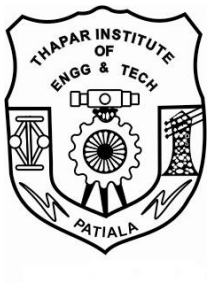
Designing and Implementation of GFP-T Frame Mapper

A Thesis

Submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Electronics and Communication Engineering



By:
Gaurav Vashishtha
(Regn No: 8044110)

Under the supervision of:

Mr. Balwant Singh
(Sr. Lecturer, ECED)

Mr. Rajesh Khanna
(Assistant Professor, ECED)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA – 147004

JUNE 2006

Certificate

I hereby certify that the work, which is being presented in this thesis, entitled **"Designing and Implementation of GFP-T Frame Mapper"** in partial fulfillment of the requirements for the award of degree of **Master of Engineering in Electronics and Communication Engineering** at Electronics and Communication Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Mr. Balwant Singh, Sr. Lecturer, ECED and Mr. Rajesh Khanna, Assistant Professor, ECED.

I have not submitted the matter presented in the thesis for the award of any other degree of this or any other university.

(Gaurav Vashishtha)

(Regn. No: 8044110)

This is to certify that the above statement made by the candidate is correct and true to best of my knowledge.

(Mr. Balwant Singh)

Supervisor
Sr. Lecturer
Electronics and Communication
Engineering Department
Thapar Institute of Engineering &
Technology, PATIALA-147004

(Mr. Rajesh Khanna)

Supervisor
Assistant Professor
Electronics and Communication
Engineering Department
Thapar Institute of Engineering &
Technology, PATIALA-147004

Countersigned by

(Dr. R.S. Kaler)
Professor & Head
Electronics and Communication Engineering Department
Thapar Institute of Engineering and Technology
PATIALA-147004

(Dr. T.P. Singh)
Dean of Academic Affairs
Thapar Institute of Engineering and Technology
PATIALA-147004

Acknowledgement

It is with the deepest sense of gratitude that I am reciprocating the magnanimity, which my supervisors **Mr. Balwant Singh** , Sr. Lecturer , Electronics and Communication Engineering Department & **Mr. Rajesh Khanna** , Assistant Professor , Electronics and Communication Engineering Department have bestowed on me by providing individual guidance and support throughout this Thesis work.

I am thankful to **Dr. A.K. Chatterjee**, P.G. Coordinator, Electronics and Communication Engineering Department for the motivation and inspiration that triggered me for my thesis work.

I would also like to thank all the staff members and my co-students who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis.

I take pride of myself being son of ideal great parents whose everlasting desire, sacrifice, affectionate blessings and help made it possible for me to complete my studies.

I am highly indebted to my wife **Preeti Vashishtha,** without whose inspiration and constant help, it would not have been possible to complete this work.

I am also thankful to the authors whose works I have consulted and quoted in this work.

Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

(Gaurav Vashishtha)

(Regn: 8044110)

ABSTRACT

Ethernet and SONET/SDH/OTN are on a crash course. With Ethernet continuing to push its way out of the enterprise into the access and metro sectors, developers of SONET equipment are being pushed to map Ethernet frames over SONET/SDH/OTN links. Packet traffic in a multitude of co-existing protocols is expected to dominate the flow of network communications traffic in the foreseeable future. The bulk of data transport will continue to be via SONET/SDH or subsequently Optical Transport Networks (OTN). Thus, an efficient generic enveloping protocol for adapting multiple types of packet traffic to SONET/SDH and OTN is needed. The generic framing procedure (GFP) fills this critical need and enables efficient provisioning of these multi-service data connections. GFP provides a generic mechanism to adapt traffic from higher-layer client signals over an octet synchronous transport network. Xilinx has provided a GFP-T core that performs this generic framing. But this Xilinx GFP core does not performs 8b/10b encoding/decoding rather it takes the data after 8b/10b encoding/decoding and forms a GFP-T frame. The requirement here is to send eight serial channels on a GFP-T stream. Incoming streams can be of type Gigabit Ethernet, Fiber channel (1Gb or 2Gb) or ESCON. As the GFP-T core does not perform 8b/10b encoding/decoding, thus before feeding the data to the GFP core there is a need to convert the serial data to the parallel (8-bit). This is done using the Xilinx CPCS Core, which performs serial to 10-bit conversion & 8b/10b encoding/decoding. Next step is to connect these two-xilinx cores (CPCS and GFP). There are eight streams coming from CPCS (each stream per CPCS core) and GFP core interface needs single stream. Thus for the whole setup to work it needs some glue logic in between these two cores. This glue logic is **GFP-T Frame Mapper**. The function of the GFP-T frame mapper is to multiplex the incoming streams from the CPCS and feed it to the GFP-T framer as a single stream.

