UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot • your knowledge partner

Implementation of an SDH simulator using SDR

by

A.D.Brandt

Thesis presented at the University of Stellenbosch in partial fulfilment of the requirements for the degree of

Masters of Science in Electronic Engineering

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Dr G-J. van Rooyen

Co supervisor: Prof JG Lourens

April 2006

DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis in my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

______________________________ ______________________

SIGNATURE DATE

SYNOPSIS

A *Synchronous Digital Hierarchy* (SDH) point-to-point bi-directional link was implemented at a base *Synchronous Transfer Mode level 1* (STM 1) signal rate. The full STM-1 multiplexer was implemented and the functional code developed to *Virtual Container level 4* (VC4) level. The implementation was realized using a *Software Defined Radio* (SDR) architecture that managed and linked the SDH atomic units into a STM-1 SDH multiplexing structure. These atomic units have been well defined in recommendation G.707 [1].

The functional description of each unit was based on the G.783 [8] recommendation which specifies a library of basic building blocks and set of rules by which these atomic functions should be combined into various functional layers. These layers interconnect to ultimately form a bi-directional path in the SDH network.

A *SDH Management Sub network* (SMS) was implemented using a graphical user interface to perform a monitoring function for the bi-directional link.

OPSOMMING

‘n *Sinkroniese Digitale Hiërargie* (SDH) punt-tot-punt tweerigting-kommunikasieskakel is geïmplementeer teen ’n basiese STM 1 kapasiteit. Die implementering hiervan is gedoen deur gebruik te maak van ’n sagteware- gedefinieerde radio (SDR) argitektuur wat die SDH subeenhede bestuur en verbind om ’n STM-1 SDH multiplekseerderstruktuur te vorm. Hierdie subeenhede is gedefinieer in aanbeveling G.707 [1].

Die funksionele beskrywing van elke eenheid is gebaseer op die G.783 [8] aanbeveling, wat ’n biblioteek van basiese boueenhede beskryf en reëls spesifiseer waarvolgens hierdie eenhede gekombineer kan word om die verskeie funksionele vlakke te definieer. Hierdie vlakke word onderlangs verbind om uiteindelik ’n eenrigting-skakel in die SDH-netwerk te vorm.

’n *SDH Bestuur-Subnetwerk* (SMS) is geïmplementeer deur gebruik te maak van ’n grafiese gebruikerskoppelvlak om ’n moniteringsfunksie te verrig oor die tweerigting-kommunikasieskakel.

ACKNOWLEDGEMENTS

- To the coffee addicted DSP crew. Thanks for friendship, laughter and the right mix of light heartedness that made everyday pleasant.
- To those who attempted proofreading my thesis, thanks.
- A special thank-you to my supervisor Prof JG Lourens, for accepting me onto the program and acting as my supervisor while I was doing my course work.
- A special thank-you to my supervisor Dr G-J van Rooyen, for keeping me on track, helping me to stay motivated and for helping me out of some very dark corners. Your efforts and time are well appreciated.
- To friends and family who fervently kept me out of work. How can I ever thank you enough.
- And for the silent prayers I said at desperate times. I thank my heavenly Father.
- Three cheers for not giving up.

TABLE OF CONTENTS

LIST OF FIGURES:

LIST OF TABLES:

GLOSSARY OF ACRONYMS

ADM	Add Drop Multiplexer
AIS	Alarm Indication Signal
AU	Administrative Unit
AUG	Administrative Unit Group
BBE	Background Block Errors
C	Container
C-12	Container one type two
CORBA	Common Object Request Broker Architecture
E-1	European level 1
ES	Equipment Site
ES	Errored Second
GNE	Gateway Network Element
HP	Higher Order Path
ITU	International Telecommunications Union
ITU-T	International Telecommunications Union - Telecomms
LOF	Loss Of Frame
LOP	Lower Order Path
LOPOH	Lower Order Path Overhead
LP	Lower Order Path
MS	Multiplex Section
MSOH	Multiplex Section Overhead
MST	Multiplex Section Termination.
NE	Network Element
OAM	Operations Administration and Management
OAM&P	Operations Administration Management and Provisioning
OS	Operations Management
PDH	Plesiosynchonous Digital Hierarchy
POH	Path Overhead
PTE	Path Terminating Equipment
RDI	Remote Defect Indicator
REI	Remote Error Indication
RFI	Remote Fault Indicator
RS	Regenerator Section
RSOH	Regenerator Section Overhead
RST	Regenerator Section Termination
SDH	Synchronous Digital Hierarchy
SDR	Software Defined Radios
SES	Severely Errored Second
SMN	SDH Management Network
SMS	SDH Management System
SOH	Section Overhead
SONET	Synchronous Optical Network
SPE	Synchronous Payload Envelope
STM	Synchronous Transfer Mode
STS	Synchronous Transport Signal
T-1	T carrier level 1
TCA	Threshold Crossing Alarm
TM	Terminal Multiplexer
TMN	Telecommunications Management Network

GLOSSARY OF ACRONYMS CONTINUE:

TU	Tributary Unit
TUG	Tributary Unit Group
VC-12	Virtual Container level one type two
VC-3	Virtual Container level three
VT	Virtual Tributary

CHAPTER 1
INTRODUCTION

1.1.Motivation for this work

The S*ynchronous Digital Hierarchy* (SDH) forms a critical part of the telecommunications backbone, and effective design, reliability analysis and training is essential to managing SDH networks effectively. This thesis details the design of an SDH implementation using a *Software-Defined Radio* (SDR) platform. The design has been implemented with some of the core SDH functionality in mind and hence does not cover all aspects of the wide range of SDH adaptation and trail termination functions usually associated with larger real world SDH networks. A single STM-1 SDH multiplexer was implemented and the *Container One type Two* (C12) was used as a mapping opportunity for non-SDH payloads to propagate along the SDH path.

The main purpose of such an implementation is investigative; additionally, a software implementation can be used for operator training, fault simulation and even as a design tool. Although a software-defined radio platform is used to realise the SDH functionality, it should be noted that this is motivated by efficiency and flexibility demands, although the system is not a “radio”. A final objective of the research is to demonstrate the use of SDR techniques for a wider range of data-streaming applications.

This software SDH system would be useful as a training and academic tool if built using the SDH *International Telecommunications Union* (ITU) standards. The SDH system makes use of the SDR architecture described in [5]. The SDR architecture will be evaluated for its effectiveness in managing and transporting large data payloads using a network of smaller atomic functional units strung together to form a bi-directional STM-1 SDH link.

1.2.Background

SDH is a digital transport structure that appropriately manages the transported payloads while transporting them through transmission mediums. SDH has been standardised by the *International Telecommunications Union* ITU since 1988 and is based on SONET developed in BELL laboratories in the United States of America. SDH takes care of layer 1 (the physical layer) and layer 2 (the data-link layer). European telecommunications operators have used SDH since the early 90’s. Before the implementation of SDH, a non-standardised *Plesiochronous Digital Hierarchy* (PDH) was used in European and North American networks and is still widely implemented.

After the digitisation of voice into a base unit 64Kb/s voice channel by generating 8 bits 8000 times per second, the PDH carrier systems grew as the demand for telephone circuits increased. The *T carrier Level 1* (T-1) trunking networks in North America and the *European Level 1* (E-1) in Europe was the PDH technologies used by telephone companies to address the problem brought on by unprecedented growth in telephone demand.

PDH signals are mapped into STM signals on a SDH network and *n* STM signals form an STM-*n*. The standardised SDH technology takes over from PDH, and improves on the lack of standardisation and management of PDH. Some of the advantages of employing such a SDH network are:

- Simplified multiplexing and demultiplexing;
- Direct access to low-rate tributaries;
- Enhanced OAM capabilities;
- Easy transition to higher bit rates [22].

SDR offers very attractive features to implement a software version of the SDH and was used as the base architecture for this implementation. The SDR system used was developed by Stellenbosch University and is undergoing continuous further development in terms of architecture while at the same time growing its library base from the work of individual contributors.

The architecture takes care of signal handling between and within layers. The converter blocks, which will be discussed in Chapter 4, offers designers the opportunity to implement functional code of choice, without regard for the management and execution of these blocks, as these tasks are taken care of by the SDR architecture. The converter blocks allowed for the development of the atomic SDH functional units based on the ITU-T recommendations, while the SDR architecture offered the interlinking and self-management features for the stringing together of these units into a SDH multiplexer.

1.3.Objectives of this study

This work has two objectives:

- Primarily it is to develop a SDH simulator, incorporating some of the functional features of SDH. SDH is a vast group of standards thus a selection of features for implementation as opposed to implementing all SDH features had to be made. This choice was based on the mapping of 63 x 2Mb/s E-1 PDH signals into 63 x C-12 SDH containers, to finally form an STM-1 signal. This is the full capacity to which an STM-1 can be mapped when only C-12 mapping is used. This choice reflects the South African and European implementation of the SDH structure. The functional implementation of the system was to VC4 level, whereas the full STM-1 MUX structure was implemented.

- The secondary objective of this study is that of evaluating the performance of the SDR architecture developed by Stellenbosch University, for a large-scale implementation such as SDH. The SDR system is well tested for applications having relatively small converter block arrangements and low data transfer levels. This SDH on SDR application exploits the data handling and management capabilities of the architecture.

1.4.Contributions

In accomplishing the objectives, a number of contributions were envisioned.

- A summary of the main recommendations in the SDH environment.
- The development of a software defined SDH simulator, thus allowing for easy expansion on the SDH defined atomic units. This will allow the simulator to expand to accommodate a wide range of SDH features.
- A product that could simulate an SDH environment for networking analysis.

1.5.Thesis Outline

This introduction forms the first chapter of this thesis. The remainder of the document is organised as follows:

Chapter 2: Background theory and Literature Review. In Chapter 2 various aspects of SDH are considered. A discussion on network segmentation is given. This is followed by a discussion of the overhead found in the section overhead and path overhead.

Chapter 3: SDH management. Chapter 3 concentrates on some of the key points in SDH management. The chapter is intended to serve as an introduction, and focuses mainly on the sections of SDH management that was applied in this application. In particular fault management and performance management are discussed.

Chapter 4: Software Defined Radios. Chapter 4 details the SDR architecture used in this implementation. The requirements for an SDR are briefly discussed, followed by a look at each of the layers within the developed SDR architecture.

Chapter 5: SDH network using SDR. Chapter 5 represents the implementation of the SDH on the SDR architecture. Detail is given of how these two technologies have been combined, and how the functional units of SDH have been implemented using the converter block and subcontroller units of SDR.

Chapter 6: Analysis. In Chapter 6 some functional tests were developed, and the results presented. These results cover SDH testing and SDR analysis.

Chapter 7: Conclusion. Chapter 7 is the concluding chapter, which summarises the achievements of this work and suggests further future work.

Appendixes: A number of appendixes are given detailing work described in some of the preceding chapters.

Reference: This section describes the references that were used for the topic research.

CHAPTER 2

BACKGROUND THEORY AND LITERATURE REVIEW

2.1.Introduction

SDH and SONET technology is currently the dominant choice for metropolitan-area networks as well as for accessing wavelength division multiplexing networks in wide-area networks [19]. The technology is generally implemented using double counter rotating rings, and within each ring frames are forwarded that contain a *Section Overhead* (SOH) that is used primarily for network management and a payload section. The frame duration of each frame is always 125µs, and is immediately followed by another frame. The payload of the frame is organised into one or more virtual containers (VC's), which in turn is organised into the path overhead and the payload section.

Three main functional elements make up an SDH network. The first functional element being the add or drop, responsible for mapping the payload into a *Virtual Container* (VC) at the source and extracting the payload from the VC at the destination. The second is a multiplexer responsible for multiplexing several VC's from several frames into a higher data rate. The third being a digital cross connect, which transfers a VC from one ring to another ring in its path to its destination.

The transport of the VC's through the SDH/SONET network is not self-routing but relies on fixed configured paths.

The following sections of this chapter will give a deeper insight into the workings of SDH. A breakdown of the overhead bytes in terms of structure and function will be given as well as a breakdown of the multiplexing structure.

2.2.SYNCHRONOUS DIGITAL HIERARCHY

Digital Hierarchy Level	Rates (Mbps)				SDH Designated Signal
	North American	Japan	Europe	ISDN	
0	.064	.064	.064	-	-
1	1.544	1.544	-	H11	C-11
	-	-	2.048	H12	C-12
2	6.321	6.312	-	-	C-2
	-	-	8.448	-	-
3	-	32.064	-	-	-
	-	-	34.368	H31	C-3
	44.736	-	-	H32	C-3
4	-	95.728	-	-	-
	-	-	139.264	H4	C-4
	274.176	-	-	-	-

Table 2.1: PDH and SDH container

Table 2.1 shows the three regional signal hierarchies used for global communications.

The North American digital network has 24 digitised voice channels, each with a capacity of 64kb/s. Japan uses a similar system to North America, also having 24 digitised voice channels. The European system uses 32 digitised voice channels, each with a capacity of 64 kb/s.

SDH defined several containers to accommodate all three regional standards. These containers allow the PDH signals to be mapped into SDH. The original goal of the Standards Committee was to establish an international specification for a common hierarchy among the European, North American and Japanese standards.

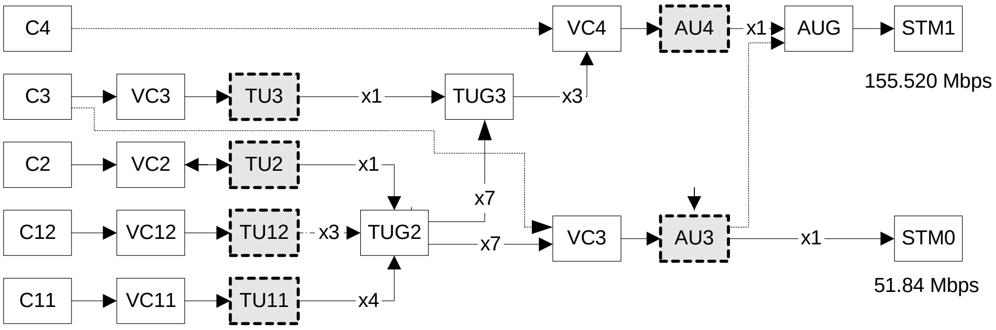


Figure 2.1: SDH Multiplexing Structure

The ITU adopted STM-1 as the first level of the SDH hierarchy at a rate of 155 520 kb/s and the zero level of the SDH hierarchy at a rate of 51 840 kb/s. Higher SDH bit rates can be obtained as integer multiples of the first level bit rate. SDH was designed to be universal in allowing the transport of a large variety of signals. In North America, the *Synchronous Optical Network* (SONET) specifications were based on the 51 840 kb/s *Synchronous Transport Signal level 1* (STS-1) signal.

The most important SDH specification used during this implementation is the ITU-T G.707 [1], "Network Node Interface for the Synchronous Digital Hierarchy (SDH)". It took the ITU four years to update the specifications; the new draft was approved in May 1996 and released in March 1997. Following this major G.707 revision, the ITU updated the G.783 [8] specification, "Characteristics of Synchronous Digital Hierarchy (SDH) Equipment Functional Block". The main goal of these new specifications is to define the SDH interfaces and equipment that will ease the use of SDH services in the access network.

Figure 2.1 shows the generally accepted multiplexing scheme. A plesiochronous 2-Mb/s tributary is mapped into a synchronous *Container* (C-12) with the addition of justification bits for frequency adaptation. After addition of *Path Overhead* (POH) information for path management the *Virtual Container* (VC-12) is obtained. The phase difference between the VC-12 and the *Tributary Unit* (TU) is indicated by a TU pointer. After addition of the pointer the TU-12 is obtained. In the first multiplexing stage three TU-12s are inserted into one *Tributary Unit Group* (TUG-2). The second multiplexing stage combines seven TUG-2s and VC-3 *Path Overhead* (POH) information in the VC-3. After addition of pointer information for phase alignment the *Administrative Unit* (AU-3) is obtained. The third multiplexing stage multiplexes three AU-3s to the *Administrative Unit Group* (AUG). Finally in the fourth multiplexing stage *N* AUGs are byte interleaved.

STM-N is obtained after addition of *Regenerator Section Overhead* (RSOH) for the management of repeater sections and *Multiplex Section Overhead* (MSOH) for the management of multiplex sections. Thus an STM-1 can transmit 3×7×3 = 63 2-Mbit/s tributaries or 3×34 Mb/s tributaries or one 140 Mb/s tributary.

2.3.Network Segmentation: SDH Layers

SDH adds no additional overhead at the higher levels of the multiplexing structure. The entire overhead needed is present even at the lowest level of the hierarchy. The overhead percentage in SDH is fixed, while overhead continually rises in other asynchronous multiplexing schemes. In SDH the ratio of payload to overhead remains constant at 3.4%, regardless of the SDH level [18]. The advantage of fixed overhead over rising overhead becomes more pronounced the higher the line rates.

As shown in figure 2.2, SDH segments the network into a *Regenerator Section* (RS), a *Multiplex Section* (MS), a *High Order Path* (HP) and a *Low Order Path* (LP). This allows for the probable location of an error or defect and provides a comprehensive view of the networks performance. The RS begins and ends with every element in the network, except for purely passive components like amplifiers.

A MS begins and ends with any element, such as an add-drop multiplexer or digital cross connect, that is capable of integrating multiple traffic sources. The HP begins at the origin of the data stream and ends at its destination. LP applies to lower-rate virtual tributaries, such as 2-Mbps circuits carried over SDH. SDH also has an *Add-Drop Multiplexer* (ADM), which never examines or processes the path overhead, with the exception of the optional tandem connection byte in the path overhead. RS are spaced at 40kms between the ADM spans. SDH also has a *Path Terminating Equipment* (PTE) device known as a *Terminal Multiplexer* (TM). There is no SDH beyond the TM in an SDH link.

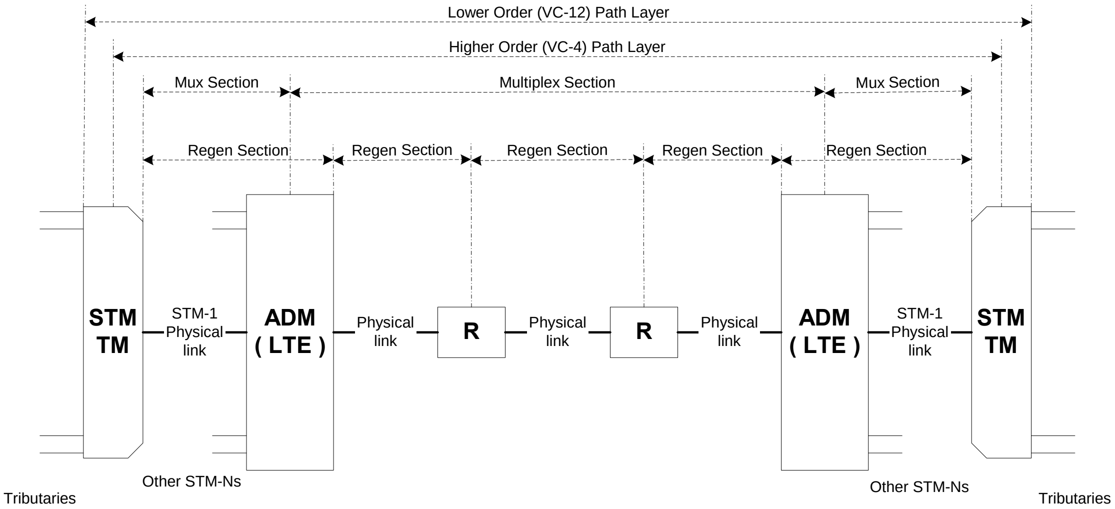


Figure 2.2: Segmentation of an STM1 SDH link

The SDH overhead is likewise divided into a Regenerator Section, a Multiplex Section, a High Order Path, and Low Order Path as shown in figure 2.2.

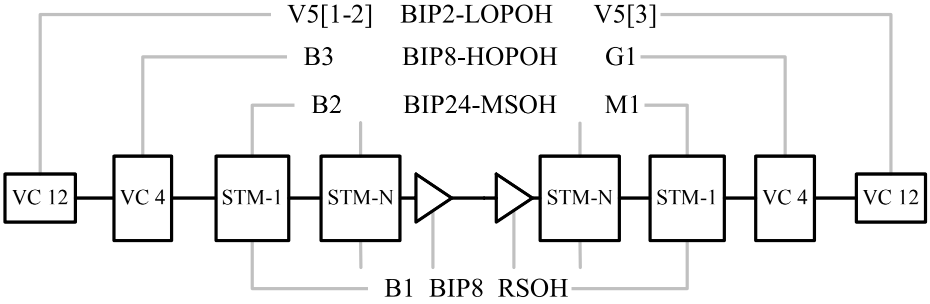


Figure 2.3: SDH overhead

The bytes shown in figure 2.3, shows the inherent error checking capabilities of SDH. *Bit Interleave Parity* (BIP) is used, where the BIP performs a routine even-parity check on the previous frame. These bytes and their application are discussed further in chapter 3 under SDH management.

The B1 and B2 byte form part of the *Section Overhead* (SOH) and are added or extracted at the regenerator or multiplexer section for block framing and performance monitoring.

The B1 byte containing the BIP-8 code is associated with the RS. The maximum number of errors that the B1 byte can detect is reduced with an increase in line rate. This is because the number of parity bits used for error monitoring of the regenerator section remains constant while there is an increase in block size as the line rate increases. The B1 byte is transported in the *Regenerator Section Overhead* (RSOH) with its purpose being that of monitoring the signal integrity of the sections between regenerators. A BIP-8 calculation using even parity is performed on the previous frame, and the result stored and transmitted in the B1 byte of the next frame. On the sink side a comparison is made between the generated BIP-8 value and the received BIP-8. Upon detecting a violation the originating equipment is alerted. This BIP-8 value is checked and recalculated at every regenerator.

The B2 byte containing the BIP-24 code is used for MS error monitoring. For the STM-N case, BIP-Nx24 is used. The maximum number of errors that the B2 can detect remains constant with an increase in line rate since the number of parity bits increases with an increase in block size. The BIP-24 even parity calculation is performed over the previous frame, with each frame 125µSec in duration. This calculated BIP-24 value is then stored in the B2 byte and transmitted in the next frame. As with the case for BIP-2 and BIP-1, the sink side then compares the calculated BIP-24 value to that of the received BIP-24 value. Any violation is reported back to the originating equipment using the M1 byte. The BIP-24 code, unlike the B1 BIP-8 code is checked only at the line termination where protection may occur. The M1 and B2 byte forms part of the *Multiplex Section Overhead* (MSOH).

The BIP-2 code in the V5 byte and the BIP-8 code in the B3 byte forms part of the *Path Overhead* (POH). The POH is used for end-to-end communications where the particular VC is terminated.

The V5 byte does a BIP-2 even parity calculation on the previous multiframe. The multiframe duration is 500µSec, and is made up of four frames with each frame being 125µSec in duration. This allows for end-to-end error monitoring at each of the VC12 lower order paths. There are 63 VC-12 lower order paths per STM-1. The V5 byte forms part of the *Lower Order Path Overhead* (LOPOH). The same BIP–2 calculations are performed on the sink side of the transmission path for each VC12. The calculated BIP-2 value is then compared with the received BIP-2 value and any violation is then reported back to the originating equipment by setting a *Remote Error Indication* (REI) bit in the V5 byte.

The B3 byte holds the BIP-8 code associated with the VC-4 path. An even parity calculation is performed on all the bits of the previous VC-4 and stored in the current B3 byte. As with the LOP, the BIP-8 values is recalculated at the terminating side and compared with the received BIP-8 value. Any violation is reported back to the originating equipment using the G1 byte. The B3 byte and the G1 byte forms part of the VC-4 Higher *Order Path Overhead* (HOPOH)

BER testing is ongoing and non-intrusive. The BIP helps in the isolation of the weaker links when end-to-end BER errors become high enough.

All three BIP checks have the same function, but the scope of operation within the STM frame differs.

		1	2	3	4	5	6	7	8	9		…	270
Regenerator Section Overhead (RSOH)	1	A1 Framing	A1 Framing	A1 Framing	A2 Framing	A2 Framing	A2 Framing	J0 Trace	National use	National use	STUFFING BYTES	J1 Pathtrace	PAY LOAD
	2	B1 BIP-8	Media dependent	Media dependent	E1 Orderwire	Media dependet	R Reserved	F1 user	National use	National use		B3 BIP-8	
	3	D1 Datacom	Media dependent	Media dependent	D2 Datacom	Media dependent	R Reserved	D3 Datacom	R Reserved	R Reserved		C2 Signallabel	
AU pointer bytes	4	H1	H1	H1	H2	H2	H2	H3	H3	H3		G1 Pathstatus	
Multiplex Section Overhead (MSOH)	5	B2 BIP-24	B2 BIP-24	B2 BIP-24	K1 APS	R Reserved	R Reserved	K2 APS	R Reserved	R Reserved		F2 User	
	6	D4 Datacom	R Reserved	R Reserved	D5 Datacom	R Reserved	R Reserved	D6 Datacom	R Reserved	R Reserved		H4 Multiframe	
	7	D7 Datacom	R Reserved	R Reserved	D8 Datacom	R Reserved	R Reserved	D9 Datacom	R Reserved	R Reserved		F3 PUC	
	8	D10 Datacom	R Reserved	R Reserved	D11 Datacom	R Reserved	R Reserved	D12 Datacom	R Reserved	R Reserved		K3 Growth	
	9	S1 Sync Status	Z1 Growth	Z1 Growth	Z2 Growth	Z2 Growth	M1 REI	E2 Orderwire	National use	National use		N1 NOB	2430

Table 2.2: SDH overhead for an STM 1

Table 2.2 shows the SDH overhead for an STM-1. A distinct feature of SDH is that its frame structure consists of 125µSec time intervals. The advantage of having a consistent frame interval is that low-level signals can be accessed directly from high level signals and that all data manipulations can be performed at byte level [22].

The STM-1 frame structure can be extended as an *n*-fold STM-*n* structure and occupies a 9-row x 270-column byte frame structure as shown in Table 2.2. This frame structure is further arranged into a 3-row x 9-column RSOH and a 5-row x 9-column MSOH. The *Administrative Unit* (AU) pointer occupies a 1-row x 9-column space and splits the upper and lower part of the SOH. The remaining 9-rows x 261-columns are reserved for the STM-1 payload, into which a single VC-4 or three VC3's can be mapped.

The RSOH is used to improve transmission reliability between regenerators. The regenerators only have access to this part of the RSOH and ignore the rest of the frame. The MSOH is used to monitor the multiplex sections. Multiplexers examine and check only the MSOH.

The AU pointer occupies 9 bytes on the fourth column of the STM-1 frame. The AU pointer consists of 3 x H1-, 3 x H2- and 3 x H3 bytes and is employed to track the shifting location of the first byte of the VC-4. The following sections will describe the function of the overhead bytes. These overhead bytes have been divided into mainly two groups namely the section overhead and the path overhead. Each of these overhead groups have been further divided as follows:

Section Overhead:

1. Regenerator Section Overhead
2. Multiplex Section Overhead

Path Overhead:

1. Higher Order Path Overhead
2. Lower Order Path Overhead

In the next sections an accompanying drawing indicates the relevant position of these overhead bytes within the SDH framework.

