it whenever it comes around. The Port Interface steps through each of the four queues and appropriately generates a "request", one STS frame row ahead on TXOHADDR, to which the external device would react. The external device's reaction (on TXOHDAT and TXOHVALID) is interpreted by the Port Interface. The external device's information is appropriately stored in the Data Tables, from which the Core Interface, on the next row would access to generate the overhead operation signals (OHDAT, OHMASK, and OHOPCODE). These overhead operation signals are later used by each of the cores to perform the overhead replacement, replacement with mask, or XOR function.

4.3 Core Interface

The Core Interface decodes the timing information from the individual STS-12 cores. In the four STS-12 core case, the common timing information comes from the first core (core #1). The common timing information consists of the column number, row

number, STS-3 number, and STS-1 number. A set of timing information unique to each core comes directly from each of the 4 cores. The core-unique information consists of the J1 indicator and the SPE indicator.

Using the timing information, the Core Interfaces generate control signals for the Overhead Address Queues and the Overhead Byte Tables. As well, the Core Interfaces output the corresponding overhead byte (OHDAT), mask (OHMASK), and operation code (OHOPCODE) to each of the cores downstream where overhead insertion operations are done.

There are 4 overhead insertion operations that are encoded on the OHOPCODE signal: data byte replacement without mask, data byte replacement with mask, XOR function, and invalid or idle. Table 1 describes the operation codes.

Table 1 Overhead Insertion Operations

Operation	Code	Comments
Data Replacement without Mask	00	OHMASK is invalid and should be ignored.
Data Replacement with Mask	10	OHDATA and OHMASK are both **valid.**
XOR Function	01	The XOR byte is contained in OHDAT. OHMASK is invalid.
Invalid / Idle	11	OHDATA and OHMASK are both invalid. Both should be ignored.

All of these operations are done downstream in each core. Each core may or may not execute the operation requested depending on its configurations.

The relationship between the timing information and the overhead operational output signals (OHDAT, OHMASK, and OHOPCODE) is described. The operational output signals are outputted one cycle after receiving the timing information. Figure 12 illustrates this timing relationship between the timing information and the overhead operational output signals.

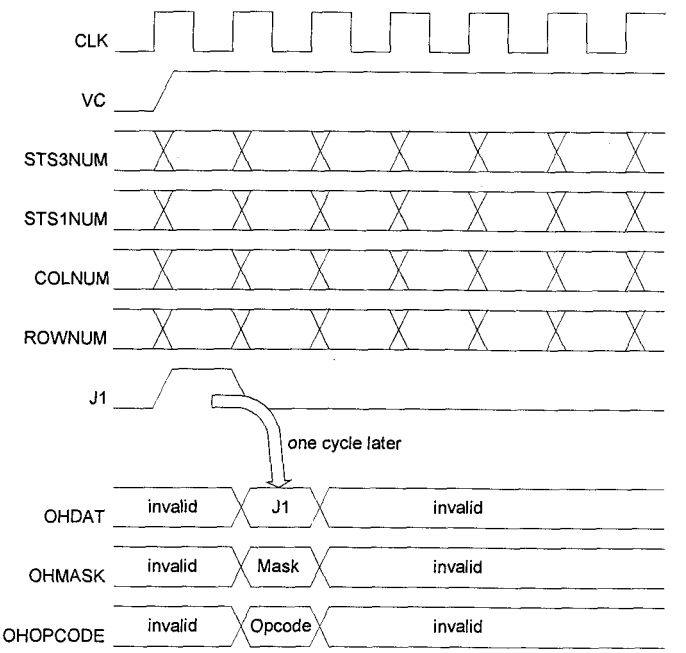


Figure 12 Core Interface Timing Diagram

4.4 Overhead Address Queues

There are 4 Overhead Address Queues, one for each STS-12 core. Each Overhead Address Queue stores up to 1 SOH start-of-row indicator and 12 POH byte

identifiers. These indicators and identifiers are stored in the order of their output from the 4 STS-12 cores.

Queues are necessary because the outputting order of the overhead bytes needs to be maintained and temporarily stored to facilitate the timely re-filling of the Overhead Byte Tables (used in the overhead byte generation data path). Only one SOH start-of-row indicator is stored per queue because all of the SOH bytes come consecutively within a row and they all come simultaneously from the four data path cores. Twelve storage bytes per queue are needed by the POH identifiers because there may be up to 4 STS-12 cores x 3 STS-1 POH = 12 POH bytes per row of a STS-48 frame.