The GFP-T Frame Mapper was implemented in hardware using VHDL language and simulations were done using Model SIM software and synthesis using Leonardo Spectrum software.

This work represents a critical first step and approach towards achieving an architecture through which any number of incoming channels can be mapped on the GFP-T core.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ATM	Asynchronous Transfer Mode
cHEC	Core HEC
CID	Channel ID
CoS	Class of Service
CRC	Cyclic Redundancy Check
CSF	Client Signal Fail
CPCS	Configurable Physical Coding Sublayer
DWDM	Deployed dense Wavelength Division Multiplexing
eHEC	Extension HEC
EOF	End of Frame
ESCON	Enterprise Systems Connection
EXI	Extension Header Identifier
FC	Fibre Channel
FCS	Frame Check Sequence
FICON	Fibre Connection
FIFO	First In First Out
GFP	Generic Framing Procedure
GFP-T	Transparent GFP
HDLC	High-level Data Link Control
HEC	Header Error Check
IFG	Inter-Frame Gap
IP	Internet Protocol
IPG	Inter-Packet Gap
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
LCC	Last Control Character
LOS	Loss of Signal
LSB	Least Significant Bit

MTU	Maximum Transmission Unit
NE	Network Element
OA&M	Operations, Administration & Maintenance
ODU	Optical Data Unit
OTN	Optical Transport Network
PDU	Protocol Data Unit
PFI	Payload FCS Indicator
PLI	Payload Length Indicator
PPP	Point-to-Point Protocol
PTI	Payload Type Identifier
SDH	Synchronous Digital Hierarchy
SOF	Start of Frame
SONET	Synchronous Optical Network
SP	Source Port
SPE	Synchronous Payload Envelope
SRC	Source
tHEC	Type HEC
UPI	User Payload Identifier

Chapter 1

THE GENERIC FRAMING PROCEDURE

1.1 Introduction

Transmission rates in the backbones of the public transport networks will continue to rise with each new innovation in electro/optical transceiver/ transponder technology. As technology matures, existing Local Area Network (LAN), Metropolitan Area Network (MAN), and Wide Area Network (WAN) data networking solutions become candidates for integration into the existing transport network infrastructure, preferably under a common data transport framework. Over the last few years, a need has arisen for a simple traffic adaptation mechanism that can be used to integrate the current diverse spectrum of physical and data link layer formats into the common public transport network infrastructure. The desired traffic adaptation mechanism must be simple enough to scale gracefully with the ever-increasing transmission rates in the core of the transport networks. However, it must be flexible enough to accommodate diverse data transmission requirements: from stringent delay and throughput constraints in storage networking, to differentiated quality of service (QoS) handling in LAN interconnect, to best effort delivery service for internet browsing.