2.4.Regenerator Section overhead

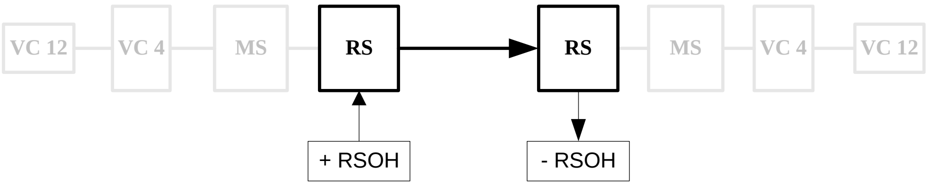


Figure 2.4: Regenerator Section Overhead accessed by Regenerator Section

As mentioned before the RSOH is the overhead that is used by the regenerators. Figure 2.4 shows how the regenerators access the RSOH only. The RS overhead is added to the RS and then stripped away at the receiving RS. A description of the RSOH and their allocated functions will now be given. The location of the RSOH bytes with respect to the SDH frame can be examined in table 2.2.

2.4.1. Framing: A1, A2

The A1 and A2 bytes form the frame alignment word, where the word length is 3xN in an STM-N for N = 1,4,16 and 64. Thus, for an STM-1 there will be 3x1 A1 bytes, and 3x1 A2 bytes to form a 48-byte frame alignment word. The A1 and A2 framing byte pair have a code value of (1111-0110-0010-1000 or F628) as shown in figure 2.5. These bytes are never scrambled and uniquely identify the start of each STM-N frame [1].

A1	1	1	1	1	0	1	1	0
A2	0	0	1	0	1	0	0	0

Figure 2.5: Framing bytes A1and A2

For the STM-0 frame, the frame alignment word is composed of one A1 byte followed by one A2 byte. For the STM-N frame case, where (N = 1, 4, 16, 64) the frame alignment word is composed of 3 x N A1 bytes followed by 3 x N A2 bytes.

2.4.2. Regenerator Section Trace: J0

The J0 byte is transmitted repetitively so that a section receiver can verify its continued connection to the intended transmitter. The J0 byte contains a Section Access Point Identifier, which can either be a single byte containing the code 0-255 or the Access Point Identifier as defined in [11]. For international boundaries and boundaries between different network operators, the format defined in [11] shall be used unless otherwise mutually agreed by the operators [1]. A 16-byte frame is used to transmit the Section Access Point Identifiers, with the first byte of the 16-byte frame being the header byte containing the results of a *(7 bit Cyclical Redundancy Check)* CRC-7 calculation over the previous frame followed by 15 bytes transporting 15 T.50 characters as shown in Table 2.3. The T.50 format is defined in [12].

Byte#	bits							
1	1	C1	C2	C3	C4	C5	C6	C7
2	0	x	x	x	x	x	x	x
3	0	x	x	x	x	x	x	x
4	0	x	x	x	x	x	x	x
:	:							
16	0	x	x	x	x	x	x	x

Table 2.3: T.50 character map

The CRC-7 checksum is calculated for the current frame at the transmitter and only transmitted in the next frame. The receiver then calculates a receive checksum, and compares it with the received checksum. The receiver only checks the CRC-7 checksum every 16th SDH frame. If no error is detected in the checksum then the 15 bytes carrying the T.50 ASCII characters did not change either.

2.4.3. Section BIP-8: B1

A single interleaved parity byte is used to provide STM-N error checking. B1 is used for regenerator section error monitoring and is a *Bit Interleaved Parity 8* (BIP-8) code using even parity. This is done on all the bits in the previous STM-N frame after scrambling and this value is placed into the B1 byte of the current frame before scrambling.

During the parity checking procedure, the first bit of the BIP-8 field of the current frame is set so the total number of ones in the first positions of all octets in the previously scrambled STM-1 frame is always an even number. The same procedure is followed for the second bit, except the check is done on the second bit of each octet of the frame.

2.4.4. Orderwire: E1

This is a 64kb/s channel reserved for voice communications between remote terminals and regenerators for maintenance communications. The idea is that maintenance personnel can connect a telephone at the repeater location into the provided voice jack and have voice access on the fibre span.

2.4.5. User: F1

The F1 byte is reserved for user purposes. The implementation of the F1 field is not standardised, and vendors have the flexibility of deciding the use of the 64 kb/s channel. Use of the F1 byte is completely optional. This allows vendors total freedom of use, but limits vendor interoperability.

2.4.6. RS Data Communications Channel: DCC; D1-D3

D1-D3 form a 192kb/s (3x64kb/s) channel, used for message based operations such as *Operations Administration Management and Provisioning* (OAM&P). The DCC carries information such as alarms, administration data, signal control information and maintenance message. These message are either internally generated, externally generated and manufacturer specific messages.

2.5.Multiplex section overhead:

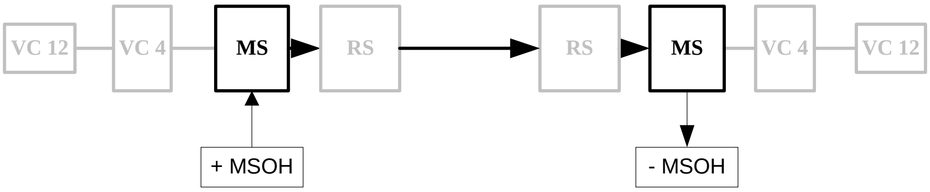


Figure 2.6: Multiplex Section Overhead accessed by Multiplex Section

The MSOH contains the information required between the multiplex section termination equipment at each end of the Multiplex section. As shown in figure 2.6, the MS only accesses the MS overhead bytes. A description of these MSOH bytes will now be given. Refer to table 2.2 for the location of these bytes with respect to the SDH frame.

2.5.1. Multiplex section BIP-8: B2

This byte is used for multiplex section bit interleaved parity code (MS BIP-24). The B2 byte is used to determine if an error has occurred over a multiplex section using the same BIP calculation procedure described for the BIP-8 calculation of the regeneration section. STM-1 uses a BIP-24 parity code.

The key difference between the multiplex section B2 byte and that of the regenerator section is that for higher order STM-N frames, NxB2 bytes are used for the multiplex section case, but only 1xB1 byte is used for the regenerator section.

2.5.2. Automatic Protection Switching (APS): K1/K2

The K1 and K2 bytes communicate automatic protection switching commands between multiplex sections and are specifically used for link recovery following network failure.

2.5.3. Multiplex Section DCC: D4-D12

Bytes D4-D12 form a 576 kb/s multiplex section data communications channel. Typical messages may be for maintenance, administration or alarms.

2.5.4. Synchronisation Message: S1

The S1 byte carries synchronisation status messages between SDH network elements. This allows the network elements to choose the best clocking source from among several timing sources.

2.5.5. Orderwire: E2

The E2 Orderwire byte performs the same function as the E1 byte and is used as a 64kb/s voice channel at the multiplex section level.

2.6.Higher Order Path Overhead (VC-4)

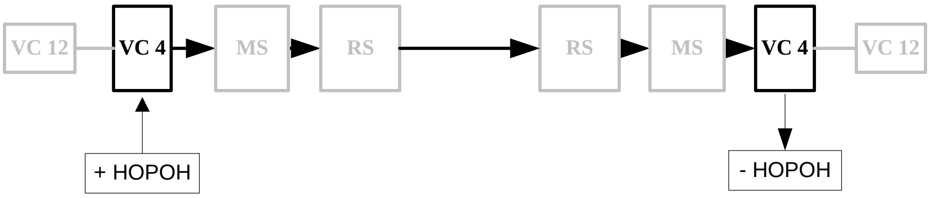


Figure 2.7: VC4 Higher Order Path accessed by VC4

The higher order path terminating equipment only accesses the HOPOH. Figure 2.7 shows this arrangement, where the VC4 HOPOH is added to the VC4 path and removed at the receiving VC4. Nine bytes have been defined for the VC-4 POH as: J1, B3, C2, G1, F2, H4, K3 and N1. The HOPOH is accessed only at the path terminating equipment and executes various functions required for reliable VC-4 path connection.

The following section will give a description of these nine bytes and their function within the HOPOH with their location shown in table 2.2.

2.6.1. Path trace byte: J1

The J1 byte is the first byte of the VC-4 and is used to transmit repetitively a *Path Access Point Identifier* (PAPI), so that the path-receiving terminal can verify its continued connection to the intended transmitter. The byte is user programmable and is a 15-byte E.164 [14] format string, plus one CRC-7 byte, forming a 16-byte word in total, which is identical to the 16-byte frame defined for the J0 byte in 2.4.2.

Within a national network, or within the domain of a single operator, the section access point identifier may use either a single byte containing code 0-255 or the access point identifier format as defined in Appendix A [1]. At an international boundary, or at the boundaries between the networks of different operators, the E.164 format shall be used unless otherwise mutually agreed by the operators providing the transport.

2.6.2. Path error monitoring BIP-8: B3

One byte in each VC-4 is used for path error monitoring. A BIP-8 code is used to determine if a transmission error has occurred over a path. Even parity is used and theBIP-8 value is calculated over all the bits of the previous VC before scrambling and placed in the B3 byte of the current VC-4 frame. The calculation of this BIP-8 value does not include the fixed stuff bytes.

2.6.3. Signal label: C2

The C2 byte is used to indicate the composition of the maintenance status of the VC-4. Appendix B specifies a list of standard binary values for C2.

2.6.4. Path status: G1

The trail termination sink uses this byte to convey the path terminating status and performance of the VC-4 back to the trail termination source as detected. It allows for bi-directional path monitoring from either end of the path or any point along that trail.

	REI				RDI	RESERVED		SPARE
G1	1	2	3	4	5	6	7	8

Figure 2.8: VC-3/VC-4 path status G1

2.6.5. Path User channel: F2

The F2 byte is allocated for path user communication purposes between equipment. The network service provider can use this byte for internal network communications.

2.6.6. Multiframe indicator: H4

The H4 byte provides a multiframe and sequence indicator for virtual VC-3/4 concatenation and a generalised position indicator for payloads [1]. For the case of a VC-2/1 payload, H4 can be used as a multiframe indicator with the last two bits of the H4 byte forming a counter that repeatedly cycles through 00, 01, 10 and 11.

2.6.7. Automatic Protection Switching: K3

K3 bits 1-4 are allocated for APS signalling for protection at the VC4 path level. Byte K3, bits 5-6, is allocated for future use and has no defined value where as K3 bits 7-8 are reserved for a higher order path data link [1].

2.6.8. NOP (Network Operator Byte): N1

The N1 byte provides a *Tandem Connection Function* (TCM).

2.7.Lower Order Path Overhead (VC-2/VC-1)

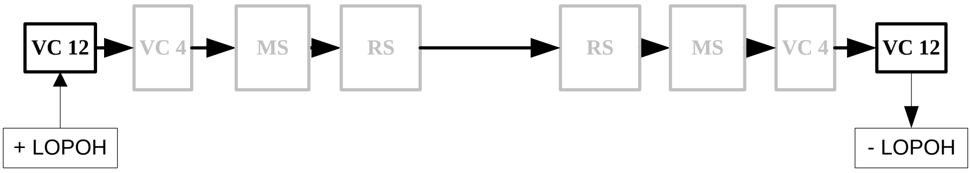


Figure 2.9: VC12 Lower Order Path Overhead accessed by VC12

The LOPOH is dedicated to each of the lower order path VC's and executes various functions for reliable path connection of the lower order VC payload. Figure 2.9 shows this arrangement. The LOPOH is added to the VC12 for each VC12 in the STM-*n*. A description of these overhead bytes follows.

2.7.1. V5 overhead byte:

Bytes V5, J2, N2 and K4 form the VC-2/VC-1 POH. The V5 byte is the first byte of the multiframe, and the TU-2/TU-1 pointer indicates its position.

	BIP-2		REI	RFI	Signal Label			RDI
V5	1	2	3	4	5	6	7	8

Figure 2.10: Lower-Order Path Overhead V5

Byte V5 provides the function of error checking, signal label and path status of the VC-2/VC-1 paths. The bit assignments of the V5 byte are shown in figure 2.10.

- V5 [1-2]: These bits are reserved for the BIP-2 even parity code. Bit 1 is the even parity over all the odd bits of the previous frame, and bit 2 is the even parity over all the even bits of the previous frame and excludes the V1 through V4 bytes except for the case where V3 has data.
- V5 [3]: This is the path remote error indication bit. A high in this location indicates one or more errors in the received BIP-2.
- V5 [4]: The remote fault indication byte is undefined for VC-2 and VC-12 mapping. For VC-11 mapping a high at this location indicates a persistent defect where the duration of persistence of the defect is configurable. This practically means that there is a threshold time level set for the defect, and if the defect persists longer that the threshold time, the remote fault indicator is set high.
- V5 [5-7]: These are the signal label bits. These signal labels bits are filled with specific codes defined in Appendix C.
- V5 [8]: This byte is used as a remote defect indicator. A value of 1 indicates a connectivity and signal failure at the VC-2 or VC-1 level.

2.7.2. N2 Overhead Byte:

Byte N2 shown in Figure 2.11 is allocated for Tandem Connection Monitoring for the VC-2, VC-12 and VC-11 level.

	BIP-2		"1"	AIS	TC-REI	OEI	TC-Apid, TC-RDI,	
N2	1	2	3	4	5	6	7	8

Figure 2.11: Lower-Order Path Overhead N2

No tandem connections will be implemented in this application. A description of this byte is given in Appendix D.

2.7.3. J2 overhead byte:

Byte J2 is used to repetitively transmit a Lower-Order Access Identifier so that a path-receiving terminal can verify its continued connection to the intended transmitter. A 16-byte frame is defined for the transmission of the Path Access Point Identifier. This 16-byte frame is identical to the 16-byte frame of the J1 and J0 byte as shown in table 2.3 of this chapter.

The J0, J1 and J2 byte perform the same function, but the scope of these trace bytes are different. The J2 byte is employed between the VC-12 lower order path terminating equipment. The J1 byte is employed between VC-4 higher order paths. In the case of VC3 mapping, the J1 byte is used as lower order trace byte, with the same J1 byte used as trace byte between the VC4 higher order path.

2.7.4. K4 overhead byte:

Byte K4 bit 1 is used as an extended signal label along with the V5 byte. This extended label field is used if the V5 byte at bit locations five, six and seven is set to “101”when virtual concatenation is used in SDH. For all other values of V5 bits five through seven, the extended signal label bit is undefined and ignored by the receiver.

Bit 1 of the K4 byte, the extended signal label, is used as part of a 32-frame multiframe used in virtual concatenation of tributaries. As for this application, byte K4 is not utilised, but a description of this byte is given in Appendix E.

2.8.Pointer Application: V1, V2 as a TU-12 pointer.

A discussion will now be given on the TU-12 pointer. An example of how the pointer bytes work is given in Appendix E.

Bytes V1, V2, V3 and V4 are allocated the SDH pointer bytes, designated for mapping the following signals:

- A C-2 signal into a VC-2 signal then into a TU-2 signal
- A C-12 signal into a VC-12 signal then into a TU-12 signal
- A C-11 signal into a VC-11 signal then into a TU-11 signal.

When the VC is aligned in the TU, a pointer is added which indicates the phase of the particular VC, which changes during transmission. Phase variation can be due to jitter from regeneration and multiplexing equipment and wander due to temperature differences within the transmission media. Jitter and wander refers to short- and long-term movements in signal rates across the network. SDH is by definition a synchronous system. Therefore, phase stability of clock and data signals throughout the network is fundamental. The four TU pointer bytes are distributed over a 500 µ Sec TU multiframe interval with single frame duration of 125 µ Sec each. These pointer bytes are at fixed positions within the TU.

The TU-1/ TU-2 pointer provides a method of allowing flexible and dynamic alignment of the TU11, TU-12 or TU-2 *Synchronous Payload Envelope* (SPE) within the TU multiframe independent of the actual contents of the envelope [9]. Each 125µSec frame with the bit rates as shown in table 2.4, forming a multiframe is lead by one of the four TU-n (n=11,12 or 2) overhead bytes namely V1, V2, V3 or V4.

Application	Bytes/Frame 125 u sec	Bytes/multiframe 125 u sec
TU-11	27	108
TU-12	36	144
TU-2	108	434

Table 2.4: Frame and Multiframe bit rates

V1 and V2 are used as the *Virtual Tributary* (VT) pointers, V3 is the pointer action byte and V4 is reserved for future standards. V1 and V2 form a 16-bit word as shown in figure 2.12 with the following fields:

- *New Data Field* (NDF): The first four bits of V1 and V2 are the NDF bits and provides a method of allowing flexible and dynamic alignment of the TU-11/TU-12 or TU-2 *Synchronous Payload Envelope* (SPE) within the TU multiframe independent of the contents of the envelope. The TU multiframe is a 500 µ Sec interval.
- TU size field: Bits 5 and 6 indicates the TU size field with TU-12 (10), TU-11 (11) and TU-2 (00). For TU–11 applications there are 27 bytes per 125 µ Sec frame. In the case of TU-12 and TU-2 there are 36 and 108 bytes respectively per 125 µ Sec frame.

- Pointer value field: The last ten bits of V1 and V2 is divided up into two groups of pointer value bits. Bits number 7, 9, 11, 13 and 15 are the increment or “I” bits, and bits number 8, 10, 12, 14, 16 are the decrement or “D” bits.

V1								V2							
0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
N	N	N	N	S	S	I	D	I	D	I	D	I	D	I	D

Figure 2.12: TU-1/TU2 pointer bytes

When the change involves a change in TU size, there shall be a new data transition in all of the TU in TUG-2. During normal operation the NDF bit location shall have a value of 0110 transmitted.

A 2Mb/s E1 PDH signal first enters a container C12 that compensates for the varying speeds via the use of stuffing bits. The VC12 POH is added to form the VC12, which uses *Bit Interleaved Parity* (BIP) to monitor errors and *Far End Block Error* (FEBE), *Remote Fail Indicator* (RFI) and *Far End Receive Failure* (FERF) to indicate errors. The signal label in the V5 byte is usually set at 2 to indicate asynchronous data. A pointer is then added to the VC12 to define the phase alignment of the VC12.

2.9.Pointer Application: H1, H2 as an AU-4 pointer.

This section will describe the H1, H2 and H3 AU-4 pointer bytes. These bytes have the same function as the TU-12 pointer bytes but the scope is different. The AU-4 pointer bytes are used for the higher order tributaries and provide ways to offset the start of the payload.

Figure 2.14 shows the H1, H2 and H3 pointer bytes. These pointers are capable of positive and negative pointer adjustments due to timing variations in the network clocks. These adjustments are done one byte at a time. These pointer values need to be set because the H1/H2 pointers cannot point just anywhere in the synchronous path envelope, but must always point to the position of the first byte of the path overhead. This reduces the need for buffer space and delays in SDH. The H3 byte immediately follows the H1 and H2 overhead bytes and is called the pointer action byte which plays a role in the process of adjusting the value of the H1 and H2 pointer bytes when more or fewer bytes need to be sent in the synchronous path envelope.

The AU-4 pointer will locate the beginning of the VC-4 payload area, which is always headed by the VC4 path overhead. When more than the required bytes for the frame need to be sent, then the H3 pointer action byte location can hold an extra byte. When fewer bytes need to be sent the byte following the H3 byte can be used to hold a special stuff byte, which the receiver will ignore.

These positive and negative justification opportunities are the result of either the input clock running slightly faster than the output clock in which case the input buffer is filled faster than it is read, or the input clock running slightly slower than the output clock in which case the input buffer is not filling fast enough. For the slightly faster input clock a negative frequency justification opportunity is created, and for the slightly slower input clock a positive frequency justification opportunity is created.

H1								H2								H3							
N	N	N	N	S	S	I	D	I	D	I	D	I	D	I	D								
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16								

N - New Data Flag　　I - Increment

S - Size Bit　　D - Decrement

Figure 2.14: H1, H2 and H3 Pointer

2.10. In Summary:

In this chapter a treatment pertaining to the SDH overhead was given with the location of these overhead bytes indicated in the SDH frame. The network was also broken down into segments, with the associated level of responsibility of the relevant overhead shown. The next chapter will discuss a method for managing the information gathered from the overhead bytes, and will show how SDH uses in circuit monitoring for fault detection and alarm generation.

CHAPTER 3:
SDH MANAGEMENT

3.1.Introduction

One of the most powerful features of SDH is the built in standard for *Operations, Administration, Maintenance and Provisioning* (OAM&P). SDH OAM&P involves the day-to-day operations and fault detection in the SDH network. This chapter gives a brief introduction to SDH management. The focus of this thesis is not on the management of SDH, but some of the principles discussed in this chapter have been implemented on the system.

3.2.SDH Management Model

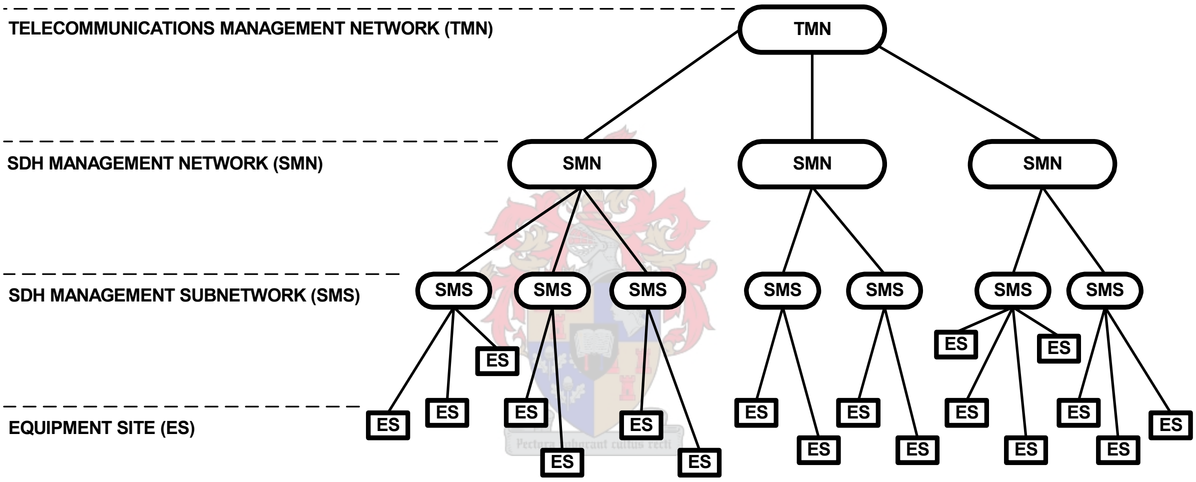

TELECOMMUNICATIONS MANAGEMENT NETWORK (TMN)
TMN
SDH MANAGEMENT NETWORK (SMN)
SMN
SMN
SMN
SDH MANAGEMENT SUBNETWORK (SMS)
SMS
ES
EQUIPMENT SITE (ES)

Figure 3.1: SDH Management Network

Figure 3.1 shows a SDH management Network. It is clear from the figure that SDH management uses a multi-tiered distributed management process. At the lowest level, the SDH management system includes the *Network Elements* (NE's) providing the transport services. These NE's form the *Equipment Site* (ES) blocks. Management of the ES blocks is through the *SDH Management Subnetwork* (SMS), and each SMS can have a number of ES blocks under its control. Some ES's will provide *Operations Management* (OS), which a workstation or personal computer can interface with via the F-interface protocols. Some ES blocks may be equipped with *Gateway Network Elements* (GNE) to interface with the OS via the Q-interface [9].

The *SDH Management Network* (SMN) will manage one or more SMSs and the *Telecommunication Management Network* (TMN) will manage SMN, SONET management networks and PDH management networks.

The OAM channels exist inside the path overhead and section overhead. The path overhead and section overhead have been discussed in chapter 2. In the path overhead specific bytes have been allocated to indicate path *Alarm Indication Signal* (AIS), *Remote Defect Indicator* (RDI), *Remote Error Indicator* (REI), *Remote Fault Indicator* (RFI) and error checking. The section overhead supports the same channels as the path overhead and in addition also supports the *Loss Of Frame* (LOF) channel and supports the network operator maintenance channel which includes orderwire (E1, E2), user channel (F1), and data channels (D1 to D12).

3.3.Fault Management.

OAM refers to the detection of failure, confirmation of failure and maintenance location of failure and isolation and recovery of failure. The AIS signal has the function of notifying a failure in the forward direction, where as the RDI reports the failure in the backward direction. The AIS and RDI signals can be generated at various levels within the SDH hierarchy, thus reporting facilities exist with these layers to accommodate the setting of these alarm conditions. The AIS is generated by setting particular bits in overhead bytes such as K2 in STM-*n*, H1 and H2 in the administrative unit, and V1 and V1 in the tributary unit. In the same way the RDI is generated by setting particular bits within the overhead bytes such as K2 in the STM-n, G1 in the virtual container level 3 and level 4, and V5 in the virtual container level 1 and level 2. These levels are made up of the lower order path, higher order path, multiplex section and regenerator section and is illustrated in figure 3.2. Maintenance is thus done systematically in a layered approach based on the layers just mentioned.

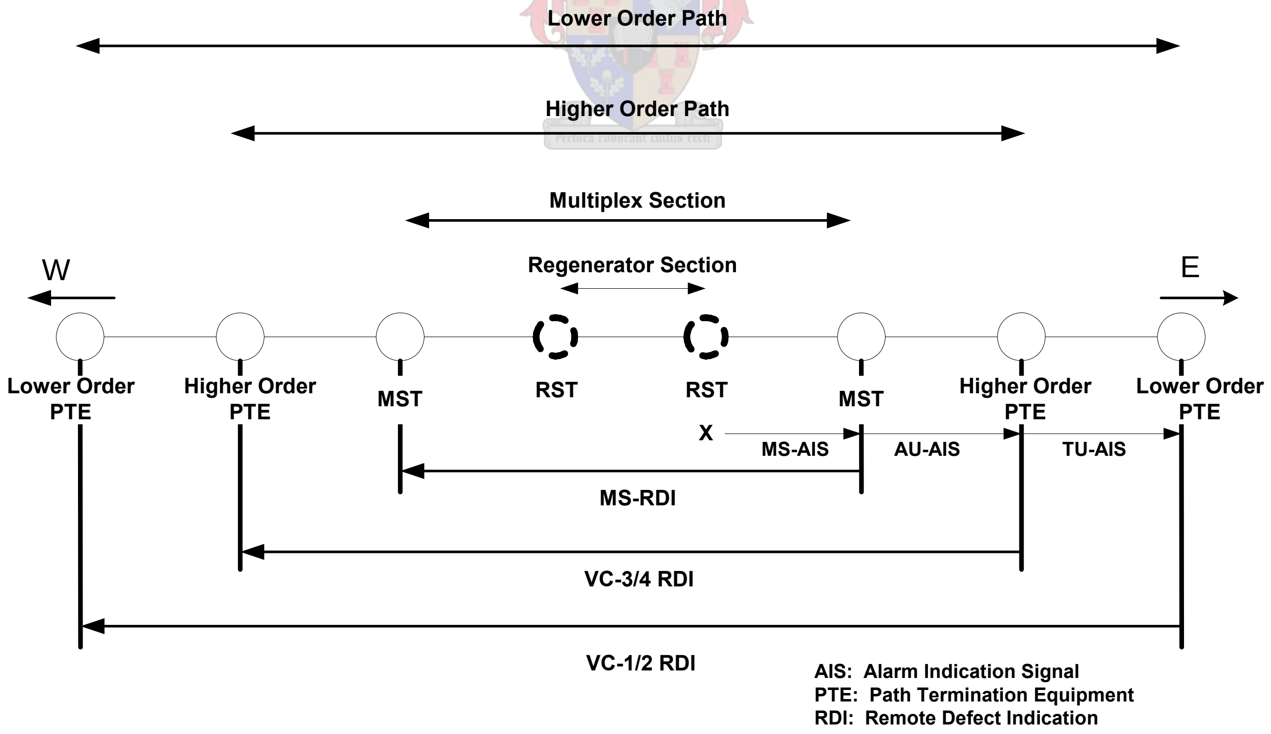


Figure 3.2: Propagation of OAM signal in SDH system.