The POH byte identifier consists of the STS-3 number, STS-1 number, overhead row number—which is very similar to the overhead byte address; hence the name "Overhead Address Queues". An "All 1s" in the "non-Row Number" fields of the identifier indicates the SOH start-of-row indicator. Table 2 illustrates the functional fields in the "byte identifier" (BYTEID).

Table 2 Address Queue BYTEID Fields

Bit	7	6	5	4	3	2	1	0
Field	STS-3 (1-4)		STS-1 (1-3)		Row Number (1-9)			

Each queue is minimally 13 words in length (1 for SOH and 12 for POH). The diagram below illustrates the 13-word queue structure.

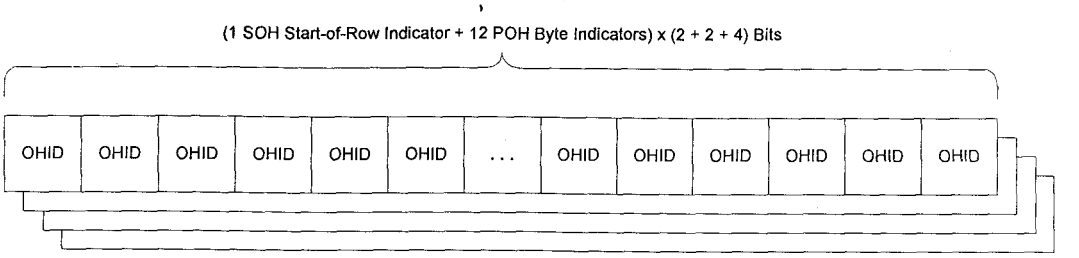


Figure 13 Overhead Address Queuing Structures

The write port and the read port of each queue are controlled by the Queue Controllers (which are group together with the Overhead Address Queues to form the "major" sub-block). There are 4 Queue Controllers, one per queue. Each Queue Controller tracks the status of each queue and forwards a queue empty or invalid (AQUEUE_RD_INVALID) status to the Port Interface.

4.5 Overhead Byte Tables

The Overhead Byte Tables store an entire row's worth of Section and Path Overhead bytes, in advance of the time needed. The table allows random access to each overhead byte within a row. The random access is needed because the order in which the Path Overhead bytes are outputted can shift in time making any order difficult to predict.

There is a total of 4 Overhead Byte Tables (one for each STS-12). The table is structured such that one dimension represents the link number (STS1 x STS3 = 3 x 4 = 12) and the other dimension, the Section and Path columns (3 + 1 = 4). The 2 dimensions form a memory map of 48 words in length. This structure is implemented

using maps of 64 word lengths for ease of implementation. These maps are shown in the Figure 14. The size of the word is 18 bits (8 for the byte, 8 for the mask, and 2 for the OPCODE). These word fields are shown in Table 3.