The Generic Framing Procedure (GFP) is a simple but flexible traffic adaptation mechanism specifically designed to transport either block coded or packet-oriented data streams over a byte-synchronous communications channel. GFP generalizes the error control-based frame delineation scheme successfully employed in asynchronous transfer mode (ATM) [1] to both fixed and variable data length. Unlike frame delineation mechanisms based on codeword delineation patterns, GFP does not require special preprocessing of the client's byte stream. Rather than relying on embedded data/control bits such as in 8B/10B and 64B/66B encoding [2, 3], or delineation flags such as in HDLC framing [4], GFP relies on the length of the current payload and an error control check for frame boundary delineation. Successful validation of these two pieces of information, conveyed in the GFP frame header, is used to determine proper data link synchronization and the number of bytes to the next incoming frame. By facilitating the processing of arbitrary blocks of bytes at a time, GFP substantially reduces processing

requirements for data link mappers/demappers. By exploiting the low bit error rate performance in modern fiber-based communications media for the data link synchronization logic, GFP further decreases receiver logic. This reduced implementation complexity makes GFP particularly suitable for high-speed transmission links such as point-to-point Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH links) [5, 6], wavelength channels in an Optical Transport Network (OTN) [7], or even dark fiber applications.

GFP allows the implementation of multiple transport modes that may coexist within the same transport channel. One mode, referred to as Frame-Mapped GFP, is optimized for packet switching environments where resource management functions are delegated to the native data clients. This is the transport mode used for native Point to-Point Protocol (PPP), IP, Multi Protocol Label Switching (MPLS), or Ethernet traffic. A second mode, referred to as Transparent - Mapped GFP, is intended for delay-sensitive storage area network (SAN) applications that require bandwidth efficiency and transparency to the line code data.

1.2 Need of GFP

GFP finds its roots in the data-driven traffic growth of the late ’90s. The initial applications were envisioned as a transport solution for data centric traffic over readily available dark fibers [8] from recently deployed dense wavelength division multiplexing (DWDM) systems. Later on, a need emerged for a standard-based mechanism to accommodate the diverse data link transport technologies prevalent in the enterprise market over the existing public transport infrastructure. GFP soon found applications over SONET and SDH as well as the emerging OTN technology. Various factors have contributed to these developments.

1.2.1 A Transport Layer Perspective

Although there has been a long desire to extend the reach of SONET/SDH-based transport networks to the very edge of corporate networks, three major concerns have hindered the widespread acceptance of SONET/SDH technology as the preferred transport vehicle for data traffic into the WAN. First, most commercial SONET/SDH

Add/Drop Multiplexers (ADMs) and Broadband Cross-Connect systems (BXCs) had been heavily optimized to transport voice traffic, including support of timing and protection requirements, over the transport requirements of other traffic types. Although such optimizations were not harmful to packet switching technologies being deployed in the long-haul backbones (e.g., frame relay and ATM), the access model was burdensome to the bulk of LAN and MAN technologies typically deployed in enterprise networks. Second, the SONET/SDH management and operational framework had traditionally presumed direct end-user access to SONET/SDH channels (at the SONET/SDH path level). The higher cost structure of such an access model was inconsistent with the low-cost transport needs of the enterprise networking market. Third, even though it is possible to transport LAN technologies [9] over the public SONET/ SDH-based transport network infrastructure by defining a suitable mapping of the LAN bit stream onto the SONET/SDH payload area, both the rigidity of the readily available SONET/SDH channel sizes and the lack of a common adaptation procedure of the native data streams into the SONET/SDH path further slowed a more proactive deployment. Lastly, a native packet-oriented transport mechanism would enable alternative path protection and sharing options for both linear and ring configurations.