Figure 3.2 illustrates an example of a failure in the regenerator section of the network. The failure is detected in the *Regenerator Section Termination* (RST) and sends an “all 1” signal (MS-AIS) in the forward direction. This is detected by the *Multiplexer Section Termination* (MST) in the east and an AIS in the forward direction replaces the AU 3/4 signal in the failed STM-*n* with “all 1” data, while at the same time sending a MS-RDI signal backward to the westerly MST.

This “all 1” AIS condition is detected by the higher and lower order path termination equipment in the east, which then in turn transmits the RDI to the PTE in the west [22].

As mentioned before, AIS enables the detection of upward link failure and RDI enables the detection of downward link failure, thus allowing for single ended monitoring of the network path since failure conditions are reported to either side of the link.

Fault management deals specifically with alarm surveillance. Two other fault management functions are testing and external events. Both these functions are for further study by the ITU-T. Alarm surveillance deals with the detection of relevant events, which occur in the network or network elements, and are indications that are automatically generated by a network element resulting from events [9].

From the example above it is clear that AIS and RDI are the major maintenance signals used for reporting events in the SDH network, and can be generated at all levels within the SDH structure.

3.4.Performance Management: Monitoring Function.

The purpose of performance management is the detection of performance degradation at the transmission layer before any interruption at the service layer.

Performance data collection refers to the event counting of each of the performance events namely *Background Blocked Errors* (BBE), *Errored Seconds* (ES) and *Severely Errored Seconds* (SES). Network elements gather performance-monitoring information based on information such as:

- Regenerator section BIP byte, B1. B1 is the performance primitive.
- Multiplex section BIP, B2. B2 is the required performance primitive.
- STM-1 path BIP, B3. B3 is the required performance primitive.
- VC path BIP-2, bits 1 and 2 of the V5 byte. Here the first two bytes of V5 is the performance primitive [9].

For SDH applications REI is the second primitive, required when there is an event of error impairment generated by the NE.

All performance parameters need threshold registers and NE’s must accumulate information on all performance parameters. When the threshold register count exceeds the specified threshold a threshold cross alert must be sent to the associated OS.

Recommendation G.826 [20] specifies the following definitions for error performance events for paths:

- Errored Block: A block with one or more bits in error.
- Errored Second: A one second period with one or more errored blocks or at least one defect.
- Severely Errored Second: A one second period which contains ≥30% errored blocks or at least one defect.
- Background Block Error: An errored block not occurring as part of a severely errored second.

In service monitoring of blocks are performed through inherent *Error Detection Code* (EDC). These EDC bits are physically separated from the blocks to which they apply and it is not possible to determine whether a block or its EDC bits are in error. In such cases the controlling block is always seen as in error.

For the VC12 path a BIP-2 EDC is used with a block size of 1120 bits, and for VC-4 a BIP-8 EDC is used with a block length of 18 792 bits. Other path types for which EDC and block sizes have been defined are VC-11, VC-2 and VC3.

Accumulated count of BIP violations can be used to estimate the number of errored blocks. By counting these violations over 1 second, the number of errored blocks in that second can be estimated. For both BIP-2 and BIP-8 it is recommended that the number of errored blocks in the measurement period equal the number of individual parity violations in the measurement period [20].

By monitoring the SES events for both directions at a single path end point, it is possible to determine the unavailable state of the path. A SES is declared as a second having at least 600 errored blocks in one second for the VC12 path, and 2400 errored blocks for the VC4 path.

3.5.In Summary.

In this chapter an overview was given regarding the hierarchical structure of the SDH management system. This short discussion on SDH management is by no means a comprehensive look at the vast management functions employed in a functional SDH management system. The topics covered serves as a background to those implemented in this SDH system. In chapter 4, the SDR architecture used to develop the SDH architecture is discussed and will give insight into the general workings of the SDR architecture.

CHAPTER 4:
SOFTWARE DEFINED RADIOS

4.1.Introduction

The SDH simulator uses a SDR architecture to manage and link the individual components. This architecture was developed at Stellenbosch University. This section gives an overview of this architecture.

The SDR Forum defines SDR as follows: [2]
Software Defined Radio (SDR) is a collection of hardware and software technologies that enable reconfigurable system architectures for wireless networks and user terminals.

The SDR forum was founded to promote this technology. To ensure a common view the term “software radio” was proposed to be a generic term by using a set of tiers to describe facilities present in each system: [3]

- Tier 0. Hardware Radio (HR)
- Tier 1. Software Controlled Radio (SCR)
- Tier 2. Software Defined Radio (SDR)
- Tier 3. Ideal Software Radio (ISR)
- Tier 4. Ultimate Software Radio (USR)

A software radio fits into the tiered structure depending on the degree or means by which the device parameters can be changed by software. Using the SDR forum definition, a SDR is therefore a Tier 2 software radio.

The United States Federal Communications Commission (US FCC) adopted a more focused definition for SDR:
A radio that includes a transmitter in which the operating parameters of frequency range, modulation type or maximum output power (either radiated or conducted) can be altered by making a change in software without making any changes to hardware components that affect the radio frequency emissions.

Although the definition is more specific than the broader definition provided by the SDR Forum, the FCC definition can be considered as a subset of the broader definition [4].

An SDR does not process the *Radio Frequency* (RF) signal directly and makes use of a hardware stage to down convert the RF signal to an *Intermediate Frequency* (IF) or directly to *Baseband* (BB) for processing by the software that defines the SDR. Hence the software running on it defines the functions of the radio. The software defines processes such as modulation and all aspects of signal processing.

The following sections will give a full description of the SDR used for the implementation of the SDH network. A secondary objective of this thesis is to evaluate this SDR system.

4.2.Software Defined Radio Architecture

In his thesis “Software Architecture Design of a Software Defined Radio System” [5], J.J.Cronjé sets about designing and implementing a general-purpose SDR system.

In searching for a SDR hardware platform, and using the work of Bose [6] and Pucker [7], where a comparative study is made between *applications specific integration circuits* (ASIC), *digital signal processors* (DSPs), *general purpose processors* (GPPs) and *field programmable gate arrays* (FPGAs), Cronjé [5] makes the following conclusion:
No single platform is ideally suited to SDR systems … Until a platform is developed that caters specifically for SDR systems, the optimal solution is a hybrid solution that contains FPGAs and DSPs or GPPs.

This conclusion is important since the realisation of a real system depends strongly on the availability of hardware that would effectively and closely mimic the performance of hardware radios while at the same time moving closer to the conceptual universal software radio.

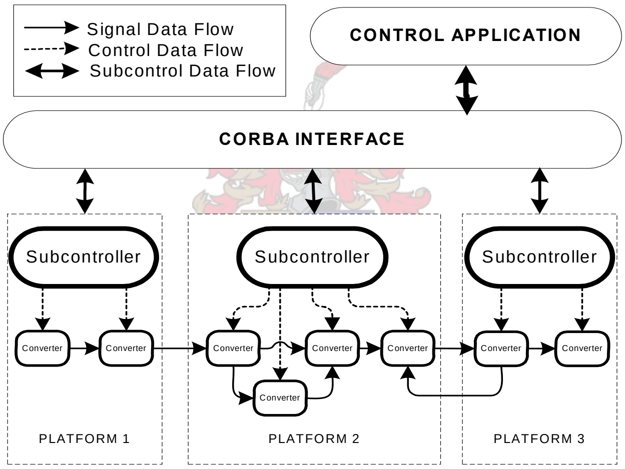


Figure 4.1: SDR system Architecture

The SDR system architecture in Figure 4.1 was designed with a layered approach to aid control and communications. Specifically defined interfaces provide upper and lower layer interconnects. Each layer is independent of the others, and the complete system is built by interconnecting these layers [5].

4.3.Converter layer

In the context of the design shown in Figure 4.1, a converter is an atomic unit that performs a well-defined signal processing function [5]. Each such signal-processing block is abstracted as a unit that converts incoming sample streams into output sample streams – hence the name "converter". In the converter layer, such blocks (e.g. amplifiers, demodulators, modulators) are defined and linked via specified ports, to ultimately realise a software–defined radio system. The converter layer therefore has to be optimised for high-speed signal processing to maximize throughput and minimize latency. It is this optimisation that makes it suitable for the simulation of SDH networks.

4.4.Subcontroller layer

The main function of the subcontroller layer is to manage the converters under its control. Using a round-robin scheduling method, all converters are given the opportunity to process their samples, and subsequent output data are then passed to the input ports of subsequent converters.

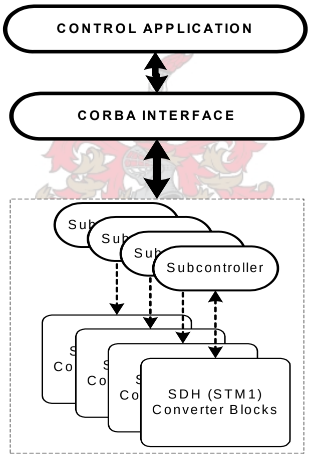


Figure 4.2: Block diagram of an SDH system built on SDR.

4.5.CORBA interface

The *Common Object Request Broker Architecture* (CORBA) standard is defined by the members of the *Object Management Group* (OMG). CORBA is a platform and programming language independent *Remote Procedure Call* (RPC) architecture that offers an elegant way to overcome cross-platform RPC difficulties.

For the system design shown in Figure 4.2, the CORBA interface only channels control information between the main application and the subcontroller layer. The interlayer communication requires very little bandwidth, as any communications are limited to system parameter modifications, the addition of new components and the initiation and termination of signal processing.

4.6.Control application (Main application)

The control application provides the user interface to the system. A *Graphical User Interface* (GUI) is typically implemented here to allow the user to interact with the components of the system. For example, a general-purpose GUI may allow the user to add, select and remove subcontrollers, and to query and adjust their signal processing parameters.

4.7.In Summary

This chapter gave an overview of the SDR architecture developed at the University of Stellenbosch. The SDR architecture is hierarchical with the SDR converters being the first layer. The SDH architecture is developed on this layer, by stringing together these converter blocks in a predefined order as defined as discussed in chapter 2. The following chapter will show how the SDH architecture was implemented on the converter layer, and how these converters were managed by the subcontroller.

CHAPTER 5:
THE IMPLEMENTATION OF AN SDH BI-DIRECTIONAL LINK USING SDR

5.1. Introduction

This chapter describes the developed implementation for simulating a bi-directional SDH multiplexer at a base STM-1 rate, using the SDR architecture described in chapter 4. The multiplexer has been implemented to allow for C-12 mapping only, thus allowing the E-1 2.048Mb/s signals a mapping opportunity onto the SDH structure. The C-12 then follows the SDH multiplexer path via the VC4 to eventually form an STM-1. The multiplexer was implemented first, followed by the functional implementation of the attributes of each converter block. For the purposes of this implementation, the functional implementation has been developed to the VC4 level only. The RSOH and MSOH have not been included in the functional implementation but have been included as part of the SDH multiplexing structure.

Figure 5.1 illustrates a high-level overview of the SDH STM1 transmission system, using the SDR architecture for component management. Initially one subcontroller is used as a platform for both systems. This single subcontroller is responsible for handling all communications to the converter blocks, and will manage the information transfer between all the converter blocks, which form the STM1 SDH system.

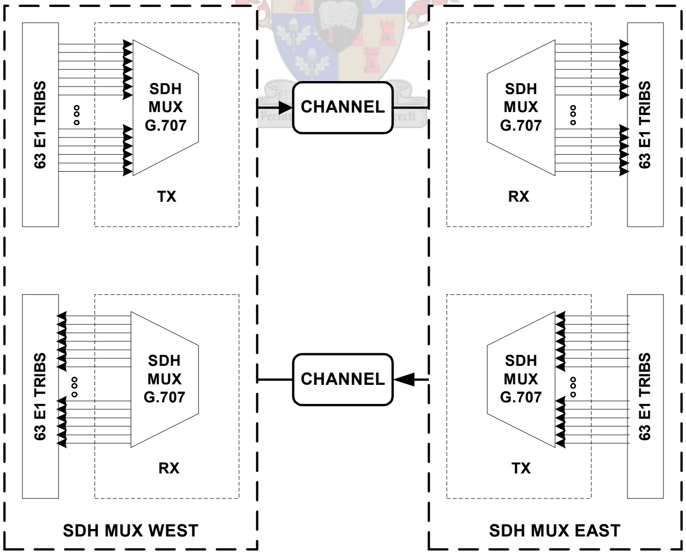


Figure 5.1: STM terminal -direction EAST and WEST

Before starting the discussion of this implementation a very important feature and possibly the most important feature of SDH, which has thus far not received any attention, is that of synchronisation and timing. As the name suggests SDH is a synchronous hierarchy, which distinguishes itself from other networks. Synchronisation problems arise from jitter and the wander of low quality oscillators. As mentioned before, jitter and wander refers to short- and long-term movements in signal rates across the network. In SDH a primary reference clock specified by the ITU-T in G.811 is used which is accurate enough to ensure proper synchronisation of the network. Primary reference clocks cannot be deployed on all network elements since the economic effect of such an engineering practice will be too costly. A synchronisation network is used, where some of the transmission links are used to transport the synchronisation signal [**23**]. Clocks are placed into a hierarchy based on relative performance levels, which are numbered as strata 1 to strata 4. Each stratum level has a progressively less accurate and expensive clock [**9**].

In this thesis the timing considerations of SDH are not exploited. The simulated network is never plagued by clock inaccuracies, and thus it is assumed that the receiver and transmitter are in a perfect synchronised condition. The effect of pointer justification in this simulator has not been considered. A fixed mode multiplexing structure has been implemented. This mode will be further described in section 5.3.4 of this chapter.

Extensive use was made of recommendation G.783 [**8**] in defining the components and methodologies for the SDH system. The recommendation illustrates specification methods based on the functional decomposition of equipment into atomic and compound functions. This functional decomposition of equipment into atomic and compound functions is key to the development of the converter layer. These atomic units also support an environment where well-defined units can be reused in the same or other applications.

5.1.1. SDR converter blocks

The SDH simulator has been developed using the SDR architecture. The SDR architecture provides converter blocks that can be used as areas where code can be defined for execution.

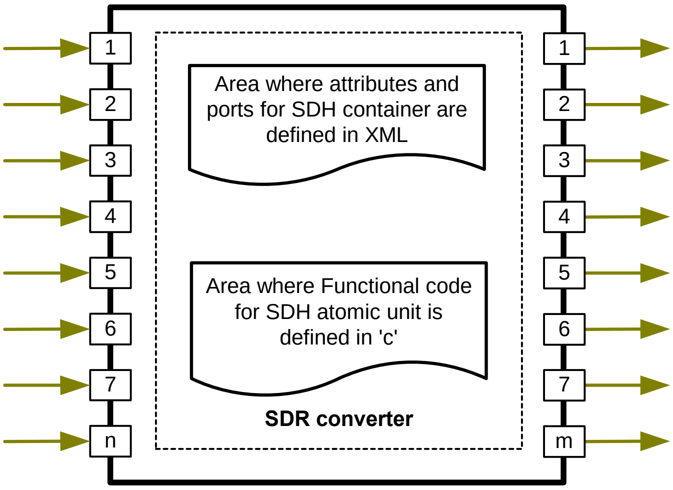

1
2
3
4
5
6
7
n
Area where attributes and ports for SDH container are defined in XML
Area where Functional code for SDH atomic unit is defined in 'c'
SDR converter
1
2
3
4
5
6
7
m

Figure 5.1.1: SDR Converter block

Figure 5.1.1 shows the SDR converter block used for developing the SDH atomic units. These blocks of code then form the individual atomic units of a greater system. A feature of these converter blocks is that they can be interconnected via input and output ports using a predefined syntax. The number of input and output ports per converter block as well as the number of attributes is under the control of the system developer.

In the SDH architecture and functional description of the atomic units, reference will be made to various converter blocks or containers such as E1, C12, VC12, TU12, TUG2, TUG3, VC4, RS, MS and STM1. These SDH containers have been developed using the SDR converter block structure shown in figure 5.1.1, and have been interconnected according to the SDH multiplexing structure.

Holding this structure together is the SDR subcontroller. The subcontroller manages all the SDH containers and ensures that each container has a chance to execute its code, and manages the transfer of data between the SDH containers.

The following sections will describe the SDH multiplexer. Whenever reference is made to a container or converter block, then a similar structure to that shown in figure 5.1.1 was used in the development of that container or converter block.

5.2. Architecture

This section describes the design considerations when developing the architecture of the SDH system using SDR. This is a detailed overview of the multiplexing structure, and how the structures that form a STM-1 signal was interpreted and built into SDR models.

The architecture was developed using a layered approach, where the one layer communicates with the following layer using specifically assigned ports. Except for the interconnecting ports, which besides attaching one layer to the next also have the function of transporting data (section 5.3), these layers are totally independent from one another. A consideration throughout this design was that of layering and breaking down complex functions into atomic units. This made the implementation of a complex SDH system into SDR more natural since the SDR architecture has been developed to deploy well-defined atomic units. Well-defined atomic units support an environment where atomic units can be re-used. This is one of the strengths of SDR, and all through this SDH design attempts are made to clearly define the range of responsibilities of each atomic unit.

The following sections have been divided into two categories:

ARCHITECTURE:
Deals with the implementation of the multiplexing structure. The function of the overhead is not considered. The function of the multiplexing structure is to combine the individual 2Mb/s tributaries or clients into a STM-1 signal. Once in a multiplexed format, this data stream can then be transported via a channel, which can be altered to simulate certain signal degrading conditions.

FUNCTIONAL UNITS:
The functional units describe the function of the overhead bytes, and how these bytes work together in this implementation of a SDH network. SDH has been designed to perform certain maintenance and monitoring tasks via the overhead bytes that was added during the development phase of the SDH multiplexer.

Each stage of the discussion will gradually build the SDH multiplexer to STM1 level and finally the functional units will be introduced to give meaning to the overhead that was added during the development of the architecture.

5.2.1. ARCHITECTURE: - The development of the PDH unit.

Though PDH does not form a part of the SDH network, its function in this development cannot be ignored, as it is the starting point for the generation of data in the SDH network. The PDH model represents clients renting 2Mb/s channels on the SDH link. In a real world situation these 2Mb/s channels can have their origin at geographically different places, and traverse these distance via microwave radio links. The point is that these channels are independent in terms of the data being carried, and can even be from different PDH networks.

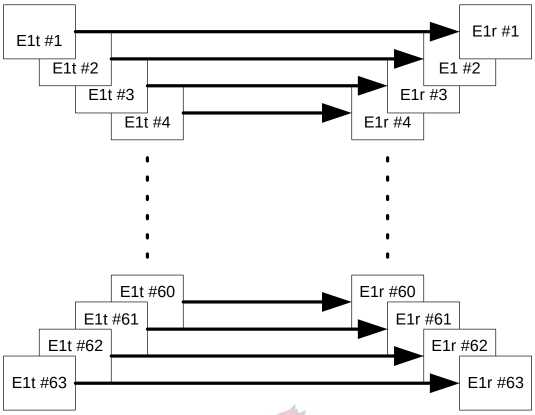


Figure 5.2: 63 x E1 transmit and receive atomic blocks.

Figure 5.2 is an illustration of how 63 2Mb/s channels were represented using the SDR architecture. The exact reason for illustrating 63 channels as opposed to any other number of E1 channels is a function of the STM-1 implementation and will become clearer as this discussion continues.

Each converter block marked as E1t#n {n =1...63} for the transmit side and E1r#n {n=1...63} for the receive side represents a 2Mb/s link. Each of these 2Mb/s links is further divided into 32 channels, with each channel having a channel capacity of 64kb/s. The 32 channels correspond with the European PDH level 1 signal. Two of these 32 channels are dedicated for signalling and alignment, thus only 30 channels are user channels.

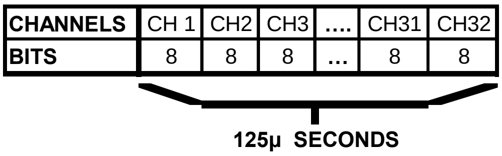
In real world applications sampling a 4kHz bandwidth analogue signal 8000 times per second and then representing these samples with an 8-bit code obtain these 64Kb/s signals.

Two very important details arise from this sampling and coding process.

1. A sampling rate of 8KHz produces a sampling period of 125µSec.
2. Each sampling point is represented by an 8-bit code.

As will be seen throughout this implementation, this 125µSec-sampling period forms the basis of all frames developed in SDH, leading up to the STM-1 signal. In fact higher-level signals having multiple rates of STM-1 like STM-4, STM-16, STM-64 all have frame durations of 125µSec.

Table 5.1 shows one 125µSec frame of an E1 signal. The channels are marked from channel 1 to channel 32, and each channel represents some sampled quantity in an 8-bit format.



CHANNELS	CH 1	CH2	CH3	….	CH31	CH32
BITS	8	8	8	…	8	8

125µ SECONDS

Table 5.1: Representation of 32 8-bit channels forming an E1.

The E1 transmit converter blocks and the E1 receive converter blocks have been developed to simulate table 5.1. The E1 transmit converter block generates random 8 bit codes or characters. Since there are 32 channels in an E1 signal, 32 random 8-bit codes or characters are generated to represent sampled data at an instance of time. After 32 eight-bit codes have been generated, the cycle is repeated, thus each channel has randomly varying consecutive 8-bit codes.

The E1 receive converter block as shown in figure 5.2 has been designed to receive each of these random 8 bit codes at the relevant channel. From figure 5.2, the course that this 8 bit code has to take seems obvious, since the E1 transmit converter block has been directly linked to the E1 receive converter block. Though obvious now, these direct links will become less obvious as the multiplexing stages are implemented. The goal however, is to ensure that the 8-bit code at E1 transmit channel 1 goes to E1 receive channel 1 and the 8-bit code at E1 transmit channel 2 goes to E1 receive channel 2 and so on until we reach channel 63.

Omitted from this E1 implementation so far is the signalling and synchronisation channels. These two channels have no real meaning in this application, hence the 8-bit code for these channels is no different from the 8-bit codes generated for any of the other channels.

Not shown in figure 5.2 is an Error Count converter block that was developed to aid in the end-to-end testing of each channel. The Error Count converter block kept track of the generated data at the E1 transmit converter block and the received data at the E1 receive converter block, and then did a comparison to detect any difference between the received and transmitted data. This Error Count converter block was used throughout the development of the SDH multiplexer as an immediate code testing tool.

The E1 transmit converter blocks can be seen as a generator and therefore has no input ports, and a single output port. The 32 random 8-bit codes are transmitted serially via the output port of the E1 transmit converter block.

Wrapping up the discussion on the E1 transmit and receive converter blocks, the following must be noted:

1. After the generation of 32 random 8-bit codes a period of 125µSec has passed. This is the frame duration of the 32-byte frame.
2. Four of the consecutively generated 125µSec frames produce a multiframe.
3. Each of the 32 random 8-bit codes represents a 64 Kb/s channel.

5.2.2. ARCHITECTURE: - The development of the LOWER ORDER PATH. C12

Owing to the vertical structure of SDH a pragmatic approach is used to develop the rest of the STM1 SDH structure using SDR components. The transmit and receive of each developed converter block at the same hierarchical level, in this case the lower order path, can be interconnected. This interconnection of converter blocks, aid development in that the intermediate structures of the SDH multiplexer can be tested long before the final multiplexing structures have been implemented.

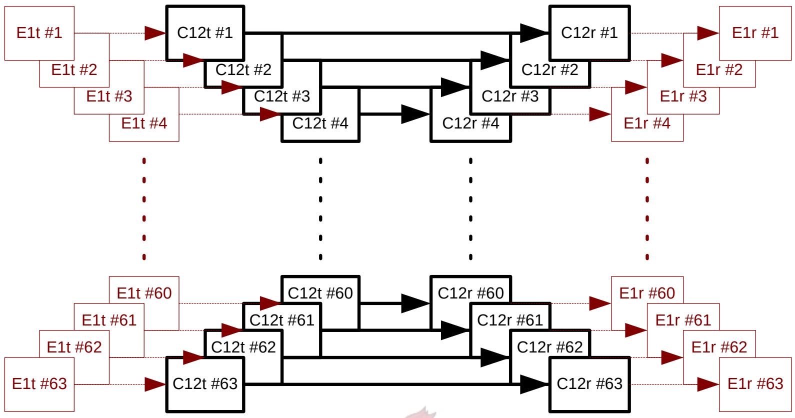


Figure 5.3: C12 transmit and receive converter blocks.

Figure 5.3 illustrates the next stage of the implementation of the multiplexer. The highlighted converter blocks represent the mapping opportunity for the previously developed E1 signals. These converter blocks are labelled as C12t#n {n = 1…63} and C12r#n {n= 1…63} as shown. These transmit and receive converter blocks represent an SDH container 1 of type 2. These containers are defined in the SDH multiplexing structure and provide a mapping opportunity for PDH E1 signals onto the SDH multiplexer. Various containers exist to provide mapping opportunities for various non-SDH signal rates. The only mapping opportunity developed for this application has been that of the C12 container.

The C12t converter block will be discussed first, followed by the C12r converter block.

C12t:

The C12 transmit converter block has a single input port and a single output port. Figure 5.3 shows how these ports are connected. C12 transmit gets its input from the output of the E1 transmit converter block. From the discussion in the previous section, recall that the E1 transmit converter block repeatedly transmits a 125µSec frame, consisting of 32 random 8-bit codes. Once transmitted, these consecutive frames are buffered at the input port of the C12 transmit container before being processed by the C12 transmit container.

The C12 transmit converter block reads a single frame from the input buffer. This input frame is 32 bytes in length, and places a stuffing byte before the first byte and a stuffing byte after the last byte of this frame.

Table 5.2 illustrates this frame. The frame generated by the C12 transmit converter block has to maintain a frame duration of 125µSec, even though the frame length has increased by 2 bytes. Thus a total of 272 bits are transmitted per frame producing. 2.176 Mb/s

	1	2	3	4		32	33	34
CHANNELS	STUFF	CH 1	CH2	CH3	**....**	CH31	CH32	STUFF
BITS	8	8	8	8	**...**	8	8	8

125µ SECONDS

Table 5.2: Representation of one C12 frame.