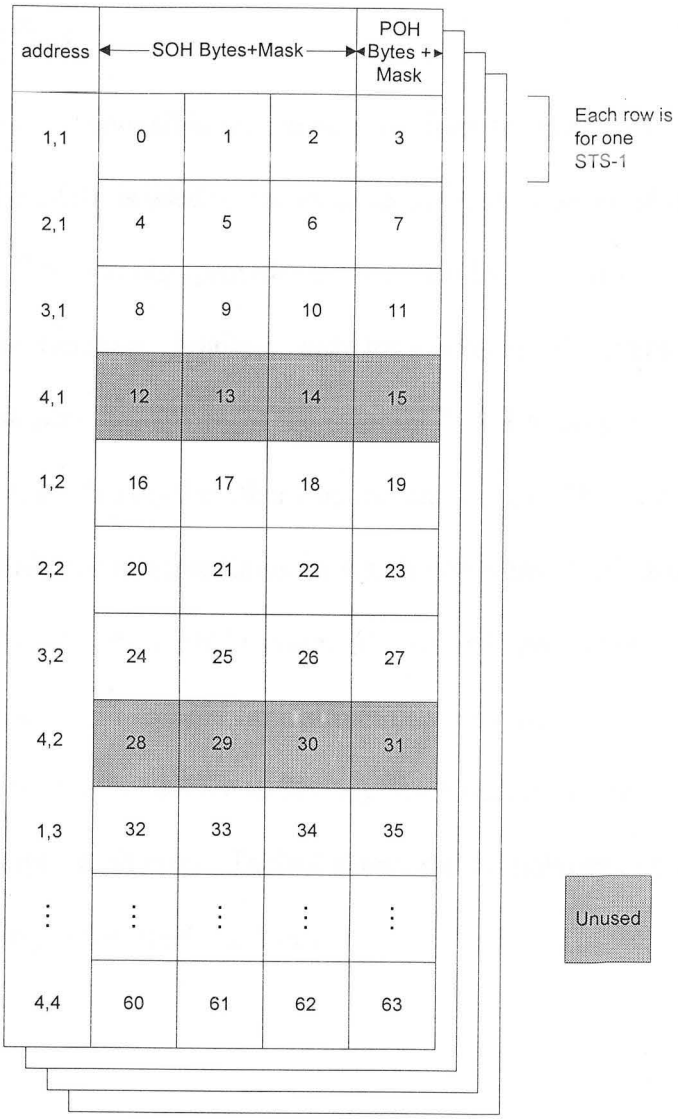


Figure 14 Overhead Byte Table Structure

Table 3 Data Table Word Fields

Bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OHOPCODE[1:0]		OHMASK[7:0]								OHDAT[7:0]							

4.6 Port Interface

The Port Interface is responsible for transactions from the Generic Overhead Insertion Port (GOIP). The GOIP is used by the external device to insert/replace every SOH, LOH, and POH. The port also provides 2 additional byte functions: a replacement mask and an XOR function. Available only for a subset of the overhead bytes, the replacement mask provides a way for the external device to only replace a fraction of the overhead byte, and leaves the other fraction untouched. This feature is needed for overhead bytes with two or more fields, in which one is inserted automatically by the ASIC and the other by the external device. The second special feature is the XOR function and is also only available for a subset of the overhead bytes. This XOR function allows the external device to flip any combination of bits in an overhead byte to cause, for example, a bit error. Table 4 shows the availability of the 2 special functions according to the overhead bytes.

Table 4 Overhead Byte Special Function Availability

	Mask	XOR	NOP
A1		X	
A2		X	
J0/Z0			X
J1 (POH)			X
B1		X	
E1			X
F1			X
B3 (POH)		X	
D1			X
D2			X
D3			X
C2 (POH)			X
H1	X	X	
H2	X	X	
H3			X
G1 (POH)	X		
B2		X	
K1	X		
K2	X		
F2 (POH)			X
D4			X
D5			X
D6			X
H4 (POH)	X	X	
D7			X
D8			X
D9			X
Z3/F3 (POH)	X		
D10			X
D11			X
D12			X
Z4/K3 (POH)	X		
S1/Z1	X		
M0/M1/Z2	X		
E2			X
Z5/N1 (POH)	X		

The XOR and replacement operations are interpreted by this block and passed to each of the core, where the actual operation takes place.

The Port Interface consists of a TXOHADDR generating component (outgoing path) and a TXOHDAT path (incoming path) component.

4.6.1 Outgoing (TXOHADDR generating) Path Component

The outgoing path component cycles through all 4 Overhead Address Queues in a round-robin fashion and processes the queues accordingly. The controller does not process any start-of-row indicator until this indicator is in all 4 Overhead Address Queues. With the BYTEIDs read from the 4 queues, the controller generates the transmit overhead address (TXOHADDR), along with the overhead clock (TXOHCLK). Figure 15 illustrates the state flow diagram for this round-robin process.