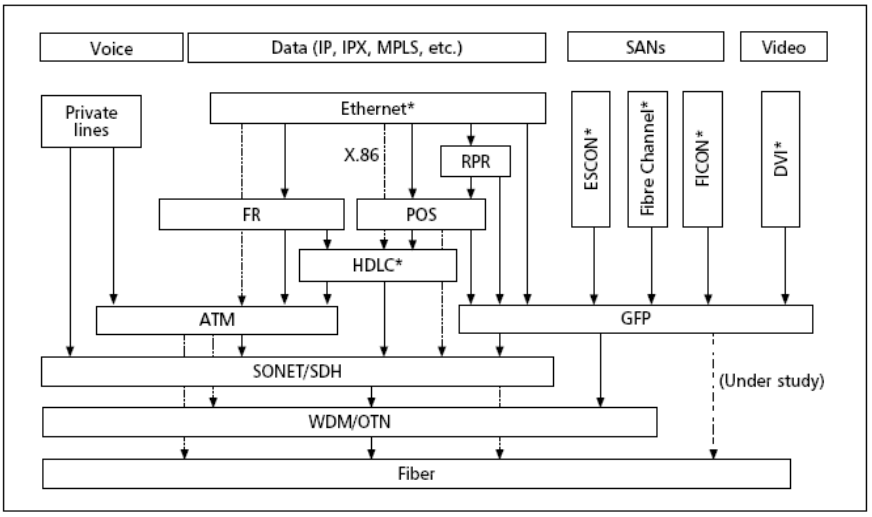


Figure 1.1: Adapting voice, data, storage, and video traffic over the public transport network infrastructure.

The first two issues can easily be handled by the adoption of a hybrid ADM and BXCs model that incorporates both SONET/SDH and packet based access interfaces into a single system as well as support for flexible provisioning of traffic resiliency options. This approach preserves a well-known service interface toward the end user while placing the adaptation of the native bit-stream into the SONET/SDH payload inside the hybrid transport system, rather than within the end user's Customer Premises Equipment (CPE). The introduction of inverse multiplexing facilities (the ability to bundle multiple independent lower-rate channels to create a logical higher-rate channel), referred to as "virtual concatenation" of synchronous payloads, and the specification of dynamic Link Capacity and Adjustment Schemes (LCAS) to dynamically and on demand provision and reconfigure Time-Division Multiplexed (TDM) channels to fit end-user needs, addressed granularity concerns with SONET/SDH (and OTN) transport. The final step, a common adaptation mechanism, is provided by GFP.

1.2.2 A Services Layer Perspective

Figure 1.1 illustrates a simplified taxonomy of the transport options for end-user applications such as voice, data, storage, and video traffic over the public network infrastructure. Frame relay, Asynchronous Transfer Mode (ATM), and Packet Over SONET/SDH (POS) represent the dominant service layer technologies for data delivery to small office/ home office and large corporations. High-level Data Link Control (HDLC) is the traffic adaptation mechanism (to a bit/byte synchronous physical transmission channel) in native frame relay and POS services. Until recently there has been limited support for connectivity services for protocols for storage networking or video over the traditional service technologies. When handled at the physical level Ethernet and Storage Area Network (SAN) protocols such as Fibre Channel, Enterprise Systems Connection (ESCON), and Fiber Connection (FICON) have traditionally been transported over the public network infrastructure by means of proprietary solutions that either optically extend the reach of the native signal or mimic an electrical "repeater" function.

Given the widespread availability of inexpensive 10/100/1000 Mb/s Ethernet interfaces for Customer Premise Equipment (CPE) switches/routers, the growing need to improve

data center/SAN interconnectivity, as well as the recent additions of virtual-LAN based Virtual Private Networking (VPN) and QoS capabilities, there is renewed interest in a common QoS friendly standard-based mechanism to transport IP, Ethernet, and SAN traffic over both TDM and Wavelength Division Multiplexing (WDM) networks. Although ATM and HDLC are the most common mechanisms employed to adapt native data traffic to the public transport network infrastructure, both have their own limitations as the base for such a simple common adaptation solution. In the case of ATM, it provides both transport and service layer integration that is far more than required to support simple connectivity services of interest to service providers. The large ATM cell tax is also an issue when bandwidth efficiency is a concern. HDLC has been the technology of choice for best effort wide area services, but it has long been distrusted as a reliable broadband transport mechanism for value-added connectivity services.