Four of these C12 transmit frames are called a multiframe. In a real world application these stuffing bytes would have different bit assignments per frame in the multiframe depending on the type of mapping that was used. These bit assignments offered stuffing opportunities to accommodate for differences in the signal rate of the mapped E1 signal to that of the transporting SDH system.

In this application only fixed stuffing “STUFF” [table 5.2] bytes were used. This model of the mapping sequence from the PDH environment in the SDH environment does not consider the effects of variations in these two signal rates.

C12r:

The main function of this converter block is to remove the STUFF bytes from the frame, and to present the subsequent signal to the converter block of the lower layer. Similar to the C12 transmit converter block, the C12 receive converter block has a single input port and a single output port. The removed STUFF bytes are discarded when removed from the frame, resulting in a 32-byte frame. Once this frame in transmitted by the C12 receive converter block, the bytes then appear at the input port of the next converter block.

In summary:

- 32 bytes representing a single 125µSec E1 frame is presented to the input port of the C12 transmit converter block.
- This signal is mapped into the C12 transmit converter, where 2 STUFF bytes are added. These STUFF bytes are located at the beginning and end of the resulting frame.
- The C12 receive converter removes the STUFF bytes from the received signal.
- This 32 byte frame is then transmitted by the C12 receive block to the input of the converter block at the lower level.
- Thus figure 5.3 represents a model of the mapping process in SDH where 63 E1 signals are mapped into 63 C12 containers.

5.2.3. ARCHITECTURE: - The development of the LOWER ORDER PATH. VC12

The next discussion of the implementation involves the development of the VC12 converter blocks. Here a multiframe is always considered since different overhead bytes are added to each of the four frames of the multiframe. A period of 500µSec is considered.

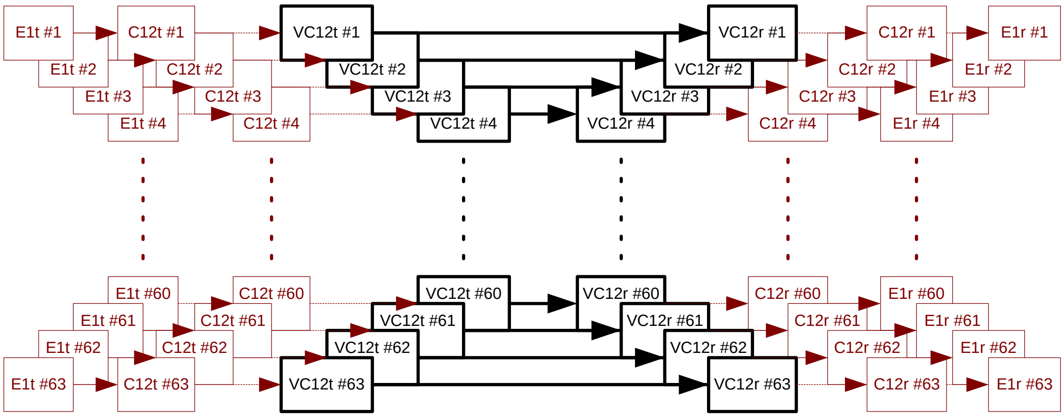


Figure 5.4: VC12 transmit and receive converter blocks.

The highlighted section of Figure 5.4 shows the next implementation of an atomic unit called a VC12. There are just as many VC12 converter blocks as there are C12 converter blocks. Where the main function of the C12 converter block was to allow for mapping, the VC12 converter block has the function of adding overhead bytes to the frame. A different overhead byte is added to each frame of the multiframe. This process is then repeated for all multiframes.

Shown in figure 5.4 are 63 converter blocks marked as VC12t#n {n = 1…63} for the transmit side and VC12r#n {n=1…63} for the receive side.

VC12t:

The VC12 transmit converter block has two input ports and a single output port. The connection to the second input port is not shown in this figure but is used for monitoring the overhead bytes of this container. The converter block describing the monitoring function will be discussed later in this section.

The reason for considering a multiframe in this discussion is that four overhead bytes namely V5, J2, N2 and K4 are added to the start of each of the frames in the multiframe and is assigned as attributes of the container. A discussion of the function of the overhead bytes will be covered later in this chapter as part of the functional description of the converter blocks. This then is the responsibility of the transmit VC12 converter block. A process of reading a single frame from the input buffer of the transmit converter, adding the relevant overhead for the particular frame and transmitting this new frame to the input port of the next converter block.

Table 5.3 shows this new multiframe and the locations of the added overhead bytes. The four frames of the multiframe can be identified on the figure from the black and grey shading. These overhead bytes (V5, J2, N2 and K4) lead every frame of the multiframe. The size of a single frame is now 35 bytes, since 1 more overhead byte was added to the frame, with the full multiframe being 140 bytes long. Only once the 140 bytes are transmitted is the process repeated.

	1	2	3		34	35																			140
CHANNELS	**V5**	**S**	**CH 1**	**....**	**CH32**	**S**	J2	S	CH 1	....	CH32	S	**N2**	**S**	**CH 1**	**....**	**CH32**	**S**	K4	S	CH 1	....	CH32	S
BITS	**8**	**8**	**8**	**...**	**8**	**8**	8	8	8	...	8	8	**8**	**8**	**8**	**...**	**8**	**8**	8	8	8	...	8	8

$\underbrace{\hspace{10cm}}_{\text{500µ SECONDS}}$

Table 5.3: VC12 500Sec multiframe indicating the added overhead bytes per frame.

VC12r:

The VC12 receive converter block has a single input and two output ports. The second output port is also not shown in this figure but is connected to the input port of the monitoring converter block and has the function of examining the overhead bytes of the VC12 receive converter block.

The function of the VC12 receive converter block is to examine and remove the VC12 overhead bytes from the start of each of the frames and to present this new frame to the next converter block.

The VC12 transmit and the VC12 receive converter blocks all operate independently.

5.2.4. ARCHITECTURE: - The development of the LOWER ORDER PATH. TU12

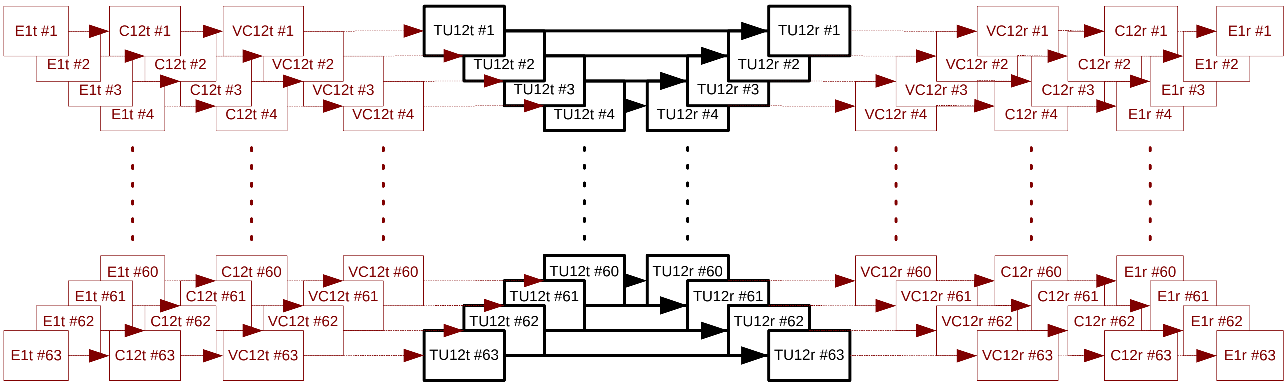


Figure 5.5: TU12 transmit and receive converter blocks.

Figure 5.5 shows the TU12t#n and TU12r#n converter blocks that have been added to the arrangement. These converter blocks have the responsibility of dealing with the pointer bytes for the lower order path. Four pointer bytes are added namely the V1, V2, V3 and V4 pointer bytes. These bytes form the attributes of the transmit and receive converter blocks.

In the transmit converter blocks, these pointer bytes are added 1 byte per frame for the multiframe. The V1 and V2 bytes have the specific function of pointing to the V5 byte, which is the leading byte of the multiframe. After the fourth pointer byte has been added, the cycle is repeated. This TU12 transmit container extends the frame by one byte, to produce a new TU12 frame of length 36 and frame duration of 125µSec. At this stage the exact working of the bytes are not important as long as the architecture provides the capacity for these bytes.

The receive TU12 converter block has the function of removing these overhead bytes from the TU12 frame. This is done at the start of each frame. Before the removal of these bytes, the pointer values are first used to find the start of the V5 byte. This V5 byte is the first byte of the multiframe.

5.2.5. ARCHITECTURE: - The development of the TUG2 and TUG3 multiplexing stages.

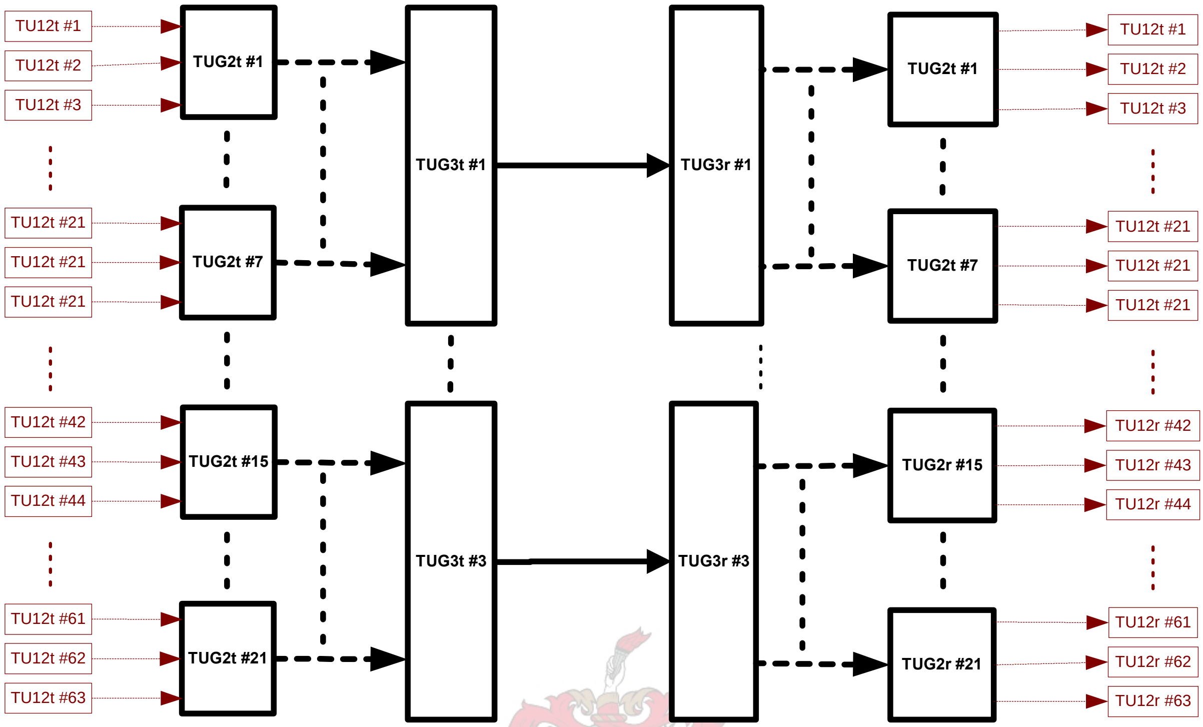


Figure 5.6: TUG2 and TUG3 multiplexing process.

Figure 5.6 shows the multiplexing stages involved in combining 63 TU12 transmit converter blocks into a single output stream. The connections prior to the TU12 transmit and the connections after the TU12 receive converter blocks are not shown. This figure leads on from figure 5.5 thus detail of converter blocks prior to the TU12 stage should be examined in the preceding figures.

From the figure two main converter blocks can be identified namely the TUG2 converter block and the TUG3 converter block. These converter blocks will be discussed next.

TUG2t:

There are 21 TUG2 transmit converter blocks. Each converter block has three input ports, and a single output port. The input ports receive data from the TU12 transmit converter blocks. These TU12 transmit converter blocks are grouped together in threes, and provide the input to the TUG2 transmit converter block.

The TUG2 transmit converter block performs the task of byte interleavingly multiplexing the TU12 input data into a single data stream. The TUG2 transmit has a capacity of 108 bytes which is the result of multiplexing three 36 byte TU12 transmit frames into a single data stream. No overhead is added during this procedure.

The procedure is to read a byte from the first input port, then a byte from the second input port and then a byte from the third input port and then repeating the process until 108 bytes have been read. The frame of the TUG2 transmit converter block is thus 108 bytes in length with a frame duration of 125µSec.

TUG2r:

There are 21 TUG2 receive converter blocks. Each receive converter block has a single input and three outputs ports. The function of the TUG2 receive converter block is opposite to that of the TUG2 transmit converter block. The serial input stream is demultiplexed into individual data streams and passed to the three output ports. Data that arrive at port 1 from the transmit TUG2 converter block must exit port 1 at the receive TUG2 converter block. The same goes for the rest of the ports. The TUG2 receive converter block therefore receives a frame of length 108 bytes, and demultiplexes it into three frames of 36 bytes each.

TUG3t:

There are only three TUG3 transmit converter blocks. These converter blocks have the same function as the TUG2 converter blocks in that it performs a multiplexing function. Each TUG3 transmit converter block has seven input ports and a single output port. One byte is read from each of the seven input ports and the process repeated until 756 (108 x 7) bytes have been read in total. To this is added 18 stuffing bytes to give the TUG3 transmit converter block a total capacity of 774 bytes. The TUG3 byte structure can be seen as a 9 row by 86-column structure as shown in figure 5.7 with the location of the fixed stuffing bytes located in column 1 and 2. The output data from the seven TUG 2 structures occupy columns 3 to 86 and rows 1 to 9.

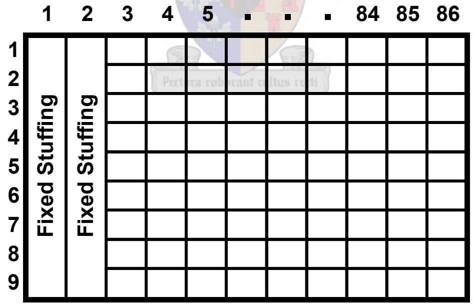


Figure 5.7: Multiplexing of seven TUG 2 converters via a TUG 3 converter

The frame duration for this 774 byte long TUG3 transmit frame is 125µSec.

TUG3r:

There are three TUG3 receive converter blocks. Each TUG3 receive converter block has a single input and seven outputs. The TUG3 receive converter block has the function of demultiplexing the serial input data stream into seven data streams for each of the output ports. The added fixed stuffing bytes shown in figure 5.7 are now removed and a data steam of 108 bytes per port is transmitted to the next stage. This 108-byte data stream has a frame duration of 125µSec.

5.2.6. ARCHITECTURE: - The development of the HIGHER ORDER PATH.

In the preceding discussions it was shown how 63 E1 signals were mapped into a C12 container and then through various multiplexing stages these input signals were combined into three data streams as the outputs of TUG3t#1, TUG3t#2 and TUG3t#3 [see figure 5.6]. The next stages will combine these three data streams into a single output and add different overhead bytes at various stages to finally form the STM-1 signal.

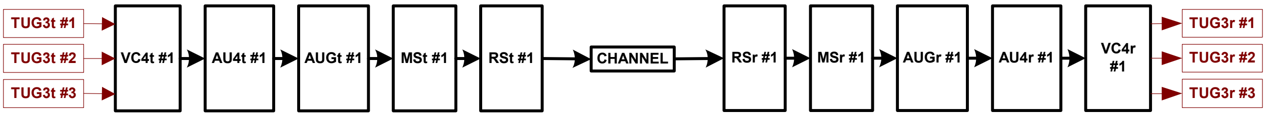
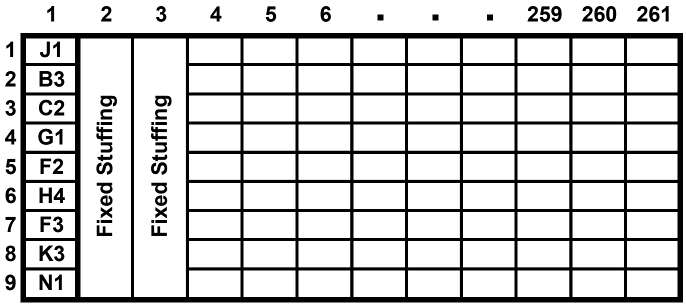


Figure 5.8: Multiplexing structure of the HIGHER ORDER PATH.

Figure 5.8 shows the VC4, AU4, AUG1, MS and RS converter blocks. Another converter block shown in the figure is the converter block labelled as channel. This block does not form part of the SDH structure and was used to inject errors into the data stream to model a fibre connection between the two end converter blocks. This allowed for the simulation of error condition. A Gaussian distribution was used in generating the channel noise.

VC4t:

The VC4 transmit converter block has the function of multiplexing the three TUG3 data streams into a single data stream and is labelled as VC4t#1 in figure 5.8. The three data streams are multiplexed into a single data stream using a byte interleaving process. 18 Stuffing bytes and 9 VC4 path overhead bytes are added to this data stream. These VC4 path overhead bytes have been assigned as attributes in the VC4 container and are shown in column 1 rows 1 to 9 of figure 5.9.



	1	2	3	4	5	6	.	.	.	259	260	261
1	J1	Fixed Stuffing	Fixed Stuffing									
2	B3											
3	C2											
4	G1											
5	F2											
6	H4											
7	F3											
8	K3											
9	N1											

Figure 5.9: VC4 transmit container capacity

Three frames, each having a frame length of 774 bytes are multiplexed into a single VC4 container and with the added overhead and fixed stuffing bytes results in a total VC4 transmit capacity of 2349 bytes. The 2349 byte frame has a frame duration of 125µSec.

VC4r:

The VC4 receive converter block is labelled as VC4r#1 in figure 5.8. This converter block performs a demultiplexing process as well as removing the VC4 path overhead bytes and fixed stuffing bytes from the input data stream. The converter block has a single input port and three output ports that links to the three TUG3 receive converter blocks. A fourth output port not shown in the figure links to the VC4 monitoring converter block, which monitors the VC4 path overhead bytes. The VC4 monitoring converter block will be discussed later in this section.

The rest of the converter blocks are the AU4, MS and RS. The MS and RS converter blocks were only implemented in the architectural structure, but no functional code was developed for the overhead bytes of the MS and RS converter blocks. Their function is purely to build the capacity of the STM-1 frame.

The AU4 transmit converter block labelled as AU4t#1 in figure 5.8 has the function of adding 9 pointer bytes to the received frame. The frame capacity of the AU4 converter block is now the received VC4 frame, which has a length of 2349 bytes, plus the 9 added overhead bytes, giving a total frame capacity of 2358 bytes per 125µSec transmitted frame.

The AU4 receive converter block labelled as AU4r#1 in figure 5.8, has the function of removing the pointer bytes from the received frame. The payload is then extracted and passed to the VC4 receive converter block.

The MS transmit converter block and the RS transmit converter block add 45 bytes of overhead and 27 bytes of overhead respectively. As mentioned before, the functional implementations of the MS and RS overhead bytes were not implemented.

	1 2 3 4 5 6 7 8 9	10	11	12	13	5	6	▪	▪	▪	268	269	270
1	27 Bytes of RS overhead bytes are added	J1	Fixed Stuffing	Fixed Stuffing									
2		B3											
3		C2											
4	AU4 pointer bytes	G1											
5	45 Bytes of MS overhead bytes are added	F2											
6		H4											
7		F3											
8		K3											
9		N1											

Figure 5.10: Full STM1 frame.

After adding the MS and RS overhead bytes, the resulting frame is considered an STM-1 frame with a frame length of 2430 bytes. This frame is shown in figure 5.10 where the location of the added overhead is clearly visible. As with the other frames generated after each converter block, the frame duration of this frame is 125µSec.

This completes the discussion on the developed SDH architecture, where an attempt was made to resemble the SDH multiplexer discussed in chapter two and shown in figure 2.10. As mentioned before only the C12 mapping route was implemented which allows for E1 mapping only. A key feature of this developed architecture has been the hierarchical approach to the design.

Mainly two factors facilitated this approach:

- SDH is a vertical structure and therefore implementing such a structure in software naturally lead to a bottom up type development methodology.
- SDR facilitated this approach by providing the architecture and methods for the seamless implementation of this vertical structure.

Because of the vertical structure of SDH, each developed layer could be tested as the structure developed. Features of the SDR architecture allowed these converter blocks to be interconnected during the initialisation phase of the execution. Multiple converter blocks performing fundamentally the same tasks, such as the E1, C12, VC12 and the TU12 converter blocks where 63 of each converter block were implemented were simply defined as a class and instances of the class were called to build the converter block capacity. Even though some of these converter blocks are identical, they perform as individual converters based on the randomly generated data. This will become clear in the functional discussion that follows.

This architecture was implemented using a single subcontroller to manage the transfer of data between the converter blocks.

5.3. Functional implementation

The preceding sections of this chapter created awareness that different bytes with different responsibilities are added at different levels to the developed SDH architecture, and in some cases hints were given as to their responsibilities and functions. The next couple of sections aim to pull these bytes together in terms of how they function as a system. There are various systems within the SDH architecture, and attempts to model some of these systems using the SDH architecture developed using SDR models will now be described.

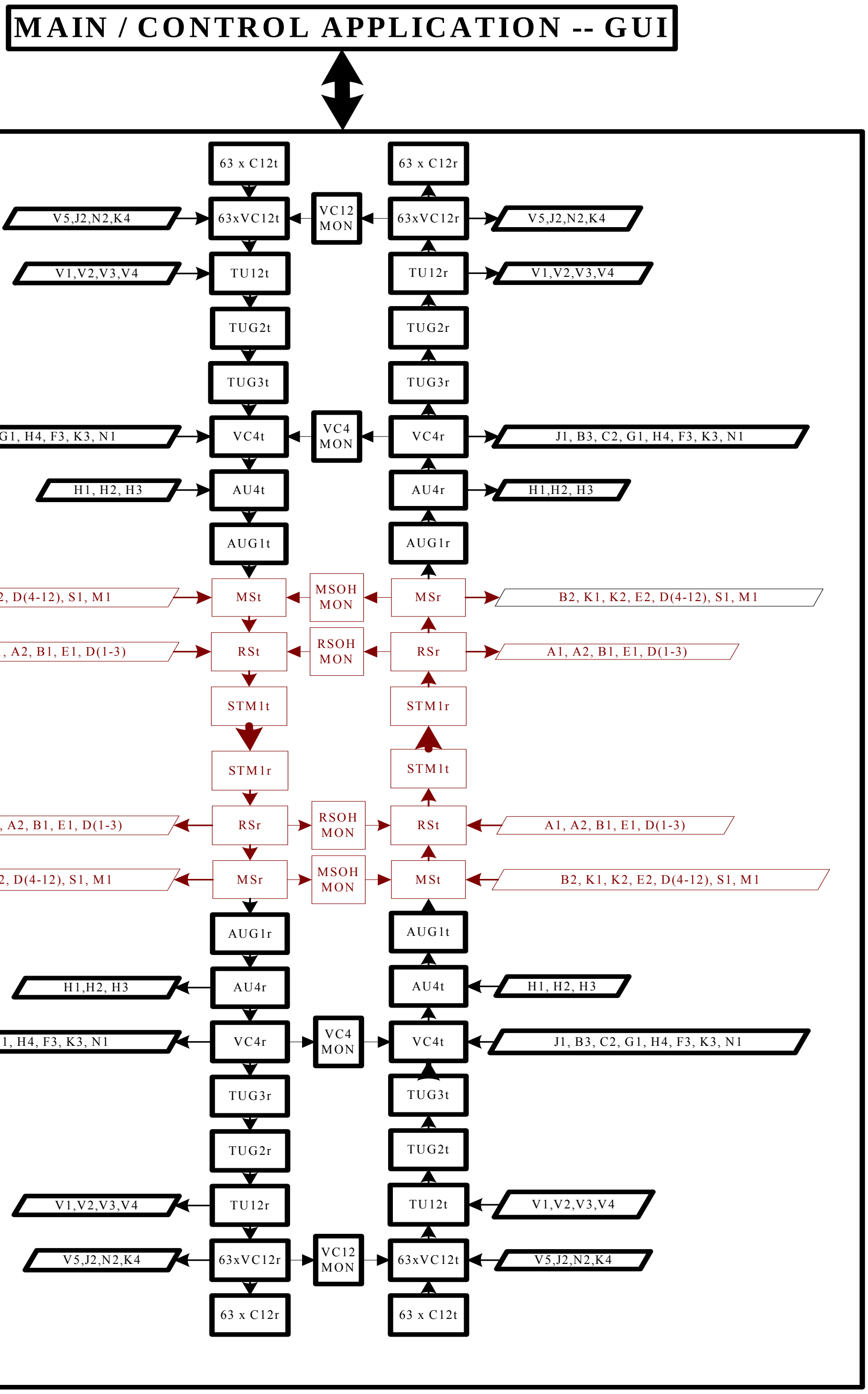


Figure 5.11: STM 1 using a single subcontroller

Figure 5.11 shows the implementation of the SDH multiplexing structure to form an STM1 signal. A single subcontroller manages all the converter blocks, which in turn are managed by the main or control application. The figure clearly identifies the overhead bytes at the various sections. The functional code for the converter blocks marked with a lighter shade has, as mentioned before, not been implemented in terms of how those overhead bytes work together to form systems. Therefore the systems for the multiplex section and regenerator section will not be discussed. The developed architecture however does allow for future development of these functional units.

In figure 5.11 all the components forming the STM-1 multiplexer are shown. The single subcontroller has the duty of handling the data transfer between the converter blocks, which now form the atomic units of the SDH link. These atomic units have specific functional code that processes the data at its inputs. The atomic units are only executed if there is data at the input of the specific atomic unit and that data must be at least the length of a single frame. Data that amounted to less than a single frame were left in the input buffer until the next round of processing opportunity. The processing opportunity is a function of the SDR architecture whereby the converter blocks are given a processing opportunity using what is called a Round Robin [5] processing technique. The principle of this technique is that only one converter block is processed at a time when using a single subcontroller, and the converter block is only processed once per round.

The network management function of SDH, and how it is implemented in this design will also be discussed by showing how such a graphical user interface was developed for the SDH system using QT designer. This chapter concludes with an example of how this SDH simulator can be further utilised as a statistical tool to possibly aid network designers.

5.3.1. Availability and Performance Evaluation – LOWER ORDER PATH

Performance monitoring is used to gather, store and set thresholds and report performance data for early detection of problems. This is a feature of SDH that is implemented in the lower order path (VC12) and higher order path (VC4) of the design. Figure 5.13 shows one layer of the SDH multiplexer shown previously in Figure 5.11, which is that of the lower order path layer. One of the advantages of using a layered or vertical structure is that the individual layers can be developed and tested independently of the other layers.

The lower order path is the first layer on the source side after mapping the E1 signal into the SDH structure and also the final layer on the sink side, which allows the payload to be extracted from the C12 container back into the receiving E1 converter block. Only a single transmit and receive path is shown, but 63 of these virtual VC12 paths have been implemented in the system. A virtual path is formed between VC12 transmit MUX WEST (Source) and VC12 receive MUX EAST (Sink). This path is fixed, and not determined by the overhead. The linking of the components which defines the paths between the components are done during the initialisation phase of the execution, and parallels the integration phase of hardware in the field in a real world system. Once the initialisation phase is executed, the SDH structure is set.