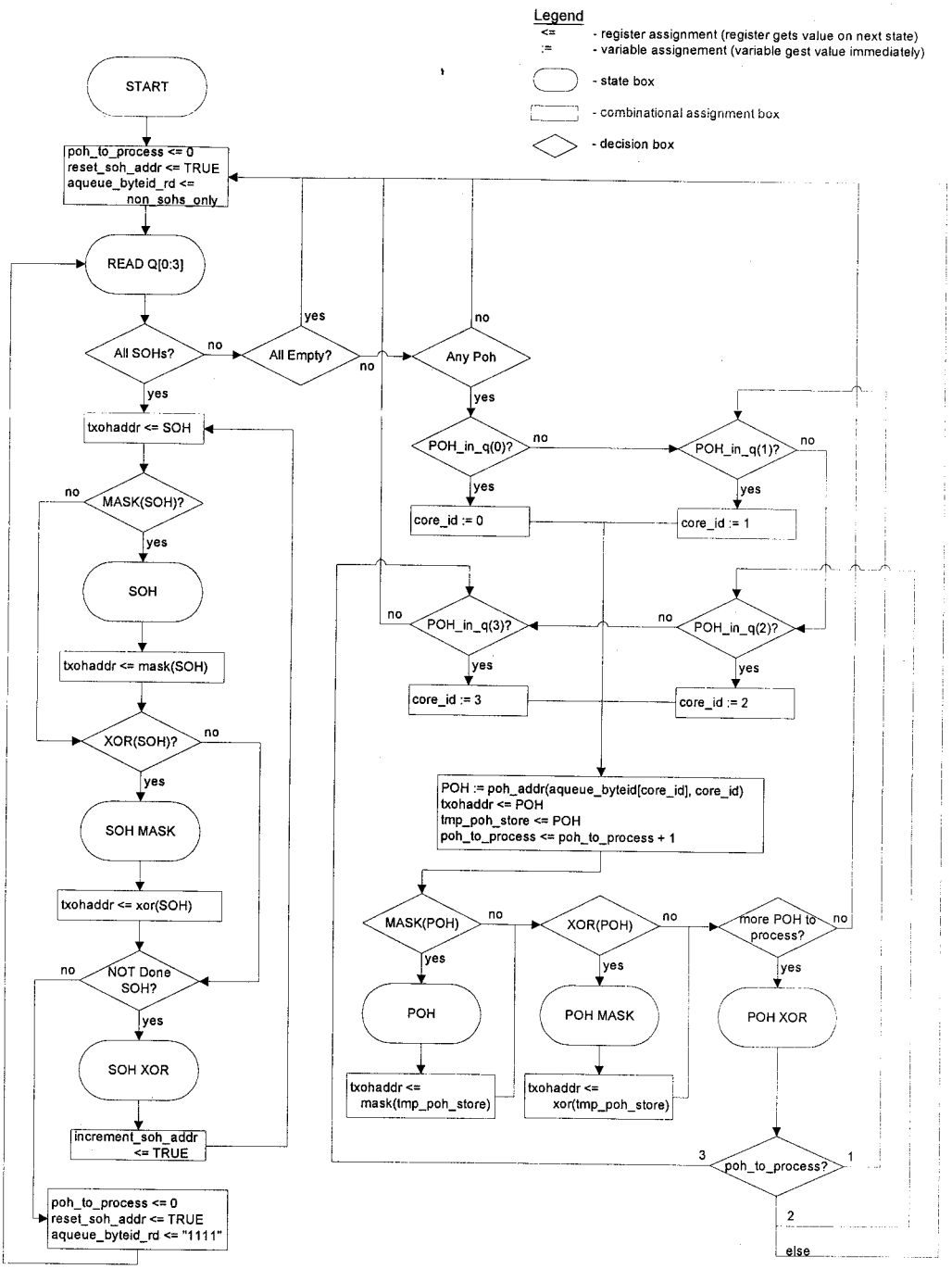


Figure 15 Port Interface State Flow Diagram

The TXOHADDR signal is actually a request for a byte from the external device.

The overhead address or TXOHADDR can be used to request for a data replacement

byte, mask replacement byte, and XOR byte. The order in which the requests come out is: "data request", "mask request", and then "XOR request". (See Figure 20 for some timing examples.) And so, encoded in TXOHADDR are the STS-1, STS-3, Core/Line number, row number and column number; this information permits the external device to access all overhead bytes in every STS-1 and SPE. Figure 16 shows the structure of the transmit overhead address, and Figure 17 shows how the mask and XOR functions are mapped in TXOHADDR.

TXOHADDR[11:6]

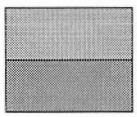
Bit	11	10	9	8	7	6
OC-48 Mode	STS-3 # (1-16)				STS-1 # (1-3)	
OC-12 Mode	Line # (1-4)		STS-3 # (1-4)		STS-1 # (1-3)	

TXOHADDR[5:0]

Bit	5	4	3	2	1	0
OH Byte	OH Row[3:0] (1-16)				OH Col[1:0] (1-3)	

Figure 16 Transmit Overhead Address Map

	col	00	01	10	11
row		1	2	3	4
0000	1	A1	A2	J0/Z0	J1
0001	2	B1	E1	F1	B3
0010	3	D1	D2	D3	C2
0011	4	H1	H2	H3	G1
0100	5	B2	K1	K2	F2
0101	6	D4	D5	D6	H4
0110	7	D7	D8	D9	Z3/F3
0111	8	D10	D11	D12	Z4/K3
1000	9	S1/Z1	M0/M1/Z2	E2	Z5/N1
1001	10	A1	A2		H4
1010	11	B1			B3
1011	12	H1	H2		H4
1100	13	B2			Z3/F3
1101	14	H1	H2		G1
1110	15		K1	K2	Z4/K3
1111	16	S1/Z1	M0/M1/Z2		Z5/N1



XOR

Mask

Figure 17 Overhead Byte and OPCODE Map

4.6.2 Incoming (TXOHDAT storing) Path Component

The incoming data path component directs the incoming transmit overhead data (TXOHDAT) into the one of the 4 Overhead Byte Tables. TXOHDAT is expected to be valid one cycle after TXOHADDR is outputted. Figure 18 shows this timing relationship.

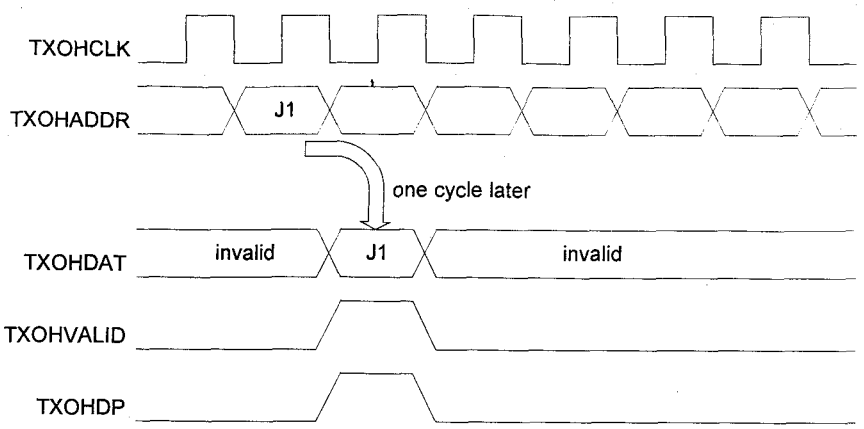


Figure 18 Generic Overhead Insertion Port Timing Diagram

Because of the way the GOIP is set up, the external device can generate valid signals to perform more than one operation. But, only one operation can be performed on each overhead byte for each time TXOHADDR is requested. A priority scheme is used and the priority is:

1) Replacement with mask.
2) Replacement without mask.
3) XOR operation.

Figure 19 is the state machine used to decode the incoming TXOHDAT and TXOHVALID signals, and to direct the TXOHDAT signals to the Data Tables.