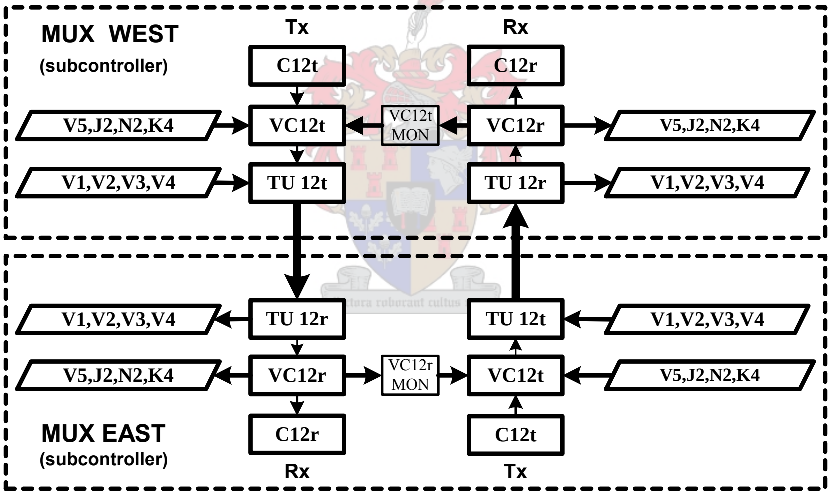


Figure 5.13: Lower Order Path Layer with overhead bytes

5.3.2. Monitoring VC12 lower order path

The error monitoring function for the lower order path is performed by the V5 byte, shown in figure 5.13. The V5 byte is added to the VC12t container and extracted at the VC12r container. Also shown are the J2, N2 and K4 bytes. The V5 byte is only added in the first frame of every multiframe. Bytes J2, N2 and K4 are added at the start of frame two, three and four. Data is read from the input port and a BIP-2 calculation is performed on all the bytes including the added overhead bytes.

After reading data equivalent to a multiframe, which is 34x4 bytes, the calculated BIP-2 value for the four frames is then stored in two bit locations in the V5 byte of the next multiframe. This V5 byte then passes through the TU12t and TU12r converter blocks unaltered.

The VC12r MUX EAST converter block does the same BIP-2 calculation on the multiframe, and passes this value to the VC12r MON MUX EAST converter block. On the arrival of the next multiframe, which is lead by a V5 byte, with the BIP-2 value of the previous frame as contents, the V5 byte is once again passed to the VC12r MON MUX EAST converter block, where a comparison is made between the BIP-2 value calculated at the receive and the BIP-2 value received from VC12t MUX WEST.

A discrepancy in the two values indicates an error or errors in the multiframe. The course of action then is to alert the source side, and this is achieved using the VC12r MON MUX EAST converter block. From figure 5.13 and focusing on the section labelled as MUX EAST, it can be seen that the VC12r MON converter block has been connected to both the VC12r and VC12t converter blocks. The same architecture is implemented for MUX WEST. This is a crucial part of the development of the SDH MUX, since this functionality allows the opportunity for relaying error information back to the source at the level of origin within the hierarchy.

The VC12r MON converter block on MUX EAST then sets a REI condition in the V5 byte, which is then transmitted to the VC12t MUX EAST converter block and transmitted all the way back to the VC12r MUX WEST converter block. Software in the VC12r MUX WEST converter block monitors the REI byte for a high condition.

Using this method, error monitoring of the entire VC12 virtual path is achieved for each of the 63 VC12 virtual paths. These errors are then processed using a method that will be described later in this chapter, and the generated alarm condition is then reported to the SDH management system.

5.3.3. Path Trace at the VC12 lower order path.

Figure 5.13 also shows another important byte in the SDH architecture, and that is the J2 byte known as the trace identifier byte. Each of the 63 VC12 virtual paths is uniquely defined within the network. This unique name is called the trace identifier. Each VC12t converter block in MUX WEST and MUX EAST has an attribute that is assigned a unique 15-byte string. At the same time the corresponding VC12r converter block is also assigned the same 15-byte string. Assigning these unique names is done during the initialisation phase where code written for this purpose generates unique 15-byte strings for each of the 63 VC12 virtual paths. Though this process has been automated in this application, this function would typically be done manually using the SDH management system to perform this task in a real SDH system.

This unique name identifies the type of path, the origin and destination, the country and company by using 15 bytes of ASCII characters. The 16th byte of the trace identifier is a CRC checksum calculated over the 15 bytes of ASCII characters. The J2 byte, the same as the V5 byte is only transmitted once per multiframe, and since the trace identifier is

16 bytes long, 16 transmissions of the J2 byte are needed for the receiver to build the received string. Thus 16 multiframe transmissions or 64 STM-1 frames are required to transmit a single trace identifier.

Once the VC12r converter block has received all 16 bytes, a CRC checksum is calculated over the 15 bytes of ASCII characters, and this result compared with the received CRC checksum that was calculated in the VC12t converter block. If this checksum differs, the entire trace identifier is rejected and a lower order path trace identifier mismatch alarm is raised.

If the CRC checksums match, the received 15-byte string is compared with the 15-byte string that was set up in the VC12r converter during the initialisation phase. If the two are identical, then the path is correctly routed and no alarm is generated. If the expected trace identifier differs from the incoming trace identifier an alarm is generated.
This process is repeated for as long as the path is active. Since the assigned unique trace identifier in the VC12t converter does not change after the initialisation phase, the CRC checksum is only calculated once and stored for retransmission. It is only on the receive side, VC12r, where the CRC checksum is recalculated after every retransmission of the 16-byte string.

Tributary	VC12 Labels direction A->Z	VC12 Labels direction Z->A
1	ZA/SU/A/Z/64K00	ZA/SU/Z/A/64K00
2	ZA/SU/A/Z/64K01	ZA/SU/Z/A/64K01
3	ZA/SU/A/Z/64K02	ZA/SU/Z/A/64K02
4	ZA/SU/A/Z/64K03	ZA/SU/Z/A/64K03
.	.	.
.	.	.
.	.	.
60	ZA/SU/A/Z/64K59	ZA/SU/Z/A/64K59
61	ZA/SU/A/Z/64K60	ZA/SU/Z/A/64K60
62	ZA/SU/A/Z/64K61	ZA/SU/Z/A/64K61
63	ZA/SU/A/Z/64K62	ZA/SU/Z/A/64K62

Table 5.4: J2 Trace identifiers for 63 VC12 virtual channels.

Table 5.4 lists some of the 63 15-byte trace identifiers that were generated during the initialisation phase of the code.

Tributary 1 of the table will be used to describe the notation:

ZA: This is the first field and identifies the country. *South Africa*.

SU: This is the second field and identifies the company. *Stellenbosch University*

A: The third field identifies the origin. *MUX WEST*.

Z: This is the fourth field, which identifies the destination: *MUX EAST*

64K00: The fifth field and identifies the type of circuit. *A 64 Kb/s circuit, with the "00" indicating the first circuit.*

The format of the *Trace Identifier* (TI) has been specified in G.707 [1]. T.50 [12], and M.1400 [13]

The N2 and K4 bytes shown in figure 5.13 have not been implemented since their use falls outside the scope of this application. The implementation of the V1, V2, V3 and V4 bytes will be described next.

5.3.4. Floating and locked mode Pointer Interpretation: TU-12 pointer

Floating mode:

Pointers are very important mechanisms within the STM-1 frame. It allows the payload to float within the fixed 125µSec frame. This means that the payload can either shift up or down the 125µSec frame, and even have one part of the payload in the first 125µSec frame, and the second half in the following 125µSec frame.

Locked mode:

In the locked TU mode, the transport is a fixed mapping of synchronous structured payloads into a VC-12. This means that there will be no phase difference between the payload and that of the synchronous frame. The tributary information is therefore at a fixed location and immediately identifiable with respect to the TU-12 pointer bytes.

For this application the locked mode has been implemented. As mentioned in the introduction of this chapter the simulated network is never plagued by timing inaccuracies. Therefore the receiver and transmitter are in a perfectly synchronised condition, which makes the locked mode an obvious choice for this multiplexer.

The last bytes to discuss in figure 5.13 and leading on from the above discussion are the V1, V2, V3 and V4 bytes. The TU-12 pointer value is contained in the V1 and V2 bytes, which form a 16-bit word. For the locked mode case, these bytes no longer have any meaning, and now form part of the payload.

The TU12t container reads from its input buffer a 35-byte frame. To this frame is added the V1 byte, which holds an arbitrary value that will be ignored by the TU12r converter block. The arbitrary V1 byte is added to the frame since the next stage of the multiplexer expects that the frame being transmitted by the TU12t converter block will be 36 bytes long.

5.3.5. Availability and Performance Evaluation – HIGHER ORDER PATH

This discussion will follow the same format as the previous discussion for the VC12 lower order path. Figure 5.14 shows the VC4 higher order path layer with the associated overhead.

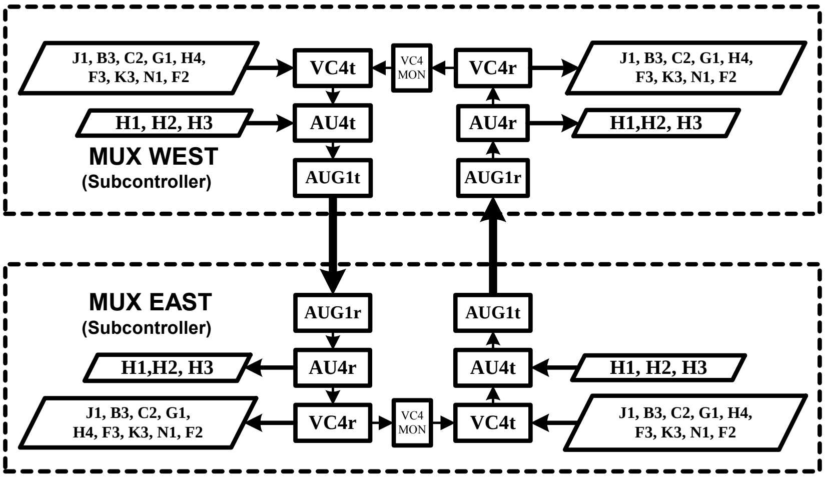


Figure 5.14: Higher Order Path Layer with overhead bytes

As mentioned in the previous section, the vertical structure of the SDH MUX allows for the independent testing of the individual layers. In the vertical SDH structure, the lower order path layers sits on top of the higher order path layer. The higher order path layer was modelled using the converter blocks shown in figure 5.14.

The overhead bytes in the figure marked as the J1, B3 and G1 byte have the same function as the J2 and V4 bytes discussed in the previous section for the lower order path. Only the scope of these bytes is different. The J1, B3 and G1 bytes only operate at the VC4 higher order path level. During operation the converter block labelled as the VC4t converter block builds a frame by byte interleaving data that is read from its three input ports. To these bytes are added 18 stuffing bytes and the nine overhead bytes shown as J1, B3, C2, G1, F2, H4, F3, K3 and N1 in figure 5.14.

5.3.6. Monitoring VC4 higher order path.

Error monitoring is performed between the VC4t and VC4r converter blocks. The B3 byte is used for this purpose, where a BIP-8 even parity calculation is performed on the current VC4 frame. This BIP-8 result is then stored in the B3 byte of the next frame and transmitted unaltered to the VC4r converter block. The VC4r converter block does the same BIP-8 calculation on the current frame, and then compares this result with the B3 value of the next frame. This process is repeated for all transmitted frames, and ensures that anomalies are detected for each transmitted frame.

If the BIP-8 value calculated at the receiver and the transmitted BIP-8 value is different, then the frame has an error. It is possible that the error introduced might not be in the frame, but in the BIP-8 itself. In such a case it is not possible to determine if the frame is in error or the BIP-8 value itself and it is assumed that the frame was in error.

The VC4MON on the MUX EAST is responsible for monitoring the B3 byte, and counting the number of detected errors in the frame. By comparing the bit values of these two BIP-8 values, the non-matching bits are counted as errors. The result of this error count is then stored in the first four bits of the G1 byte. The G1 byte can store 9 legal values, or 0-8 errors. The remaining possible values represented by these four bits are interpreted as zero errors. These four bits in the G1 byte is called the remote error indication bits.

The VC4 MON converter block on MUX EAST transmits this result to the VC4t converter block after which it is transmitted to the VC4r converter block on MUX WEST.

Monitoring software in the VC4r converter block then detects the error count in the received G1 byte. An alarm is then generated to alert the management system and the error for this path is added to an accumulator that tracks the total errors while the path is active.

5.3.7. Path trace at the VC4 HIGHER ORDER PATH LEVEL.

The J1 byte in figure 5.14 has the same function as the J2 byte of the lower order path shown in figure 5.13.

Exactly the same method as the one described in section 5.3.3 is used for the J2 byte. The main difference between the J1 byte and the J2 byte lies in its scope of operation. Assigning a unique name to the single VC4 virtual path might seem odd, since only a single VC4 is implemented. Thus is not always the case since higher rate implementations do have more than one VC4 virtual path.

5.3.8. Locked mode Pointer Interpretation: AU-4 pointer

The locked mode described in section 5.3.4 was used for the implementation of higher order path. The H1, H2 and H3 bytes had arbitrary values in the AU4t converter block. The AU4r converter block ignores these values, and extracts the payload from the received frame.

5.3.9. VC12 MON and VC4 MON converter blocks

The VC12 MON and VC4 MON converter blocks monitor each layer’s signal integrity. In the source direction it generates and adds error detection code and trail trace identification. It conveys back remote error indication signals containing the number of EDC violations in the received signal and continuously monitors the RDI and REI signals.

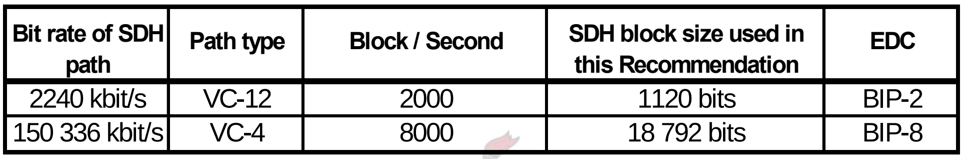
In the sink direction it monitors for some or all of the following: bit errors, connection status, near-end performance, far-end performance, server signal fail and signal loss. To enable single-ended maintenance, the defect status and number of EDC violations detected at the sink trail termination are conveyed back to the source trail termination. The defect status is conveyed via the RDI signal and the number of EDC violations is conveyed via the REI signal.

The defects are reported to the fault management process. The fault management system is integrated into a GUI and will be discussed in the next two sections, section 5.3.10 and 5.3.11 of this chapter.

5.3.10. SDH management: Converting BIP measurements into Errored blocks

One of the components of SDH management involves the constant monitoring of the lower order and higher order paths. The SDH management functions allow for the setting of thresholds and the generation of alarms and have been briefly discussed in chapter 3. In this section, the implementation of the monitoring function is discussed. The monitoring function has been integrated into a GUI, and provides a visual representation of the network and the status of the network.

The two BIP-2 bits in the V5 overhead are used for error checking in the VC12 lower order path whereas the B3 byte in the VC4 POH is used for BIP-8 measurements. Each BIP-n in the SDH path overhead pertains to a single defined block.



Bit rate of SDH path	Path type	Block / Second	SDH block size used in this Recommendation	EDC
2240 kbit/s	VC-12	2000	1120 bits	BIP-2
150 336 kbit/s	VC-4	8000	18 792 bits	BIP-8

Table 5.5: Block size of SDH path performance monitoring

Table 5.5 gives a description of the block sizes associated with the path types and the *Error Detection Code* (EDC) used. These codes were discussed in the previous sections in relation to the converter blocks. The VC12 block rate is 2000 blocks per second and the VC4 block rate is 8000 blocks per second.

The VC12 and VC4 monitoring converter blocks have been specifically developed to translate the monitored BIP-2 and BIP-8 values into more usable quantities such as ES, SES and BBE. These values are then automatically requested by the GUI, and displayed per VC12 or VC4 virtual path.

The four basic events have been mentioned in chapter 3 under SDH management. They are:

- EB, Errored Block (G.826): Block with one or several errored bits.
- ES, Errored Second (G.826): A period of one second including one or several errored blocks or at least one defect.
- SES, Severely Errored Second (G.826): A period of one second with a number of errored blocks equal to or higher than 30% of the total number of blocks in a second, or at least one defect.
- BBE, Background Block Error (G.826): An errored block not occurring as a severely errored second.

Figure 5.15 shows the flow diagram that was implemented in the VC12 and VC4 monitoring converter blocks. This flow diagram shows the method used to track the errors associated with the virtual paths of the SDH link. The monitoring is done per second. An errored second is counted if any of the 2000 blocks in that second at the VC12 level reports a code violation. At a VC4 level an errored second is counted if any of the 8000 blocks in that second has a code violation. If the errored blocks for a particular second exceed 30% of the total amount of blocks for either the VC12 or VC4 virtual path, then the SES count is incremented.

The BE, ES and SES counts are assigned to attributes in the monitoring converter blocks and is updated for the duration of the active link.

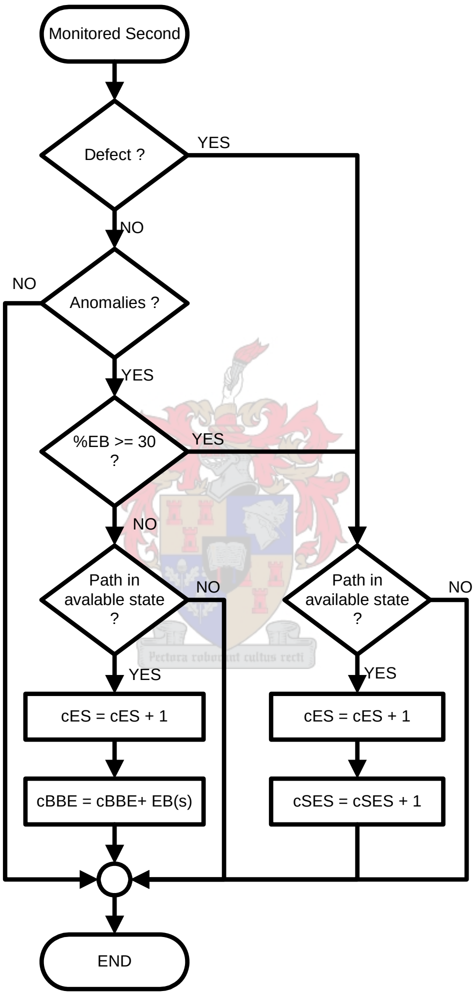


Figure 5.15: Flow chart illustrating the recognition of anomalies, defects, errored blocks, ES, SES and BBE

The availability of a link is broken into unavailable and available time. The concept of availability and unavailability has been implemented in both the VC12 and VC4 monitoring converter blocks and will be discussed next. Figure 5.16 will be used to illustrate these concepts. For a unidirectional link, an unavailable period starts with the occurrence of the first SES of 10 consecutive SES. These 10 seconds are considered to be part of unavailable time. An available period starts with the occurrence of the first non-SES of 10 consecutive non-SES seconds. These 10 seconds are considered to be part of available time [**20**].

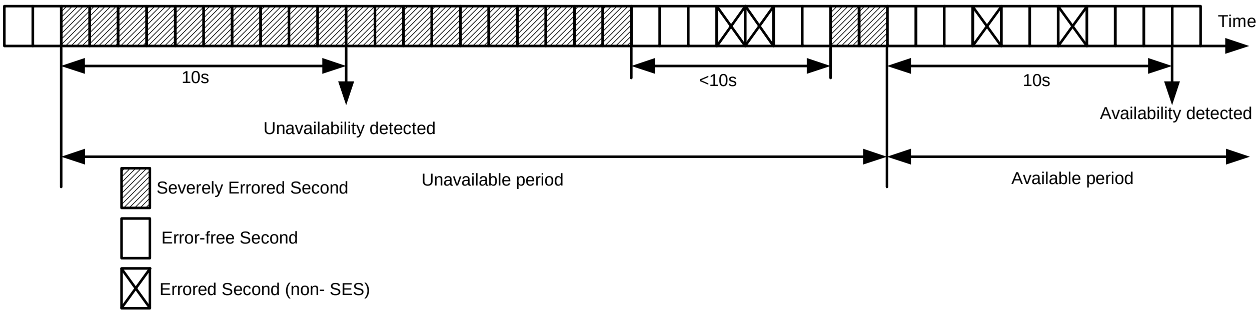
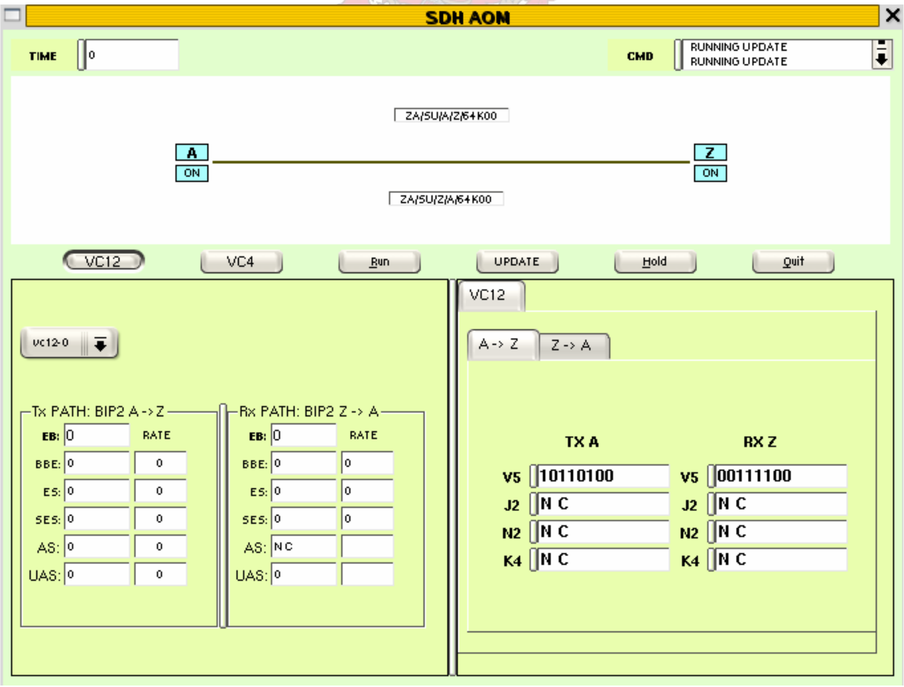


Figure 5.16: Determining unavailability

The error performance measurements are collected in both directions and are very helpful in isolating faulty paths. These error performance measurements are then displayed in a GUI and will be discussed in section 5.3.11. Each of the 63 VC12 virtual paths performs the error performance measurements based in their inherent error detection codes.

5.3.11. SDH Management using GUI

The following screen shots show the GUI that was developed using QT designer. QT designer is freely available GUI development software for non-commercial applications. The choice for this software was mostly driven by the availability of the software. Other factors included the ease of integration of QT with the SDR components and the relative ease of use of the software.



TIME: 0

CMD: RUNNING UPDATE
RUNNING UPDATE

ZA/SU/A/Z/64K00

ZA/SU/Z/A/64K00

vc12-0

Tx PATH: BIP2 A -> Z

		RATE
EB:	0	
BBE:	0	0
ES:	0	0
SES:	0	0
AS:	0	0
UAS:	0	0

Rx PATH: BIP2 Z -> A

		RATE
EB:	0	
BBE:	0	0
ES:	0	0
SES:	0	0
AS:	N C	
UAS:	0	

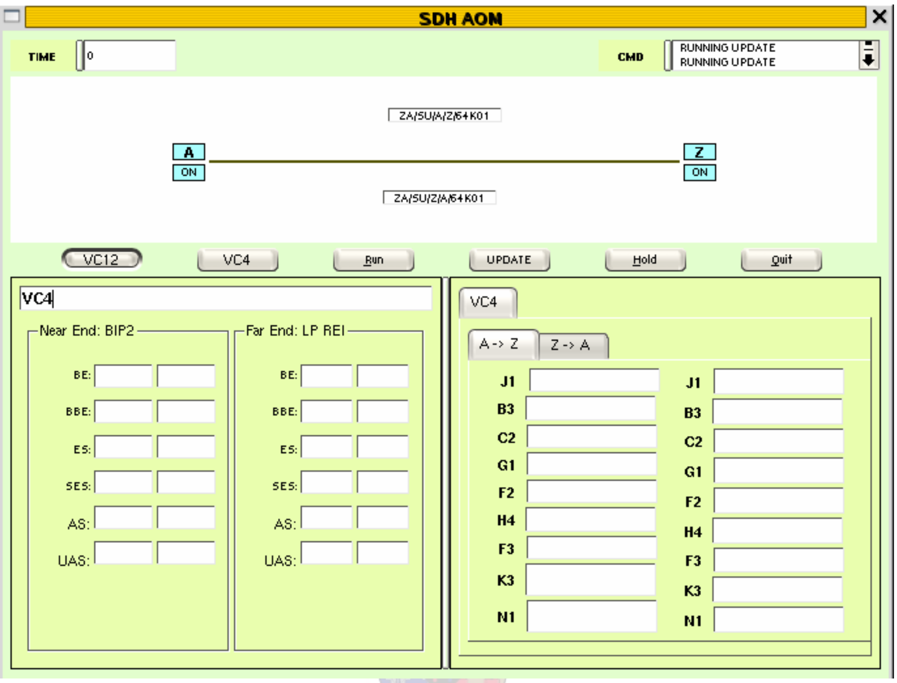
	TX A		RX Z
V5	10110100	V5	00111100
J2	N C	J2	N C
N2	N C	N2	N C
K4	N C	K4	N C

Figure 5.17: Screen shot of the GUI showing point-to-point link and VC12 monitoring function.

Figure 5.17 shows the GUI that was developed using QT designer. The figure shows two blocks marked A and Z, representing SDH MUX EAST and SDH MUX WEST. The subcontroller status is also shown. Since the SDH MUX is a group of SDR converter blocks managed by a subcontroller, the status of the subcontroller indicates the status of the SDH MUX. The SDR architecture allows one to request the current status of the subcontroller.

VC12 monitoring is also shown, indicating the EB, BBE, ES, SES, AS and UAS. These measurements are shown for both transmit and receive paths. The VC12 overhead bytes are also shown.

Since there are 63 VC12 virtual channels, a VC12 drop down menu is provided to allow for the selection of any of the 63 VC12 channels. The associated error measurements and VC12 path overhead values will then immediately be requested and displayed in the provided space.


TIME 0

CMD RUNNING UPDATE
RUNNING UPDATE

VC4

Figure 5.18: Screen shot of the GUI showing the point-to-point link and VC4 monitoring function.

Figure 5.18 shows a second screen shot of the SDH GUI. By selecting the VC4 push button the VC4 error measurements are shown. These measurements indicate the status of the VC4 path. A single STM1 signal was implemented thus only a single VC4 is shown. The VC4 path overhead bytes for both transmit and receive paths are also shown.

The next section will describe how this SDH MUX can be further used as a statistical tool to aid designer.

5.4. Application for the simulation of *Mean Time Between Failure* (MTBF)

This section demonstrates how the developed SDH architecture can be used as an effective model of a telecommunications network to solve statistical network related problems. One of these applications is as a tool to aid the designers in balancing the workforce requirements in terms of staff to maintain a certain service level agreement, based on the size of the network and the MTBF of the equipment.

The following discussion will briefly describe MTBF followed by a description of how the developed SDH architecture was adapted to produce an outcome that could be used as a guide.

5.4.1. Short Discussion on MTBF and associated problem description

This discussion of MTBF is by no means extensive. An attempt is made to give meaning to the term, and to describe the use of the MTBF value.

MTBF is an indication of the average time between failures and can either be based on historical data or estimated by vendors and used as a benchmark for reliability. MTBF assumes that the product can be repaired and resume its normal operations.

It is generally defined as the number of hours under observation divided by the number of failures. If the product is in use all the time then the MTBF is equivalent to the number of hours in a year (8760 hours) divided by the annual failure rate.

In some cases these estimates are made in advance of the actual product completion leading to a theoretical MTBF. The operational MTBF is then later based on actual field returns.

MTBF is therefore a useful measurement of reliability. On a large network, this MTBF estimate can then be used to guide designers or management in deciding on the number of technicians or engineers needed to reliably maintain a network. A service level agreement between the service provider and the user dictates how reliable the network should be. Low reliability ultimately lowers the throughput of the network.

Since MTBF assumes that a product can be repaired to resume normal operations, the response time of technical staff plays an important role. By lowering the response time of technical staff, the downtime for a particular fault can be reduced leading to increased throughput for the system. If the failures occur too often or even simultaneously at different sites, then the response time can only be decreased by increasing the staff complement.

If the key objective is to maintain the service level agreement, then a balance need to be found between the number of technical staff needed in relation to the number of serviceable units.

The next section describes how the SDH simulator can be adapted and used to solve such a statistical problem.

5.4.2. Adaptation of SDH architecture

For this solution, each of the 63 VC12 ports represents a serviceable unit and has been assigned a probability of error or failure. Once a port is in the failed condition no traffic will be able to flow through the converter. The converter will remain in that failed condition for a specified time that will relate to the response time. At the end of this period, the circuit will restore automatically resulting in normal traffic flow.

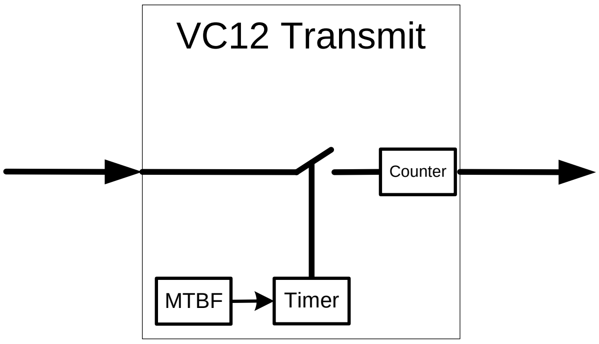


Figure 5.15: VC12 transmit converter block simulating the MTBF and response time of a VC12 circuit.

Figure 5.15 shows the VC12 transmit converter block. In the figure three functional unit blocks are labelled as MTBF, timer and counter. These three functional units have been added to the VC12 transmit converter block and their functions are as follows:

MTBF:

This functional unit is executed every time the VC12 transmit converter block is executed. It initiates the fail state of the converter block. A threshold value for the probability of error is set for the MTBF functional block and is compared against a random variable generated for each execution of the VC12 transmit converter block. If this random variable crosses the threshold value then the Timer is set, which in turn acts as an open switch by inhibiting the transmission of data via the output port.

Timer:

The MTBF functional unit initialises the timer. Once initialised the timer will start a countdown. The duration of this countdown can be set via an attribute. This countdown is used to simulate the response and restore time. The response time will be the time it takes a maintenance person to act upon and restore a faulty condition.

Counter:

The counter has the simple function of counting the transmitted bytes. This counter value is stored. A total count is taken at regular intervals over all the stored counter values of all the VC12 transmit converter blocks. This total count value is then used to indicate the throughput of the network at that particular time.

By reducing the response time to a fault, the network throughput can be maintained at an acceptable level. This acceptable level is usually an agreed upon value stipulated in the service level agreement.

If multiple faults occur simultaneously, then the response time will only be reduced by increasing the number of maintenance personnel, since faults now have to be fixed in parallel. Too few maintenance personnel will result in unacceptable response times, leading to severe degradation in the system throughput. Too many personnel, and the cost of running the network will become too high. A critical value for the staff compliment need to be found.

This simulation serves a better purpose for larger networks involving multiple network elements. For this implementation the main purpose of this simulation is to show that the SDH simulator can be used for a wider range of applications.

The results of this simulation are discussed in Chapter 6.

5.5. In Summary

In this chapter the architecture and functional implementation of the SDH system was discussed. The architecture was developed first followed by the functional implementation of the SDH overhead. A hierarchical approach was used, since the SDH architecture is hierarchical. This allowed one level to be developed and tested completely before developing the next level. It was also shown that the SDH architecture can further be used for statistical applications to aid network designers in resource management. In the next chapter the methods used for testing this system are discussed and the results shown. An analysis is also given on the performance of the SDR architecture for a large scale application such as SDH.

CHAPTER 6:
MEASUREMENT AND TESTING

6.1.Introduction

This chapter deals with the functional testing and analysis of some of the more important features used in SDH. The complete SDH system implemented on the SDR architecture is also analysed and a discussion is given on the overall performance of the implementation.

6.2.Physical Layer testing: OUT OF SERVICE TEST

This section of the test comprises the end to end testing of what would be known as the physical layer in a real world system. The SDH architecture was described in chapter 5, where the full multiplexing structure was systematically developed. Various converter blocks were linked together to ultimately form the SDH multiplexer. Exact frame lengths were built in each converter block, and a series of byte interleaving multiplexing stages were introduced. Figure 6.1 shows a high level view of the SDH multiplexer that was developed. The figure emphasises the test setup rather than the actual SDH architecture. What is visible is the 63 transmit and receive ports on MUX A, and the 63 transmit and receive ports for MUX B.

The SDR subcontroller manages the developed converter blocks representing various stages and levels within the SDH multiplexer. The subcontroller manages the transfer of information between the converter blocks, and ensures that each converter block gets a chance to execute its functional code. Both MUX A and MUX B are terminal units with no regeneration taking place in the channels between the forward and return paths.

The edge of the physical layer allows for access to 63 ports. Each port represents a 2Mbit/s connection and each connection follows a path through the originating transmit MUX, via the channel, and then into the receiving MUX. For this test the channel has no effect on the transported bytes.

The developed architecture has 63 VC12 virtual paths in the transmit direction, and 63 VC12 virtual path in the receive direction. A total of 126 VC12 paths are formed. The test set up in figure 6.1 was developed to test these paths for continuity over the testing period.

6.2.1. Test set up.

Converter block description.

charGen:

The converter block labelled as *charGen* generates random characters to represent 8-bit codes that could potentially be seen as sampled data. Each character is transmitted via two output ports. One of these ports is connected to the input port of the MUX and the other port is connected to the input port of the converter block labelled as *errCnt*. A frame of 32 random characters is transmitted at a time, to represent the 32 channels of an E1 frame that would normally be connected to the port. *CharGen* continually generates this frame until the process is stopped.

errCnt:

The converter block labelled as *errCnt* is responsible for comparing the generated frames received from *charGen* with the frames that propagated transmit and receive paths of the SDH MUX. *ErrCnt* has two input ports, and buffers the frame received from *charGen* while waiting for the other frame to propagate the complete transmit and receive paths. Each character of the 32-byte frame is then compared, and any discrepancy is then counted as number of errors per frame.

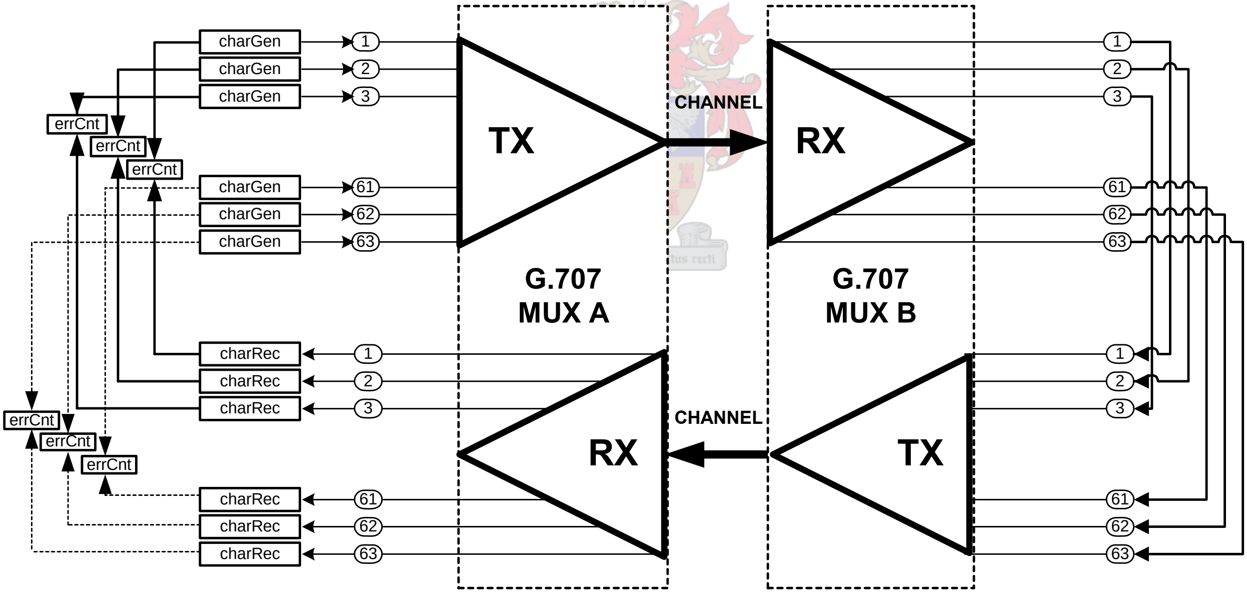


Figure 6.1: SDH Physical Layer

- A random byte generator is used to generate the input signal to the SDH system.
- The edge of the physical layer allows for access to 63 ports.
- The configuration of the set up is such that a loop back is used at the destination to return the transmitted bytes back to the originating MUX where detection for errors is performed.
- No BIT errors are added to the channel.

6.2.2. Expected results

The test is designed to test not only the SDH architecture but also the capability of the SDR architecture to handle this large-scale application. This means that a detected error could either have its origin in the SDH multiplexer or the SDR architecture.

The primary assessment being that of testing the SDH architecture in terms of the structure and in terms of whether the individual converter blocks are performing any erroneous management of the data between the time the data enters the converter block to the time the data is presented to the output port of the converter. It also tests for end-to-end connectivity of the virtual paths. A total of 1500 converter blocks have been strung together in a strictly defined order, with each converter block having strictly defined functions. Flaws in the implementation of the system in terms of how these blocks were linked together to form the SDH network can lead to errors in the received data or even a system failure. Flaws in the functional implementation of the converter blocks can lead to unexpected manipulation of the payload leading to erroneous information transfer.

The secondary assessment deals with the SDR architecture. A key feature of the SDR architecture is the management of the links between the converter blocks and the handling of the data between the input and output ports of adjacently connected converter blocks. The subcontroller has the responsibility of knowing how the converter blocks have been strung together and the type of data that will be passed from one converter to another. Flaws in the design of the SDR architecture in terms of the transfer of data between converter blocks will lead to erroneous data propagation through the system. This test stresses the ability of the subcontroller to manage large converter counts with large volumes of information transfer over a set period.

Since each layer of the architecture of the SDH multiplexer was tested during the development phase, and because no channel noise is introduced for the testing period, error free transmission is expected during the testing period.

6.2.3. Results

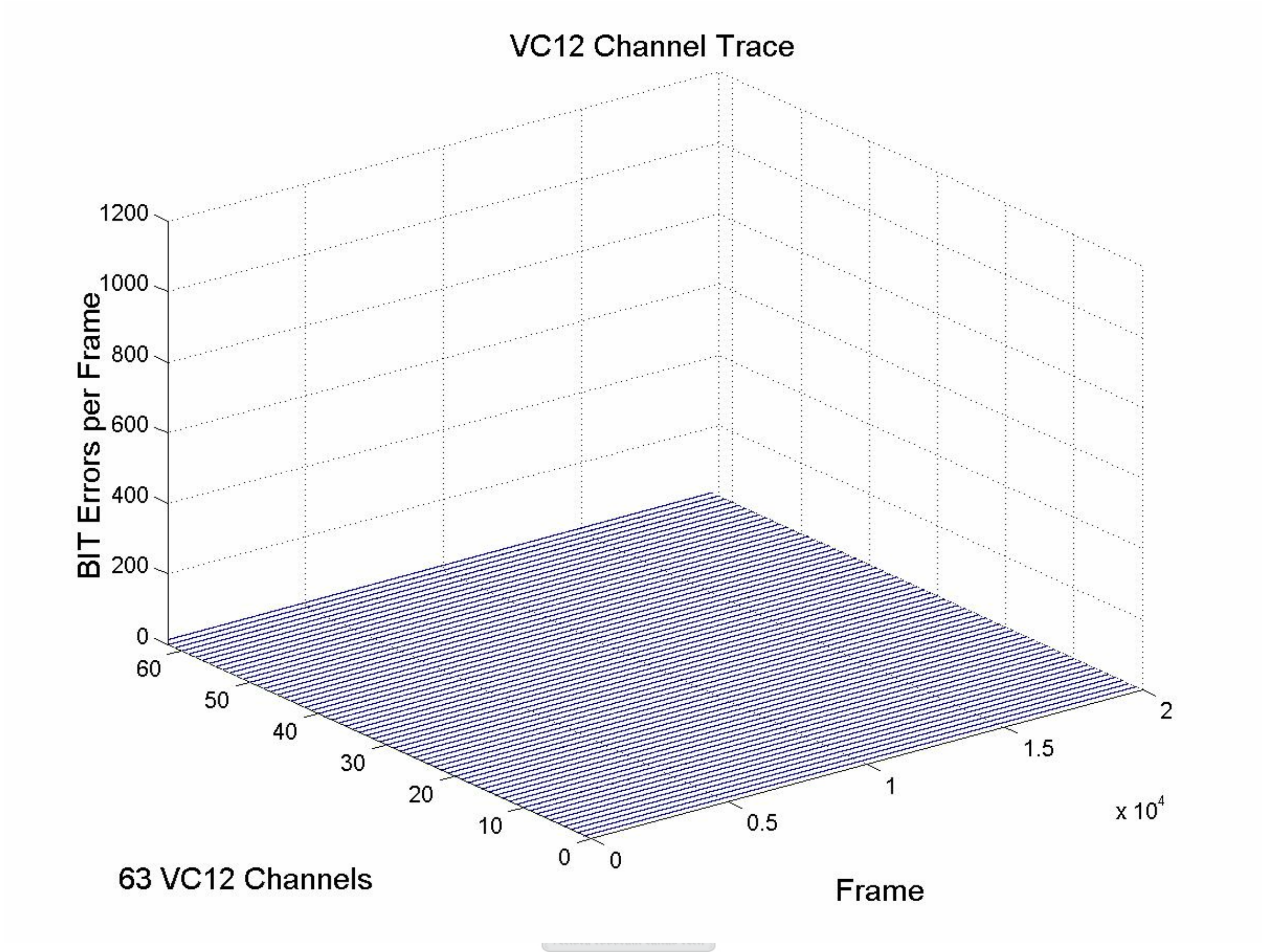


Figure 6.2: VC12 Channel trace of 63 virtual channels showing zero BIT Errors per Frame for the test period

6.2.4. Discussion

Figure 6.2 shows a plot of the results obtained for the test shown in figure 6.1. The error traces of 63 VC12 channels are shown. Each VC12 channel shown in the figure represents the error count per transmit and receive path, since each port at the far end of the SDH MUX was looped back to the transmit port. As shown in figure 6.2, no errors were detected for the duration of the test.

This result suggests that the 63 transmit virtual paths and the 63 receive virtual paths in the SDH MUX, and therefore the SDH MUX itself has been developed correctly. Another deduction from the results is that the SDR architecture coped well with the large implementation of converter blocks and high data flow. This test stressed the capability of the single subcontroller to manage the data transfer between the converter blocks.

One weakness of this test is that the two developed converter blocks, *charGen* and *errCnt* discussed at the beginning of this section, are used to test the system. These converter blocks in essence are subject to the same flaws as the converter blocks under test, and could even be a source of errors. With this is mind special care was taken when designing these converter blocks. These blocks have reduced functionality and have been well tested to increase confidence that these testing blocks will not interfere with the test results.

A better and more accurate test would have been to interface an independent pseudorandom test signal sequence to the input port of each of the VC12 channels. The receiver would then compare the received bits with a stored replica. Any detected errors would then clearly indicate a fault in either the SDH MUX or the SDR architecture.

6.3. Unit test of BIP-2 functionality in the V5 byte. IN SERVICE TEST

Chapter 5 discussed the capability of SDH to perform in-band error monitoring of each of the 63 VC12 virtual paths. The bit interleave parity code recommended for SDH is evaluated, in particular the BIP-2 code carried in the V5 path overhead byte. The BER monitoring function in SDH systems is based on violation detection of the coding rules concerned with the VC12 lower order path.

The end-to-end performance objectives per digital path are defined in G.826. Compliance with these path performance specifications ensures that a client 64kb/s connection to the system will meet its requirements. These end-to-end values apply to a 27500 km Hypothetical Reference Path that may include optical fibre, digital radio relay, metallic cable and satellite transmission systems [20]. These parameters are block-based. Each block is monitored by means of an error detection code. Since it is not possible to determine whether a block or its controlling error detection bits are in error, it is always assumed that the controlled block is in error.

6.3.1. Test set up:

CHANNEL:

The converter block labelled as *CHANNEL* in Figure 6.3 is used to model the channel between the VC12 transmit and receive converter block. As the probability of error is increased the error rate increases producing bit flips in the bytes propagating along the *CHANNEL*. The channel noise had a Gaussian distribution. Although only a single VC12t to VC12r path is shown in figure 6.3, the set up represents the test that was done on all 63 VC12 virtual paths.

charGen

The converter block labelled as *charGen* was described previously in 6.2.1.

charRec:

The converter block labelled as *charRec* performs the simple function of discarding of the received bytes. No other processing is performed in this converter block.

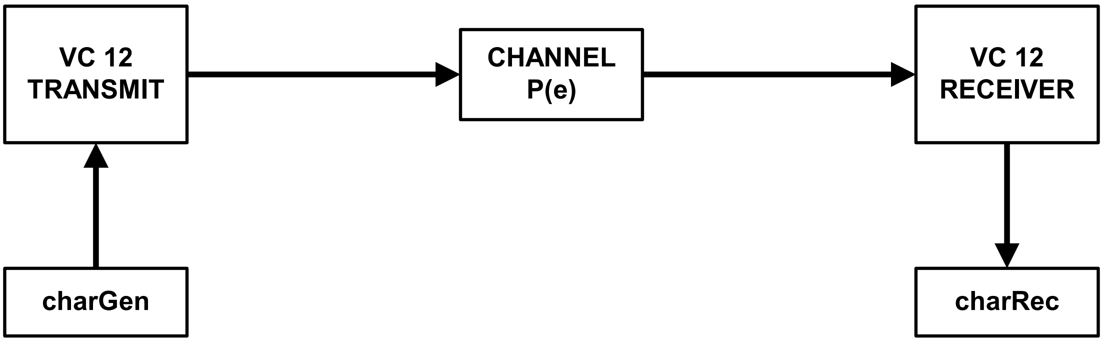


Figure 6.3:VC12 in service monitoring

- A random byte generator is used to generate the input signal to the VC12 path.
- The VC12 transmit converter block is linked to the VC12 receive converter block via a channel with a specified probability of error. This probability will be fixed over a particular VC12 block of data, and will only be incremented at the beginning of a full block of VC12 data.
- The accuracy of the inherent BIP-2 error detection code at the VC12 level will then be assessed by comparing the inherent error detection code with that of the known errors generated in the CHANNEL converter block.
- A Matlab simulation for the same EDC will also be produced and compared with that of the system.

6.3.2. Expected results

The BIP- (n, m) code requires the input sequence of binary symbols to be broken into m consecutive fragments of n consecutive symbols each. These m fragments and n symbols are represented by a n x m array $[x_{ij}]$ (i = 1, 2, …, n: j=1, 2, ..., m) [21]. A BIP- (2,140) code is used for the VC12 case.

At VC12 level, the block size has been defined as 1120bits [20], using a BIP-2 error detection code. Thus two bits are used to monitor 140 bytes or 1120 bits.

The BIP 2 code can detect between 0-2 defects per 140-byte frame. The mathematical upper limit for a frame of length 140 bytes or 1120 bits is reached for a probability of error equalling $10^{-2.7}$. The BIP-2 EDC will not track errors beyond this value.

6.3.3. Results

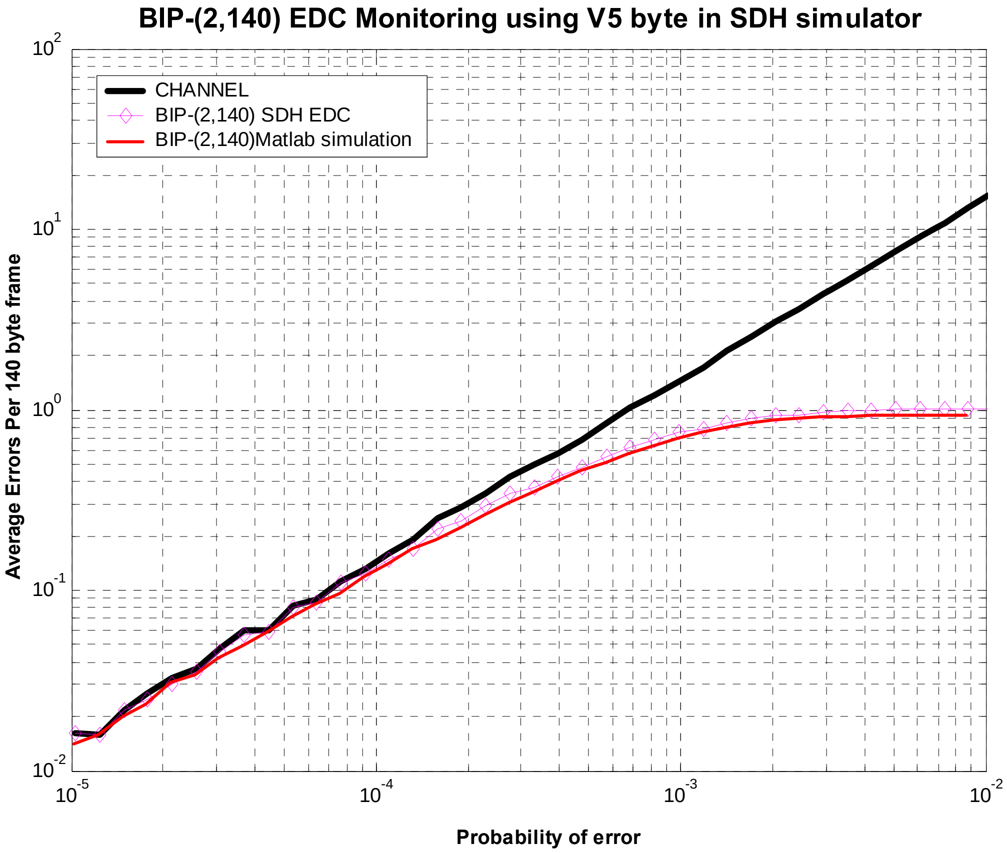


Figure 6.4: BIP – (2,140) error monitoring results using V5 byte in SDH simulator.

6.3.4. Discussion

Figure 6.4 shows how the results of the BIP – (2,140) EDC using the V5 byte in the SDH simulator. The result for a BIP – (2,140) EDC simulation using Matlab is also shown.

In both cases tracking of the channel errors for the probability of error less than $10^{-3.9}$ is very accurate. There are occasions within this region where the tracking deviates slightly from the channels errors, but these errors could result from undetected even errors. The inability of the EDC to track even errors is a limitation of the BIP – (2,140) EDC.

By further increasing the channel noise, the accuracy of the BIP- (2,140) EDC to track the channel errors is diminished to a point where the EDC code is totally insufficient to detect the generated channel errors. This limitation can be overcome by increasing the number of bits used to track the block.

The limit of the EDC is reached for a probability of error value of $10^{-2.7}$. This result compares well with the Matlab simulation of the same BIP – (2,140) EDC and also the mathematical limit mentioned in 6.3.2.

6.4. Unit test of BIP-8 functionality in the B3 byte

The bit interleaved parity codes recommended for SDH is evaluated in particular the BIP- (8,2349) code carried in the B3 path overhead byte. The BER monitoring function in SDH systems is based on violation detection of the coding rules concerned with VC4 higher order path.

6.4.1. Test set up

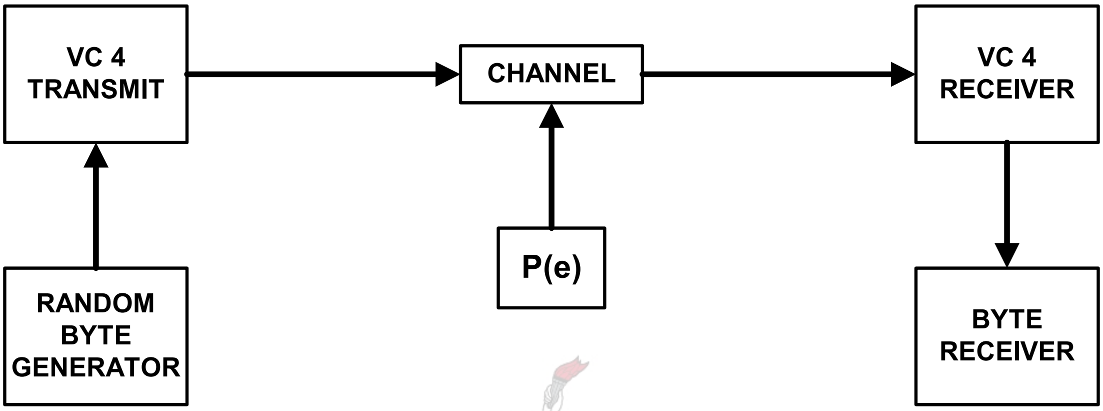


Figure 6.9: VC4 in service monitoring

- A random byte generator will be used as the input to the VC 4 path.
- The VC4 transmit converter block will be linked to the VC4 receive converter block via channel with a specified probability of error.
- The inherent BIP- (8, 2349) EDC will then be assessed.
- A Matlab simulation for the same EDC will also be produced and compared with that of the system.

6.4.2. Expected results

The block size at VC4 level has been defined as 2349 bytes or 18 792 bits using an inherent BIP- (8,2349) EDC. The VC4 has a bit rate of 150 336 kbit/s.

The BIP 8 code can detect between 0 – 8 defects per 2349 byte frame. The B3 byte is used to store the BIP- (8,2349) EDC.