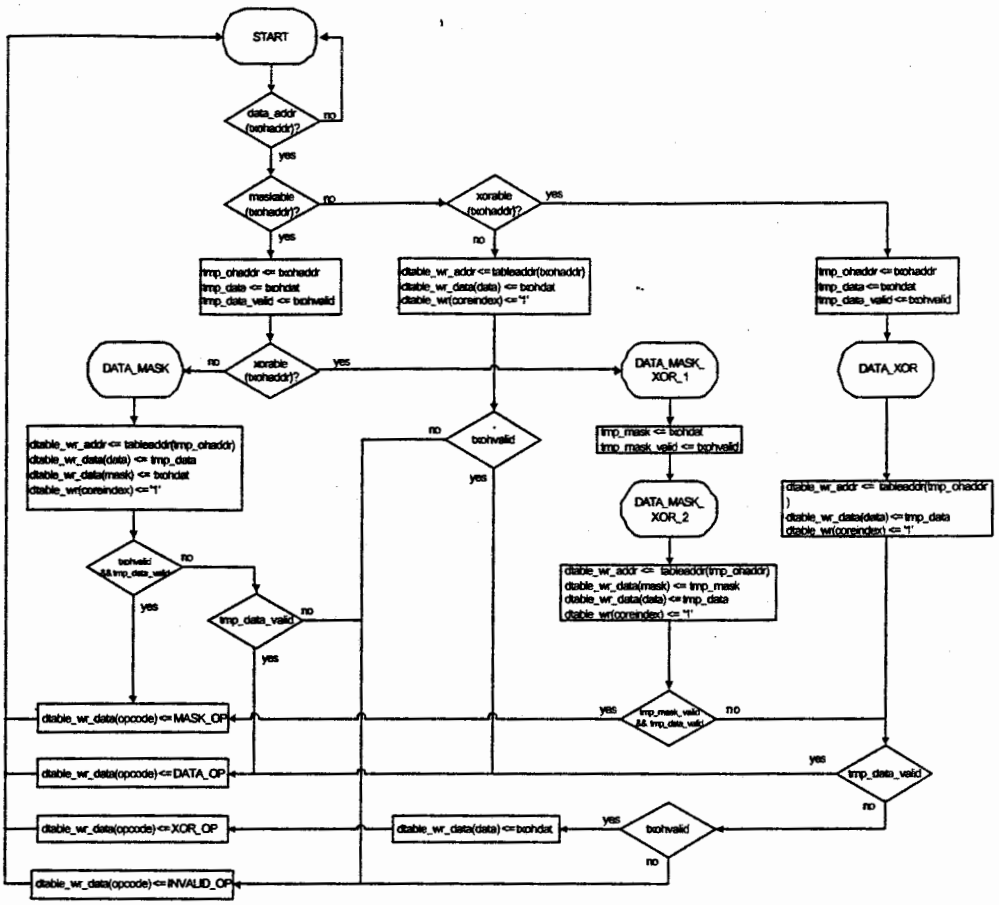


Figure 19 Incoming path TXOHDAT State Machine

The following are rules that result from this priority scheme:

a) "Replacement with a mask" is decoded when TXOHVALID signal is HIGH on both the DATA and MASK requests (of TXOHADDR), regardless of TXOHVALID being HIGH on the XOR request.

b) "Replacement without a mask" is decoded when the TXOHVALID is HIGH on the DATA but LOW on the MASK request. TXOHVALID is ignored on the XOR request.

c) "XOR operation" is decoded only when TXOHVALID is LOW on both the DATA and MASK requests, but HIGH on the XOR request.

Figure 20 has some functional timing examples to illustrate these three rules.

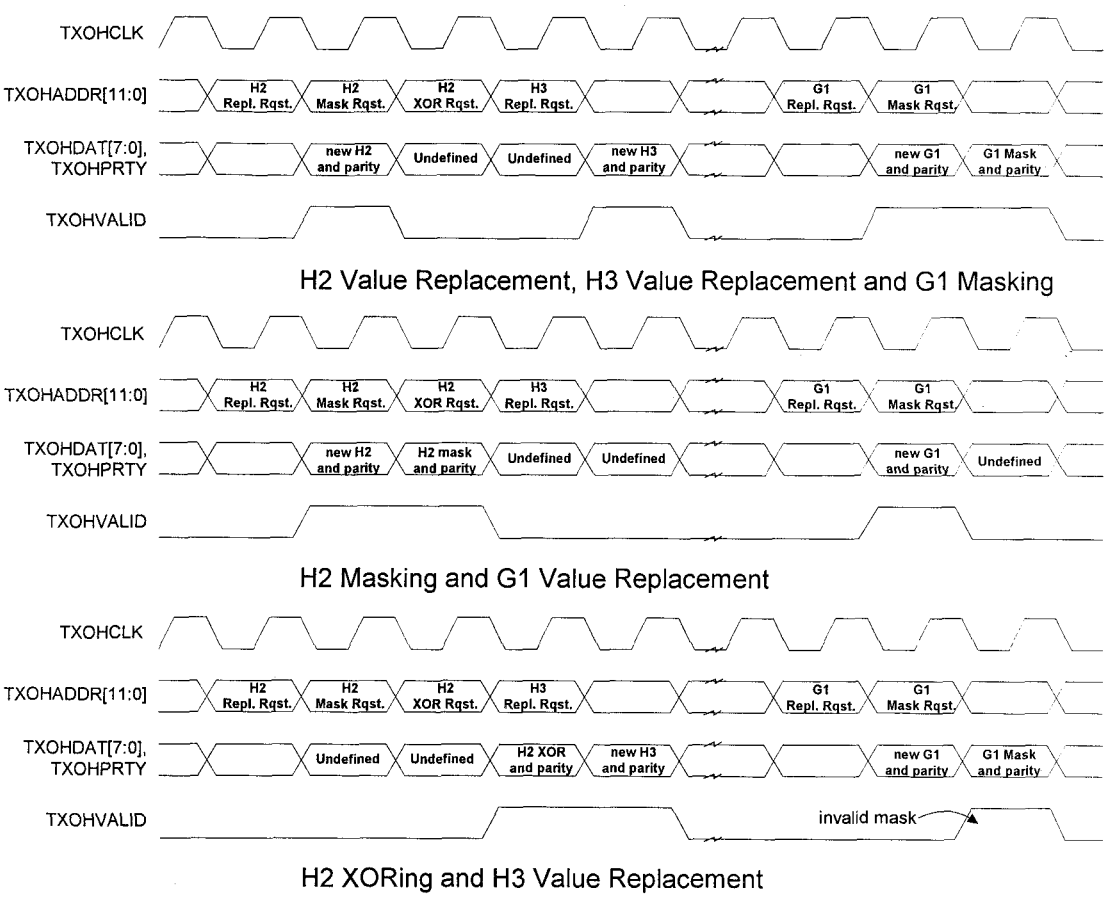


Figure 20 Functional Timing Examples of the GOIP

4.7 Connector Description

This part of the report tabulates all the connectors for the TXOHINS mechanism, providing detailed descriptions and signal types. The connectors are divided into 3

categories: core connectors (to and from the data path cores), overhead insertion port connectors (to and from the GOIP), and configuration connectors (to and from central control in ASIC).

4.7.1 Core Connectors

Table 5 Core Connectors

Connector Name	Type	Signal Type	Connector Description
CLK	I	Clock	**77.76 MHz Clock** – Master Clock.
RSTB	I	Async	**Asynchronous Reset** – Master Asynchronous Reset. Active Low.
COL[6:0]	I	Sync	**Column Number** – The current STS-1 column number. This signal is 7 bits wide: "0000000" = Column #1 and "1011001" = Column #90.
ROW[3:0]	I	Sync	**Row Number** – The current STS-1 row number. This signal is 4 bits wide: "0000" = Row #1 and "1000" = Row #9.
STS1[1:0]	I	Sync	**STS-1 Number** – The current STS-1 number. This signal is 2 bits wide: "00" = #1, "01" = #2, and "10" = #3.
STS3[1:0]	I	Sync	**STS-3 Number** – The current STS-3 number. This signal is 2 bits wide: "00" = #1, "01" = #2, "10" = #3, and "11" = #4.
J1_N	I	Sync	**J1 Byte Indicator from Core #N** – This signal is HIGH when the current byte in Core #N is a J1 byte. Together with the STS-1 and STS-3 numbers, this signal indicates the start of a SPE. **N** = 1, 2, 3 or 4.
SPE_N	I	Sync	**SPE Indicator from Core #N** – This signal is HIGH when the current byte in Core #N is a byte in a SPE. Together with the STS-1, STS-3, and J1_N signals, this signal is used to keep track of the Path Overhead column position. **N** = 1, 2, 3, or 4.

Connector Name	Type	Signal Type	Connector Description
OHDAT_N[7:0]	O	Sync	**Overhead Data to Core #N** – An 8-bit wide signal containing the replacement byte or XOR bits. OHDAT is outputted one CLK cycle after the overhead byte is indicated by COL, ROW, STS1, STS3, J1_N, and SPE_N. The operation (data replacement or XOR) is determined by OHOPCODE_N. The data replacement with "mask" needs OHMASK_N as well as OHDAT_N.
OHMASK_N[7:0]	O	Sync	**Overhead Mask Data to Core #N** – An 8-bit wide signal containing the mask data to be operated onto the current overhead byte. OHMASK is outputted one CLK cycle after the overhead byte is indicated by COL, ROW, STS1, STS3, J1_N, and SPE_N. This signal is only valid on a mask-replacement operation.
OHOPCODE_N[1:0]	O	Sync	**Overhead Operation Code to Core #N** – A 2-bit wide signal containing the overhead operation code. "00" – Data replacement (no mask) "01" – XOR operation "10" – Data replacement with mask "11" – NOP or Invalid This signal is also outputted one CLK cycle after the overhead byte indicated by COL, ROW, STS1, STS3, J1_N, & SPE_N.