The mathematical upper limit of the BIP-8 EDC for a frame of length 2349 bytes or 18 792 bits is reached for a probability of error equalling $10^{-3.37}$. The BIP-8 EDC will not track errors beyond this value.

6.4.3. Results

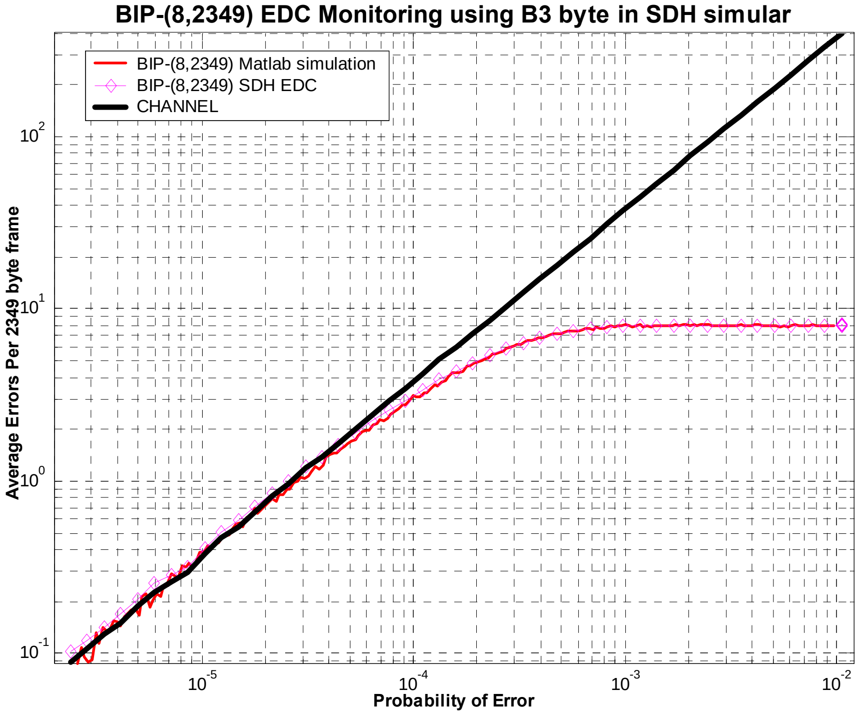


Figure 6.10: BIP – (8,2349) error monitoring results using B3 byte in SDH simulator.

6.4.4. Discussion

Figure 6.10 shows the results of the BIP – (8,2349) EDC using the B3 byte and the BIP – (8,2349) Matlab simulation. Both the B3 byte and the Matlab simulation tracks the channels errors for the lower probability of error values and reaches a limit for probability of error values greater than $10^{-3.3}$.

For the SDH simulator, four bits of the G3 byte are used as a count for the violations detected in the VC4 frame. The violations are detected using the B3 byte, and a count of these violations is then reported using four bits in the G3 byte. The count has nine legal values, namely 0-8 errors. The remaining seven possible values represented by these four bits are interpreted as zero errors. This is a stipulation in the G.707 [1] recommendation.

The theoretical upper limit for the BIP- (8,2349) EDC of $10^{-3.37}$ compares well with the Matlab and SDH EDC probability of error upper limit value of $10^{-3.3}$. For probability of error values lower than $10^{-3.3}$, the tracking of the channel errors by the B3 byte is similar to that of the Matlab simulation.

This test shows that the inherent EDC using the B3 byte for the VC4 higher order path performs well compared to the Matlab simulation and the theoretical upper limit of the BIP-8 EDC for a frame of length 2349 bytes.

6.5. SYSTEM THROUGHPUT

The purpose of this test is to find the throughput of the SDH system implemented on the SDR architecture. The test is performed for various segments of the SDH network and the results plotted.

1. Test set up

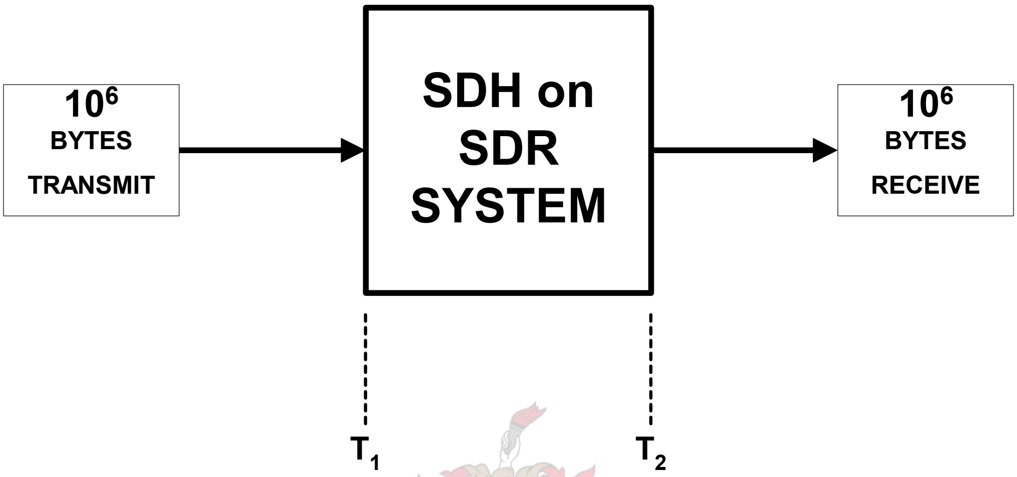


Figure 6.12: System throughput measurement

- The converter block marked as *10^6 bytes transmit* transmits 10^6 bytes to the SDH network.
- Once the first byte is received at the first converter block in the SDH network, the time is recorded.
- All 10^6 bytes then propagate through the SDH network, and once the last byte of the 10^6 exits the last converter block of the SDH network the time is recorded.
- The difference in the two times then indicates the time it takes for 10^6 bytes to propagate through the system.
- This experiment is repeated at various segments of the SDH network.

2. Expected results

The throughput of the system is affected by the length of the system and the processing that takes place in each converter block. Since the SDH network is a series of converter blocks linked together, the throughput will drop if the processing power remains constant.

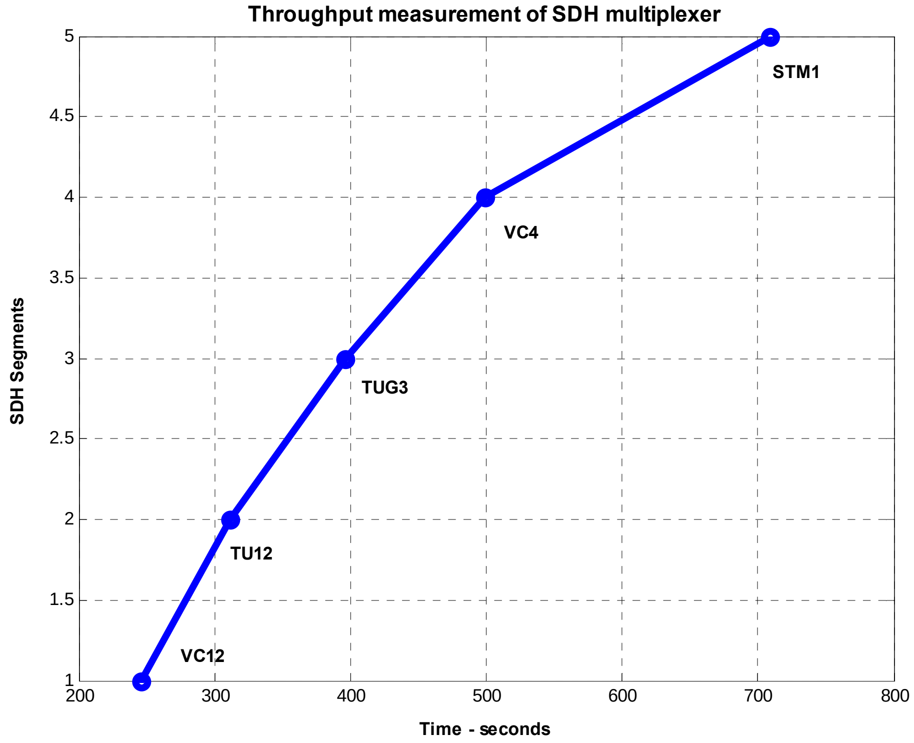


Figure 6.13: Throughput at various SDH segments.

3. Results

Figure 6.13 shows the trace for the throughput of the SDH network. The throughput was measured at various segments of the SDH network. The levels indicated as VC12 and TU12 forms part of the lower order path of the SDH network. TUG3 is the output of the multiplexing stage with in the SDH structure. The VC4 marking indicates the higher order path with STM-1 the full SDH capacity.

4. Discussion

As expected, for an increase in network complexity, the throughput of the system decreases. The results also show that the throughput is most severely affected by the STM1 segment. At the STM1 level, double the time is needed to transmit the same amount of bits as for any other stage. These two segments then also process the full capacity of the 63 VC12 containers, which explains the increase in processing time.

By focussing on the VC4 and STM1 segments, one can also concentrate on writing more efficient code to further decrease the processing times of these segments.

6.6. System Latency

Latency relates to the time it takes for information to travel through the system. The test is performed at various levels within the SDH architecture.

6.6.1. Test set up

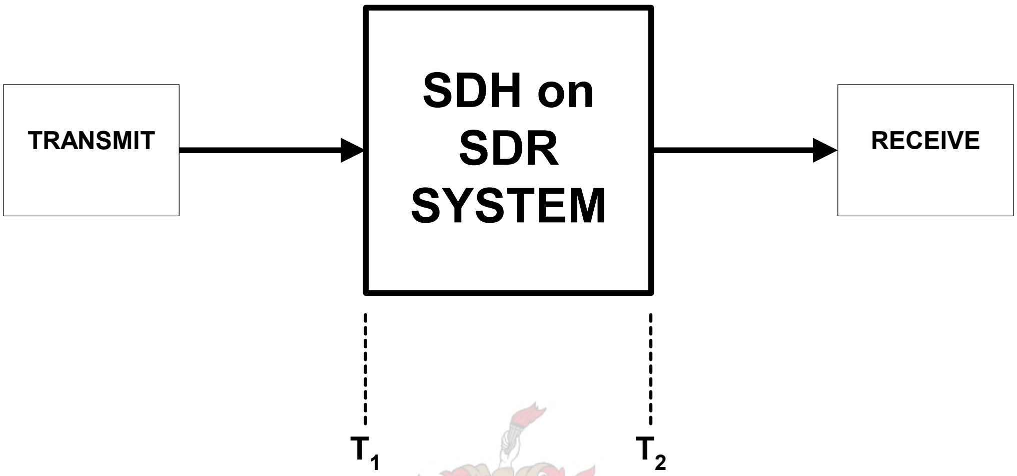


Figure 6.14: System Latency

- The converter block marked as *Transmit* transmits 10^6 bytes to the SDH network.
- Once the first byte is received at the first converter block in the SDH network, the time is recorded.
- All 10^6 bytes then propagate through the SDH network, and once the first byte of the 10^6 bytes exits the last converter block of the SDH network the time is recorded.
- The difference in the two times then indicates the time it takes for 1 byte of data to propagate through the system.
- This experiment is repeated at various segments of the SDH network.

6.6.2. Expected results

The latency test is done at various segments of the SDH network. Each segment increases the length of the SDH network.

As the length of the system is increased one would expect an increase in system latency. Another factor influencing system latency is the processing complexity of each level.

6.6.3. Results

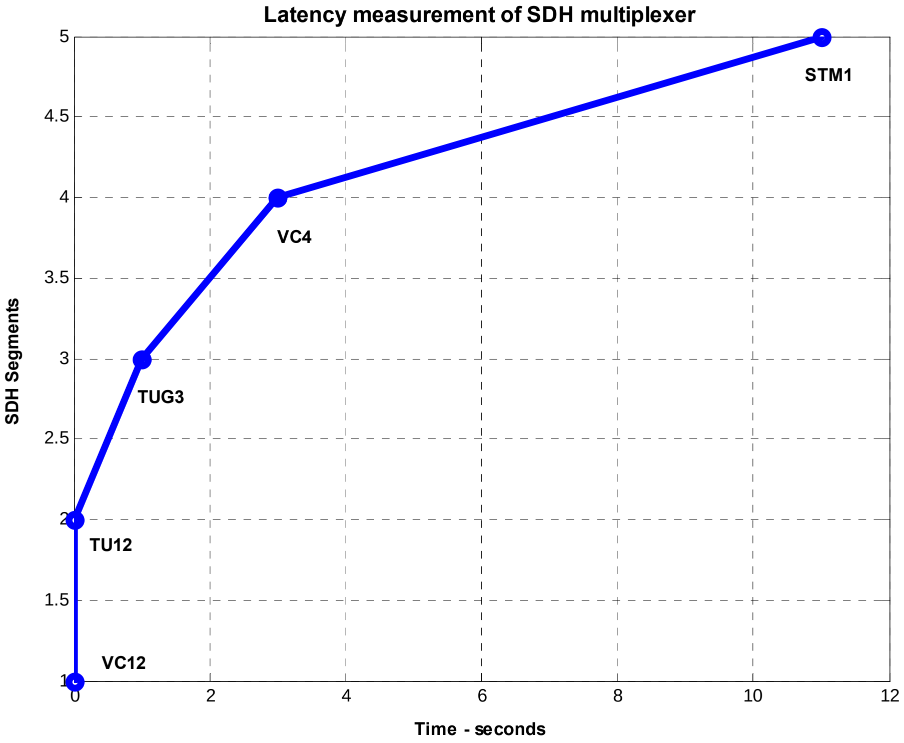


Figure 6.15: Latency at various stages of the SDH network

Figure 6.15 shows the trace for the latency of the SHD network. The latency of the SDH network was measured at various segments of the SDH network. The levels indicated as VC12 and TU12 forms part of the lower order path of the SDH network. TUG3 is the output of the multiplexing stage within the SDH structure. The VC4 marking indicates the higher order path with STM-1 the full SDH capacity.

6.6.4. Discussion

Figure 6.15 shows the system latency at various SDH stages. The greatest delays were introduced at the VC4 and STM1 levels. Here two factors influence the increase in latency, one of which is the length of the system, and the second the processing complexity of the VC4 and STM1 converter blocks.

6.7. System performance evaluation

A freely available profiling tool name *gprof*, written by Jay Fenlason, was used to collect profiling information for the SDH implementation to determine which parts of the SDH implementation took most of the execution time and also the structure of the function calls. The profiling tool collected information of not only the SDH processes, but also the SDR processes.

6.7.1. Test set up

- All code need to be compiled with profiling enabled. This is done by adding the *–pg* switch when compiling the SDH converters.
- For the GUI application QT designer was used. The *–pg* switch must be set before running *make*.

6.7.2. Results and discussion

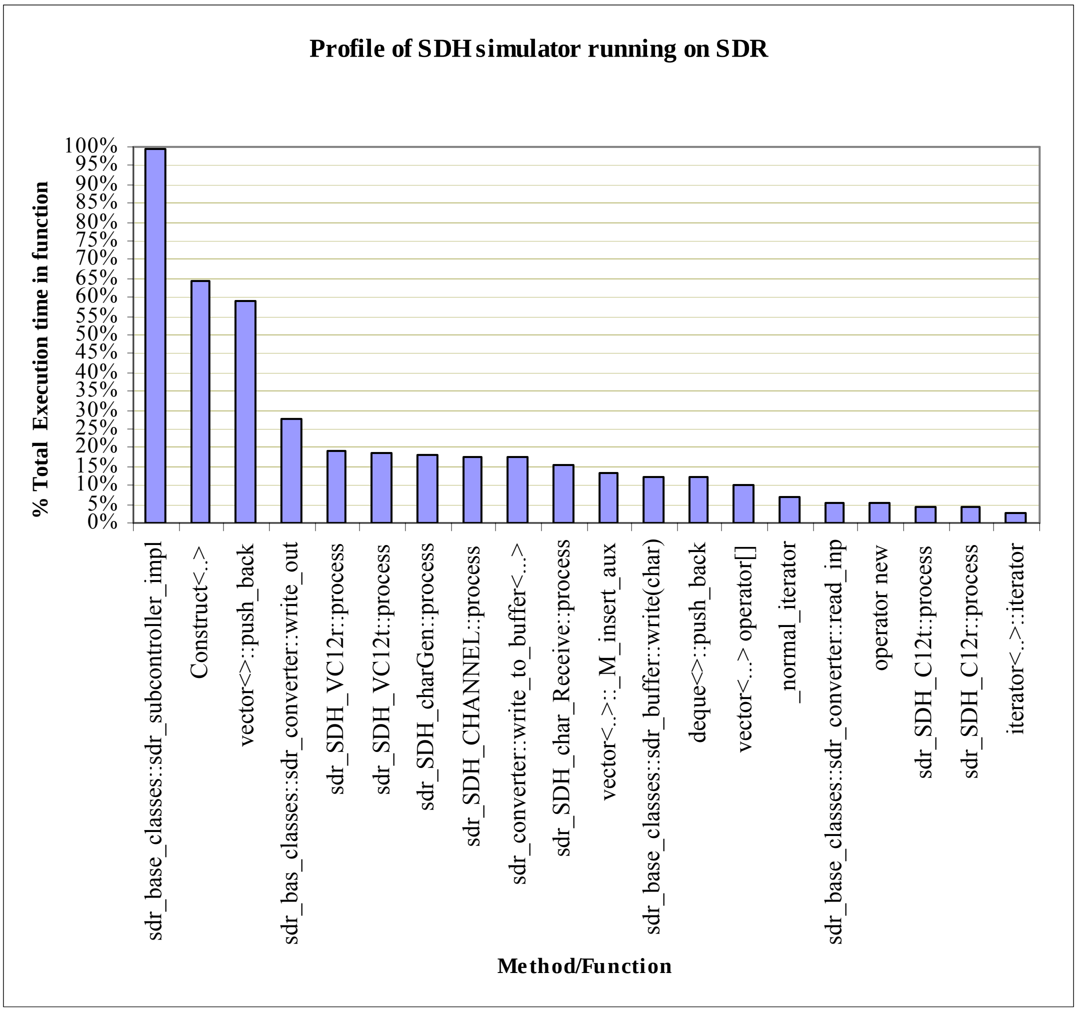
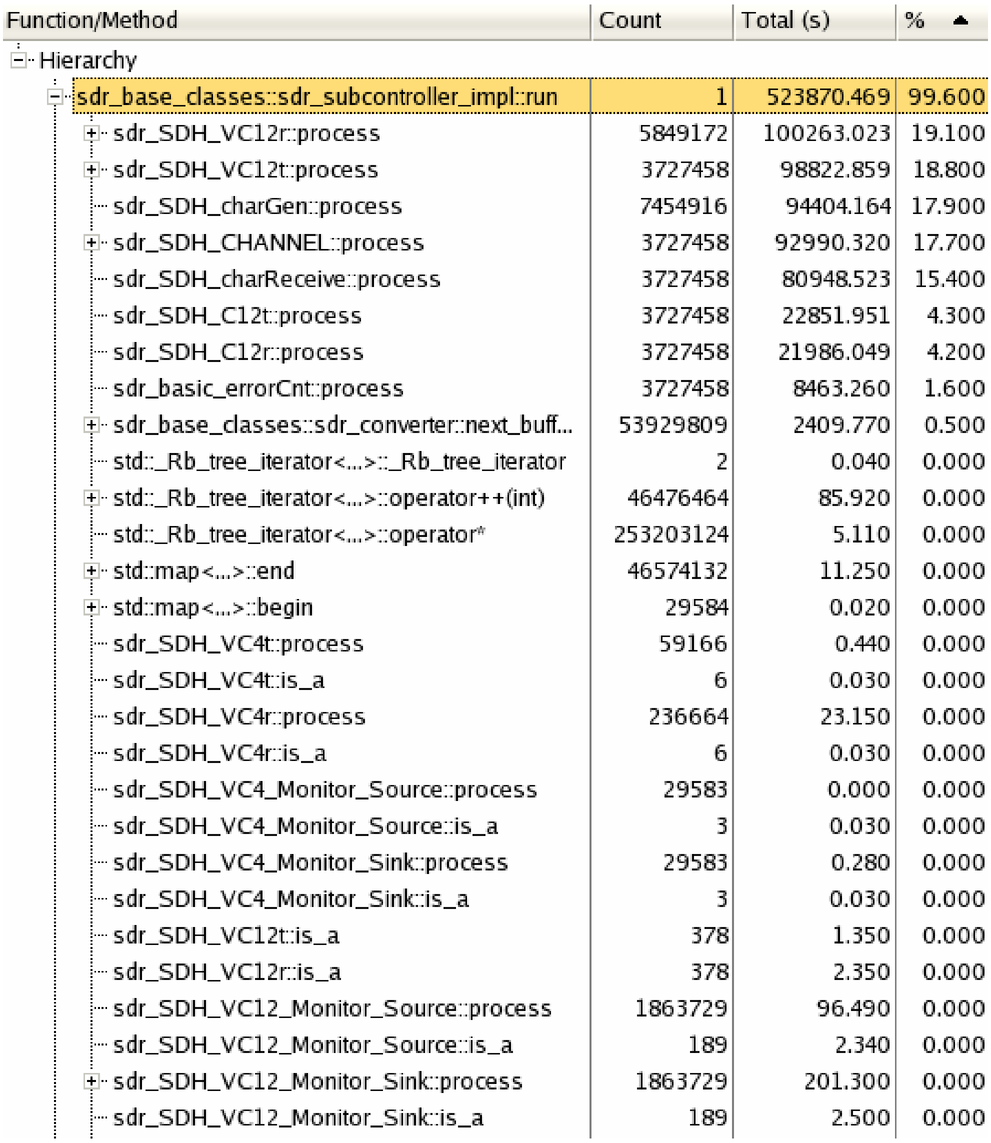


Figure 6.16: Profile of SDH implementation on the SDR platform

Figure 6.16 shows the profiling information that was extracted from a flat file generated by the *gprof* software. The percentage total execution times of each function are shown. Only those functions with a percentage total execution time greater than 2% for both the SDR classes and the SDH classes are shown. From this one can easily compare the execution times of these two systems.

A clear leader in terms of percentage execution time is the sdr_base_classess::sdr_subcontroller_impl at 99.6%. This is expected since the subcontroller has all the SDH converter blocks as its children. The expression ‘children’ or ‘child’ refers to the subroutines called during the execution of a particular function. Figure 6.17 shows a screenshot of a software tool called *kprof* that was used to interpret the *gprof* output file.



Function/Method	Count	Total (s)	% ▲
Hierarchy			
sdr_base_classes::sdr_subcontroller_impl::run	1	523870.469	99.600
sdr_SDH_VC12r::process	5849172	100263.023	19.100
sdr_SDH_VC12t::process	3727458	98822.859	18.800
sdr_SDH_charGen::process	7454916	94404.164	17.900
sdr_SDH_CHANNEL::process	3727458	92990.320	17.700
sdr_SDH_charReceive::process	3727458	80948.523	15.400
sdr_SDH_C12t::process	3727458	22851.951	4.300
sdr_SDH_C12r::process	3727458	21986.049	4.200
sdr_basic_errorCnt::process	3727458	8463.260	1.600
sdr_base_classes::sdr_converter::next_buff...	53929809	2409.770	0.500
std::_Rb_tree_iterator<...>::_Rb_tree_iterator	2	0.040	0.000
std::_Rb_tree_iterator<...>::operator++(int)	46476464	85.920	0.000
std::_Rb_tree_iterator<...>::operator*	253203124	5.110	0.000
std::map<...>::end	46574132	11.250	0.000
std::map<...>::begin	29584	0.020	0.000
sdr_SDH_VC4t::process	59166	0.440	0.000
sdr_SDH_VC4t::is_a	6	0.030	0.000
sdr_SDH_VC4r::process	236664	23.150	0.000
sdr_SDH_VC4r::is_a	6	0.030	0.000
sdr_SDH_VC4_Monitor_Source::process	29583	0.000	0.000
sdr_SDH_VC4_Monitor_Source::is_a	3	0.030	0.000
sdr_SDH_VC4_Monitor_Sink::process	29583	0.280	0.000
sdr_SDH_VC4_Monitor_Sink::is_a	3	0.030	0.000
sdr_SDH_VC12t::is_a	378	1.350	0.000
sdr_SDH_VC12r::is_a	378	2.350	0.000
sdr_SDH_VC12_Monitor_Source::process	1863729	96.490	0.000
sdr_SDH_VC12_Monitor_Source::is_a	189	2.340	0.000
sdr_SDH_VC12_Monitor_Sink::process	1863729	201.300	0.000
sdr_SDH_VC12_Monitor_Sink::is_a	189	2.500	0.000

Figure 6.17: Screenshot of the sdr_subcontroller_impl with its child processes.

The child processes of the sdr_subcontroller_impl are all the developed SDH components. The percentage execution time of all these child processes totals the sdr_subcontroller_impl processing time. From figure 6.17 it is clear that there are five major contributors to the sdr_subcontroller_impl execution time. These processes are listed in table 6.1 with the processing times. Child processes with times less than 4.2% are not shown.

Child processes of sdr_subcontroller_impl	% Time
sdr_SDH_VC12r	19.10%
sdr_SDH_VC12t	18.80%
sdr_SDH_charGen	17.90%
sdr_SDH_CHANNEL	17.70%
sdr_SDH_charReceive	17.90%
sdr_SDH_C12t	4.30%
sdr_SDH_C12r	4.20%

Table 6.1: Child processes of sdr_subcontroller_impl

Each of these processes contributes between 17% and 19% of the execution time per listed process. The execution time contributions of the sdr_SDH_C12t and sdr_SDH_C12r converters will be omitted since their execution time contributions are relatively small.

By examining the function calls of the child processes shown in Table 6.1 the following was noted:

- Calls to vector<>::push_back had a major effect on the execution time of the child processes. Up to 59.1% of the sdr_SDH_VC12r execution time was due to this call. There were also a significantly high number of calls to this process.
- A lot of time was consumed in the processes that had to call the write_output_port function. This function writes data to the input buffer of the following subcontroller and for the sdr_SDH_VC12r converter contributed roughly 27.5% to the execution time of this converter.
- Another contributor was the read_input_port function. The read_input_port function however used much less execution time than the write_output_port function.
- A high amount of calls were made to set_attribute. Though the total percentage execution time for this call is relatively small.

Five major areas for code improvement have been identified. These processes have been listed in table 6.1 and form part of the SDH structure. These processes have high execution times mainly because of the time spent writing to the output ports.

The execution time of these calls can be improved by the following:

- ❑ Try and improving the write_output_port call in the SDR architecture.
- ❑ Reduce the number of write_out_port calls per sdr_subcontroller_impl.

The latter can be achieved quite easily by reducing the amount of atomic units for the full SDH system. By grouping together more functionality in the converter blocks per subcontroller, one can reduce the amount of time spent sending data from one converter block to another, and thereby reduce the amount of write_output_port calls.

The SDH architecture has been developed with an emphasis on atomic structures, since that design methodology increased the reusability of these units. For the SDR architecture, stringing together high numbers of these atomic units increases the write_output_port and read_input_port calls, which dramatically increases the execution time of these atomic units.

If we assume that the write_ouput_port is as efficient as it can be, then a balance need to be found between designing atomic units for increased reusability and that of implementing a system where reducing the number of calls to the write_output_port and read_input_port can lead to a faster system.

Increasing the functionality per converter block has the burden of complexity and while reducing component reusability, also increases debugging time.

6.8. MTBF evaluation

MTBF was discussed in 5.4.1 of chapter 5. Here the capability to use the simulator as a planning aid is demonstrated, to show that the simulator can also be used for applications other than training.

6.8.1. Test set up

- All 63 VC12 virtual paths were assigned a probability of failure.
- Once failed, each path was assigned an automatic recovery period.
- The total throughput of the system was evaluated constantly.
- The automatic recovery period was shortened after specific time intervals.
- The current throughput was evaluated against the service level agreement.

6.8.2. Expected results

Initially one would expect that for longer recovery periods, the total throughput over a defined period would decrease. A too long recovery period would be further complicated by more circuits failing during that recovery period. The recovery period relates the response time of the technical staff to the fault at hand. For a case where there is understaffing, the response time for the second and third failure would be significantly long. This could lead to a severe drop in the total throughput of the system.

As the recovery period is decreased, the total throughput over a defined period should increase.

6.8.3. Results

Figure 6.18 shows the result of the MTBF simulation using the SDH simulator. Four traces are shown in the graph.

The dashed line indicates the STM-1 full capacity. This is the sum total of the 63 VC12 virtual path capacities and indicates the maximum throughput of the system for a zero failure state.

The dotted line indicates the service level agreement. For any service, a service level agreement governs the operational requirements of the system. These agreements specify downtime periods and average system throughput, usually as a percentage of the total throughput. The dotted line indicates a 90% throughput level.

The fluctuating solid line indicates the current throughput of the system, taking into account all circuits that might be in a fail state, whereas the chained trace forming a downward moving staircase indicates the recovery period of the circuit.

The throughput trace shows that for a too long recovery period, the throughput of the system can degrade to such a point where no communication is possible. This is a worst case scenario and indicates severe insufficiencies in the recovery period and ultimately the staffing requirements to maintain the network. The throughput levels for the first time interval falls way below the service level agreement.

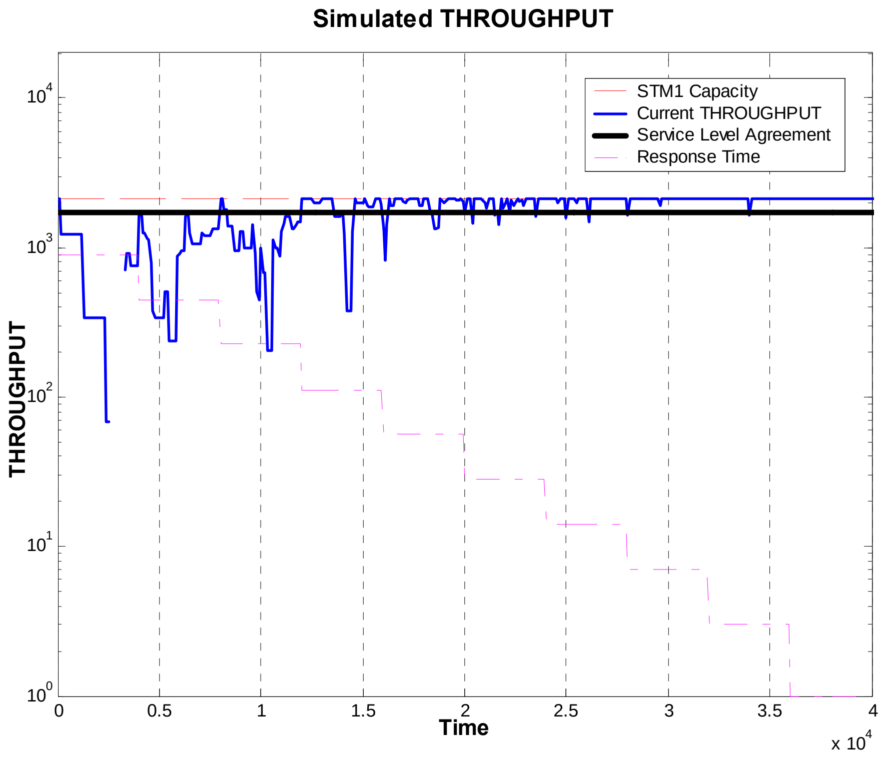


Figure 6.18: MTBF simulation using SDH simulator

As the recovery period or response time is decreased, there is an increase in the average throughput level over that period. By further decreasing the response time, a point is reached where circuit failures have a minimal effect on the system throughput, maintaining throughput levels higher than the specified service level agreement.

6.8.4. Discussion

For the simulation, the relationship between system throughput and response time over a period with a constant probability of circuit failure has been established. Throughput is shown to be inversely proportional to response time for a network having one or more fail states.

A too long response time is shown to have catastrophic results, while at the same time a very short response time can also not be humanly achievable because of factors like travel distance, fault location and isolation and actual repair time.

The region between times 1.5 and 2.5, indicate average throughput levels very close to the service level agreement throughput especially between times 2 and 2.5. This region and the associated response times could now be used as a guideline in determining the number of staff required to maintain the network to achieve the specified service level agreement throughput level.

6.9.In Summary

The lower order and higher order path was tested using in circuit monitoring. Here in particular the V5 byte was used to transport the BIP2 result calculated over the previous frame for all 63 VC12 virtual paths. The in circuit monitoring for the higher order path was also evaluated. Furthermore, the performance of the SDR architecture was also evaluated for the large scale SDH application where throughput and latency tests were performed. Finally the statistical tool was evaluated. The chapter also points out possible areas of improvement in the SDR architecture to improve system throughput. The next chapter gives some concluding remarks and areas where the SDH system can be further developed.

CHAPTER 7:
CONCLUSIONS

This chapter provides an overview of the work in the previous chapters, and highlights some strengths and weaknesses of the system.

7.1. Overview of the presented work.

7.1.1. SDH system:

The SDR-based SDH system was designed to, for the greater part, mimic the behaviour of a real-world SDH system particularly in terms of system monitoring. The vast amount of ITU-T recommendations that define SDH meant that only certain core functions of SDH could be implemented. In doing so, the complexity of the system had to be reduced to a representation that could be achievable given the time and resources at hand.

This led to a reduction in SDH functionally, in aspects such as network synchronisation, automatic link protection and concatenation of the SDH container were not considered. The reduced OAM&P functionality of the SMS provides a monitoring function, while provisioning is text based. The emphasis was placed on developing some of the core functionality of SDH and focusing on network monitoring. This was done in support of the idea that the SDH simulator should be a training tool, and that by monitoring faults for certain network conditions, insight could be gained into the working of SDH.

The SDH architecture providing a C12 mapping opportunity was implemented to STM1 level. This architecture then laid the foundation for the development of the functional units of SDH. These functional units were developed to the VC4 level. The MS and RS were not considered in terms of its functional units, but were included in the SDH multiplexing architecture to enable a full STM-1 capacity to be built from the various overhead added at the different levels of the SDH hierarchy. Since all the building blocks for the SDH multiplexer has been included in the design, any future functional development of these units will be trivial.

7.1.2. SDR system:

The system also demonstrates the use of SDR to effectively provide the communications handling of the SDH implementation. This was the secondary objective of this study, and examined the SDR architecture for a large-scale application. The SDH implementation was used to evaluate the SDR architecture for large applications, since only small converter counts and relatively low data flow implementations have thus far been considered.

A network segmentation approach was used to develop the SDH network, and once configured on the SDR architecture, each layer could be built and tested independently. This approach was very natural for both architectures, since SDH and SDR are layered architectures. Throughput, latency and profiling information provided insight into the working of the SDR architecture.

7.1.3. Contributions:

In the course of developing the SDH system, certain stumbling blocks were noted that hindered the development process and will be discussed next.

A shortcoming of the SDR architecture is the absence of a GUI to facilitate the converter linking process. Currently these converter blocks are linked using syntax defined for the SDR system. For larger systems, a visual representation of the system built with these converter blocks will aid development, since the port connections can now be followed visually. By hiding more of the SDR processes, the developer can focus more on his or her implementation.

Adding and removing components from the design also meant that the application had to quit and then recompiled in order for these components to form part of the architecture. A smoother approach would be a system where components can be added to the design, and a less invasive stop and restart command, without the need to recompile would then include these new containers into the architecture.

Once the converter blocks were linked a round robin scheduling mechanism was used to execute these functional units. This approach worked well, but there were some inefficiency since the scheduling mechanism would access converter blocks irrespective of whether the converter blocks had data at its input ports to process. Further development of the scheduling mechanism could lead to better efficiencies regarding converter execution.

Profiling the implementation of the large-scale SDH application revealed that the write_output_port method consumed a high percentage of execution time. The write_output_port method writes processed samples to the input port of the next converter. A more efficient write procedure would greatly reduce the execution time of the overall application. Another approach would be to reduce the number of write_output_port calls for the developed system such as the SDH system. This can be achieved by increasing functionality of the converter blocks where possible, in an effort to combine adjacent converter blocks. This design approach is a step back from the atomic model that is envisioned for the converter blocks, but can lead to a reduction in the total execution time spent on calling the write_output_port method. In the absence of a more efficient write_output_port method, a trade off between the number of converter blocks, and component complexity need to be found.

In addition to demonstrating the SDH simulator as a training tool, the simulator was also used to demonstrate its use as a statistical tool. By effectively modelling a telecommunications network the simulator could now perform functions to aid designers in resource management. This was achieved by adding statistical data to the converter blocks.

Together SDH and SDR provide an effective development and training tool as each converter block provides insight into the workings of an SDH network. Both architectures are hierarchical in design, which eased the implementation process.

7.2. Future work

The MS and RS could be implemented to complete the functional implementation of the SDH MUX. In addition to that, some other features like automatic protection switching, concatenation and configuration of the network via a GUI could also be considered.

SDH was optimised for voice transport only, which sets different requirements for the network than the growing data transport services. The technology enabling voice over data is called VoIP. Legacy SDH cannot support the growing data service demand. Next-generation SDH has a transport mechanism, which enables the concurrent existence of legacy and new services over the same network without disturbing each other. The technology is commonly known as the *Data over SDH* (DoS) concept. The DoS scheme consists of three technologies: *Generic Framing Procedure* (GFP), *Virtual Concatenation* (VC) and *Link Capacity Adjustment Scheme* (LCAS), all standardised by ITU-T.

The next step for this SDH implementation would be that of developing the DoS schemes to allow almost every data transport technology to be efficiently mapped over SDH.

Appendixes

Appendix A

This section on the SDH access point identification is from the G.831 recommendation section 3 and Appendix A1 [11].

SDH access point identification

An essential requirement for successful management of SDH networks incorporating features such as point-to-point and point-to-multipoint paths is a unique means of identifying significant points in the network, e.g. access points. The features of Access Point Identifiers (APIs) are:

- Each access point identifier must be globally unique in its layer network;
- Where it may be expected that the access point may be required for path set-up across an inter-operator boundary, the access point identifier must be available to other network operators;
- The access point identifier should not change while the access point remains in existence;
- The access point identifier should be able to identify the country and network operator which is responsible for routing to and from the access point;
- The set of all access point identifiers belonging to a single administrative layer network should form a single access point identification scheme;
- The scheme of access point identifiers for each administrative layer network can be independent from the scheme in any other administrative layer network.

It is recommended that the VC-11, VC-12, VC-2, VC-2-xc, VC-3, VC-4 and VC-4-xc should each have the access point identification scheme bases on a tree-like format to aid routing control search algorithms. The access point identifier should be globally unambiguous.

The API shall begin with either the country code as defined in ITU-T E.164 (one, two or three numeric characters) or, the three alphabetic character country code as defined in ISO 3166.

The remainder of the API characters that follow the country code shall be a matter for the organization to whom the country code has been assigned, provided that uniqueness is guaranteed.

These characters may be any alphanumeric characters as defined in ITU-T T.50 (International Reference Version – 7-bit coded character set for information interchange).
The alphanumeric character set consists of the characters "a" to "z", "A" to "Z", and "0" to "9".

In the case where the API uses less than fifteen characters, the API will be padded (extended) with the T.50 "NUL" character to get a fifteen byte character string.

For interworking with equipment developed prior to this version of the Recommendation that may use the T.50 space as a padding character, new equipment should be able to generate T.50 "SPACE" padding characters as an option.

A similar access and test point identification scheme is required for the section layer to support point-to-point and point-to-multipoint paths and wide area multiplexers as used in satellite sub networks.

The byte allocations for the transmission of the access point identifier at the section layer; higher-order path layer and lower-order path layer are given in ITU-T G.707.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Country Code		Sending Operator			Sending Town/Node			Receiving Town/Node			X	M.1400 serial number		
F1		F2			F3			F4			F5	F6		

Figure A1: Format of section or path identifier

F1 Country code

This field contains the fixed length two alpha character country code as defined in ISO 3166 (A2).

F2 Sending operator

A three-character alphanumeric identifier for the sending operator.

F3 Sending town/node

A three-character alpha-numeric identifier for the sending town or node. Defined by the sending operator.

F4 Receiving town/node

A three-character alphanumeric identifier for the receiving town or node. Defined by the receiving operator.

F5 X

An undefined alphanumeric character that the sending operator can use to guarantee uniqueness for any path identifier generated. When the use of this field is not necessary to guarantee uniqueness, an alphanumeric T.50 character should be used to pad this field within the path identifier.

Appendix B

Signal label C2. This byte indicates the composition or the maintenance status of the VC4/VC-3.

Hex Code	Interpretation of VC4-Xc / VC4 / VC3 Content Mapping
00	Unequipped or supervisory unequipped.
01	Reserved: Formerly "Equipment" non specific. "New receivers ignore 01 from old NE.
02	TUG structure: Mainly for "voice" tributaries.
03	Locked TU-n: For locked byte synch mode. (No longer supported)
04	Asynchronous mapping of E-3 or DS-3.
05	Mapping under development: An experimental Category
12	Asynchronous E-4 (139.264 Mbps) mapping.
13	ATM Cell mapping.
14	Metropolitan Area Network (MAN) Dist. Queue Dual Bus (DQDB) mapping.
15	Fiber Distributed Data Interface (FDDI) mapping
16	Mapping of HDLC/PPP framed signal.
17	Simple Data Link (SDL) for scrambler mapping.
18	HDKC/LAPS (Link access procedure - SDH) frame mapping (under study)
19	Simple Data Link (SDL) for scrambler mapping.
1A	10 Gb/s Ethernet frame mapping.
1B	Generic Packet Framing (GPF) mapping.
1C	Fibre Channel at 10Gb/s
CF	Reserved: Former value used HDLC/PPP frame mapping (obsolete)
D0-DF	Reserved for proprietary use.
E1-FC	Reserved for national use.
FE	Test signal.
FF	VC-AIS (Generated by tandem connection source if there is no valid incoming signal)

Table B.1: C2 byte coding

Appendix C

Signal label coding of bits 5 through 7 of the V5 byte.

V5			Interpretation
B5	B6	B7	VC-2 /VC-1 application
0	0	0	Unequipped or supervisory unequipped
0	0	1	Reserved: Formerly known "as equipment non specific" New equipment ignores 001 from old Nes.
0	1	0	Asynchronous mapping.
0	1	1	Bit synchronous E-1 (2.048 Mbps) VC-12 mapping.
1	0	0	Byte synchronous mapping
1	0	1	Extended signal label: Used with K4 byte to provide mapping details at the LO level.
1	1	0	Test signal: For any non-virtual concatenated payload.
1	1	1	VC-AIS: Generated by the tandem connection source if there is no valid incoming signal.

Table C.1: VC-1/ VC-2 Extended Signal Label byte coding

Appendix D

Byte structure of the N2 network operator byte.

	BIP-2		"1"	AIS	TC-REI	OEI	TC-Apid, TC-RDI,	
N2	1	2	3	4	5	6	7	8

Figure D.1: N2 Byte structure

- N2 [1-2]: Used as an even BIP-2 for the tandem connection
- N2 [3]: Fixed to 1. This is to guarantee that the contents of N2 are not all zeros at the TC-source.
- N2 [4]: Operates as an incoming AIS signal.
- N2 [5]: Operates as the TC-REI of the Tandem Connection to indicate errored blocks caused with the Tandem Connection.
- N2 [6]: Operates as the OEI to indicate errored blocks of the aggressing VC-n.
- N2 [7-8]: Operates in a 76 multiframe as:
  - The access point identifier of the Tandem Connection.
  - TC-RDI – Indicating to the remote end that defects have been detected within the Tandem Connection at the near end Tandem Connection sink.
  - The ODI, indicating to the far end that TU-AIS has been inserted at the TC-sink into the regressing TU-n due to detects before or within the Tandem Connection.
  - Reserved capacity.

Appendix E

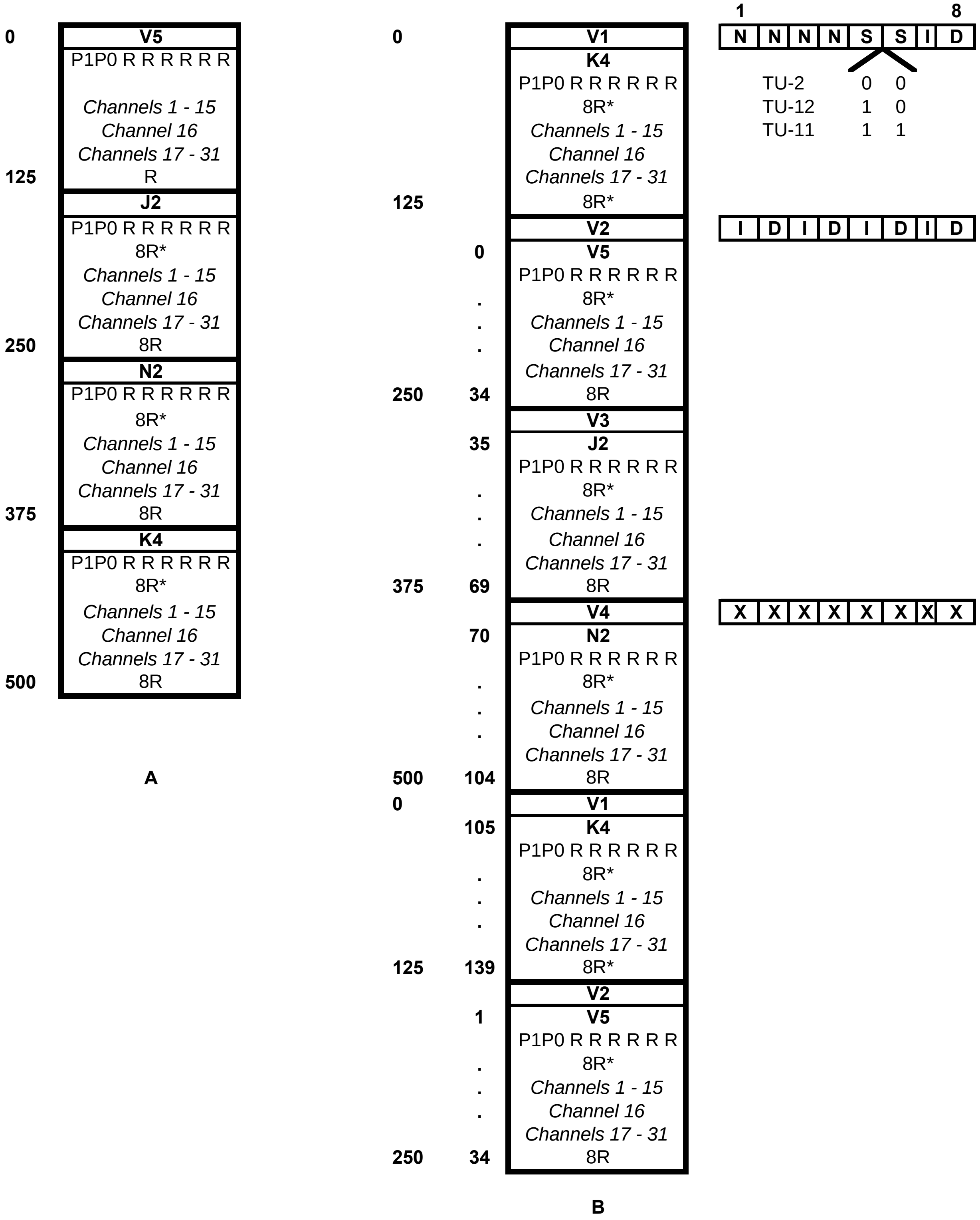
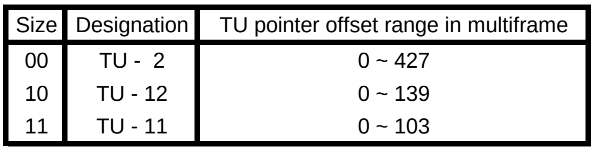


Figure E.1: TU-1 Payload Offset.

Pointer functions.

The first four bits of the pointer word carry the *New Data Flag* (NDF) information. The NDF provides a method of changing the value of the pointer if that change is due to a change in the payload. The NDF has two valid states:

- To indicate a normal operation of the SDH pointer bytes the NDF is set to NNNN = 0110 in the transmitter. See figure 2.10. The receiver should interpret the content of the pointer bytes as their normal meaning.
- To indicate an inverse operation condition the NDF is set to NNNN = 1001 in the transmitter, which the transmitter will interpret as a new alignment for the envelope, and possibly TU size. When a new size is indicated all TU pointers (1-4), in the TU group shall simultaneously indicate NDF with the same new size.



Size	Designation	TU pointer offset range in multiframe
00	TU - 2	0 ~ 427
10	TU - 12	0 ~ 139
11	TU - 11	0 ~ 103

Table E.1: TU size and pointer offset value

Table E.1 indicates the TU size bit assignment of the V1 and V2 pointer word and the allowable pointer offset range for various TU applications. In Figure E.1 the starting byte of the TU-12 synchronous path envelope is the byte immediately following the V2 byte. This is the V5 byte, which is the first byte of the multiframe.

Figure E.1 shows a TU-12 multiframe. When the last ten bits of the V1, V2 word has a pointer value of ‘0’, then the first byte following the V2 pointer byte will be the starting byte of the TU-12 synchronous path envelope. This V5 byte is also the VC-12 POH byte, which is the first byte of the TU-12 multiframe.

Any offset value will have to fall within the offset range as shown in Table E.1: TU, and indicates the offset value from the V2 byte to the start of the TU-12 synchronous path envelope. Thus an offset value of 5 will indicate that the first byte of the TU-12 synchronous path envelope, the V5 byte, will be the fifth byte following the V2 byte.

The receiver must receive a new pointer value three times consecutive in order to effect the change else the new pointer value is ignored. When the majority of the ‘I’ bits, figure 2.10, is inverted the receiver shall decode these values to mean a positive frequency justification opportunity. Similarly if the ‘D’ bits, figure 2.10, are inverted, the receiver shall decode these values to mean a negative frequency justification opportunity.

For positive frequency justification, the ‘I’ bits in the V1, V2 word in figure 2.10 is inverted at the transmitter and sent to the receiver. The bytes following the V2 byte in the TU-12 multiframe is subsequently filled by dummy bytes in the transmitter. The frame with the ‘I’ bits inverted has no meaningful pointer value, and is used purely for frequency justification indication. No changes in the pointer value are allowed for at least three frames following this positive frequency justification opportunity.

For negative frequency justification, the ‘D’ bits of the V1, V2 word in figure 2.10 are inverted at the transmitter and sent to the receiver. This is followed by the V3 byte now carrying useful data, and the pointer byte is decremented by one. When V3 is not used for negative frequency justification its value is not defined. The receiver will ignore the value contained in V3 when it is not used for negative justification.

References

1. ITU-T Rec. G.707/Y.1322. “*Network node interface for the Synchronous Digital Hierarchy (SDH)*” December 2003.
2. SDR Forum, “SDR Forum FAQs.” http://www.sdrforum.org/faq.html. January 2004.
3. SDR Forum. “Definitions & Semantics” http://www.sdrforum.org.
4. SDR Forum. “Overview and Definitions of Software Download for RF Reconfiguration DL-DFN Document SDRF-02-A-0002-V0.00”6 August 2002.
5. CRONJÉ, J., *Software Architecture Design of a Software Defined Radio System*. MSc thesis, Stellenbosch University, October 2004.
6. BOSE, V., Design and Implementation of Software Radios Using a General Purpose Processor. PhD thesis, Massachusetts Institute of Technology, June 1999.
7. PUCKER, L., “CommsDesign – Paving Paths to Software Radio Design.” http://www.commsdesign.com/csdmag/sections/cover_story/OEG20010521S0118. June 2001
8. ITU-T Rec. G.783. “*Characteristics of synchronous digital hierarchy (SDH) equipment functional blocks*” April 1997.
9. Ming-Chwan Chow, Understanding SONET/SDH Standards and Applications, 1st edition.1995, Andan Publishers.
10. GREEN, P., “Progress in Optical Networking” *IEEE Communications Magazine,* Jan. 2001, pp. 54-61.
11. ITU-T Rec. G.831. “*Series G. Transmission Systems and Media, Digital Systems and Networks. Management capabilities of transport networks based on the synchronous digital hierarchy (SDH)*” March 2000.
12. ITU-T Rec. T.50. “ *Terminal Equipment and Protocols for Telematic Services. International Reference Alphabet (IRA). Information Technology – 7-BIT Coded Character Set for Information Interchange.*” September 1992.
13. ITU-T Rec. M.1400. “*Series M: TMN and Network Maintenance: International Transmission Systems, Telephone Circuits, Telegraphy, Facsimile and Leased Circuits. Designations for interconnections among operator’s networks.*” October 2001.
14. ITU-T Rec. E.164. “*Series E: Overall Network Operation, Telephone Service, Service Operation and Human Factors. The international public telecommunication numbering plan.*” February 2005.
15. ITU-T Rec. G.785 “SDH Management”
16. ITU-T Rec. G.784 “*Digital transmission systems- Terminal Equipments – Principal Characteristics of Multiplexing Equipment for the Synchronous Digital Hierarchy. Synchronous Digital Hierarchy Management*” June 1999.
17. K Tesink. Definitions of Managed Objects for the SONET/SDH Interface Type. Network Working Group. Telcordia Technologies. March 1999
18. Walter Goralski. SONET / SDH. Third edition. McGraw-Hill/Osborne, 2002.
19. R.Clauberg Data aggregation architectures for single-chip SDH/SONET framers. IBM J. RES & DEV. VOL. 47 NO. 2/3 March/May 2003.
20. ITU-T Rec. G.826 “Digital Networks – Quality and availability targets”

21. Brugia, O., Carbonelli, M. and Perucchini, D.: “Performance analysis of the error monitoring methodologies recommended for SDH systems” Electronics Letters, 16 August 1990, Vol.26 No. 17.

22. Byeong Gi Lee , Minho Kang and Jonghee Lee.: “Broadband Telecommunications Technology” Second Edition, 1996 Artech House, Boston London.

23. Jean-Thierry Calvet., Claude Girault. “A Simulated Environment for SDH Synchronisation Network Planning” Alcatel, Route de Nozay, 91461 Marcoussis Cedex, France.