4.7.2 Overhead Insertion Port Connectors

Table 6 Overhead Insertion Port Connectors

Connector Name	Type	Signal Type	Connector Description
TXOHCLK	O	Sync	**Transmit Overhead Insertion Clock** – This clock is used by the external device to interface to the Overhead Insertion Port (GOIP).

Connector Name	Type	Signal Type	Connector Description
TXOHADDR[11:0]	O	Sync	**Transmit Overhead Insertion Address** – This 12-bit signal contains the address of the next overhead byte to be placed on TXOHDAT. TXOHADDR is outputted on the rising edge of CLK when TXOHCLK is HIGH (or moments before the edge of TXOHCLK falls). See Figure 16to see how the overhead bytes and operations are mapped to TXOHADDR.
TXOHDAT[7:0]	I	Sync	**Transmit Overhead Insertion Data** – This signal contains the data, as requested by TXOHADDR one TXOHCLK cycle earlier. TXOHDAT is captured on the rising edge of CLK when TXOHCLK is LOW (or moments before the edge of TXOHCLK rises) and is only used if TXOHVALID is HIGH.
TXOHVALID	I	Sync	**Transmit Overhead Insertion Data Valid** – This signal indicates that the TXOHDAT is valid and therefore, TXOHDAT should be used the overhead operations. TXOHVALID is captured on the rising edge of CLK when TXOHCLK is LOW (or moments before the edge of TXOHCLK rises).
TXOHDP	I	Sync	**Transmit Overhead Insertion Data Parity** – This signal indicates the parity of TXOHDAT. TXOHDP is only valid when TXOHVALID is HIGH. TXOHVALID is also captured on the rising edge of CLK when TXOHCLK is LOW (or moments before the edge of TXOHCLK rises).

4.8 Configuration Connectors

Table 7 Configuration Connectors

Connector Name	Type	Signal Type	Connector Description
ODD_EVEN_B	I	Sync	**Parity Type** – This signal indicates whether odd (1) or even (0) parity is used.
PERROR	O	Sync	**Parity Error** – For each parity error (even parity) detected at TXOHDAT and TXOHDP, this signal pulses high for 1 CLK cycle. This signal transition on the rising edge of CLK.

Async = Asynchronous input.

Reg = Register input. Should be synchronize to CLK.

Sync = Input signal generated from rising edge flip-flop clocked by a clock.

Rise = Generated by rising edge flip-flop

Fall = Generated by falling edge flip-flop

5 Conclusion

The report has introduced several concepts in the SONET standard for the purpose of providing a background to understand the need for overhead insertion. The TXOHINS mechanism is a novel method invented at West Bay for the insertion of overhead information. The implementation was presented in the body of this report.

The TXOHINS mechanism met the most important goal, which was to reduce the number of inputs and outputs. The ASIC products that applied the TXOHINS mechanism reduced its I/O count from 432 I/Os (in the synchronization-clock-data scheme) to 24 I/Os (in TXOHINS mechanism: 12 for TXOHADDR, 8 for TXOHDAT, 1 for TXOHVALID, and 1 for TXOHDP). In conclusion, the TXOHINS mechanism was essential for SONET physical interface devices. Currently, the entire line of West Bay products uses this novel method.

LIST OF REFERENCES

1. G. 707, "Network Node Interface for the Synchronous Digital Hierarchy (SDH)", ITU-T Rec., October 2000.
2. GR-253-CORE, "Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria", Telcordia Technologies/Bellcore, Issue 3, Sept. 2000.
3. J. Bellamy, "Digital Telephony 2nd Ed.", John Wiley & Sons, Inc., 1991.
4. M. Sexton and A. Reid, "Broadband Networking: ATM, SDH, and SONET", Artech House, 1997.
5. SDH/SONET Superscalabe STM-16/STS-48 Framing Processor, Advance Engineering Data Sheet, Issue 0.5.3, West Bay Semiconductor Inc., September 2000.
6. T1X1.5/99-064R2, Revised Draft ANSI T1.105 SONET Base Standard, ANSI T1X1.5, July 1999.
7. T1X1.5/99-066R1, Revised Draft T1.105.02 SONET Payload Mappings Standard, ANSI T1X1.5, August 1999.