1.3 The GFP Solution

GFP provides a flexible encapsulation framework that supports either a fixed or variable length frame structure. As opposed to HDLC like framing, GFP does not rely on a byte stuffing mechanism to delineate Protocol Data Units (PDUs). Instead GFP uses a variation of the Header Error Check (HEC) based self-delineation technique. To accommodate variable lengths PDUs, an explicit payload length indicator is provided in the GFP frame header. Thus, the GFP PDU size can be fixed to a constant value (to provide a TDM-like channel), or changed from frame to frame (to allow easy extraction of the encapsulated PDUs). The explicit frame size indication also bounds the duration of the frame boundary hunt process, which is critical for data link synchronization. This client adaptation feature supports full encapsulation of the variable-length user PDU, hence obviating the need for segmentation/ reassembly functions, or frame padding to fill unused payload space. GFP also segregates error control between the GFP adaptation process and the user data. Error control segregation allows the delivery of corrupted payloads to be handed back to the intended receiver for further processing. This is a convenient feature for the transport of audio and video streams where corrupted information is preferred to no information at all.

Functionally, GFP consists of both common and client-specific aspects. Common aspects of GFP apply to all GFP adapted traffic and cover functions such as PDU delineation, data link synchronization and scrambling, client PDU multiplexing, and client independent performance monitoring. Client-specific aspects of GFP cover issues such as mapping of the client PDU into the GFP payload, client-specific performance monitoring, and Operations, Administration, and Maintenance (OA&M). This is illustrated in Fig. 1.2.

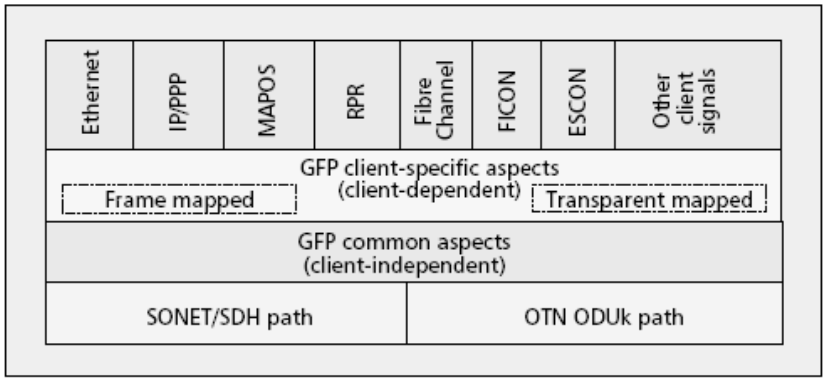


Figure 1.2: GFP relationship to client signals and transport paths.

1.4 Organization of Thesis

Chapter 2 discusses the common (protocol independent) aspects of GFP for octet aligned payloads.

Chapter 3 discusses about the payload specific aspects for transparent mapping of 8b/10b clients into GFP.

Chapter 4 describes the GFP-T Frame Mapper architecture, which maps eight incoming channels onto GFP-T.

Chapter 5 discusses the different entities which are used in the designing of the GFP-T Frame Mapper.

Chapter 6 gives the software and hardware implementation results.

Towards the end the conclusions and future scopes have been discussed.

Chapter 2

COMMON ASPECTS OF GFP

2.1 Introduction

This chapter discusses the common (protocol independent) aspects of GFP for octet aligned payloads. GFP uses a variation of the HEC-based frame delineation mechanism defined for Asynchronous Transfer Mode (ATM). Two kinds of GFP frames are defined: GFP client frames and GFP control frames. Frame formats for GFP client and control frames are discussed in the following units. GFP also supports a flexible (payload) header extension mechanism to facilitate the adaptation of GFP for use with diverse transport mechanisms. Currently defined payload extension header types are also discussed here.

2.2 Basic Signal Structure for GFP Client Frames

The format for GFP frames is shown in figure 2.1. GFP frames are octet-aligned and consist of a GFP core header and, except for GFP idle frames, a GFP payload area.

2.2.1 GFP Core Header

The GFP core header format is shown in figure 2.2. The four octets of the GFP core header consist of a 16-bit PDU length indicator field and a 16-bit core header error Check (cHEC) field. This header allows GFP frame delineation independent of the content of the higher layer PDUs.

PDU Length Indicator (PLI) Field

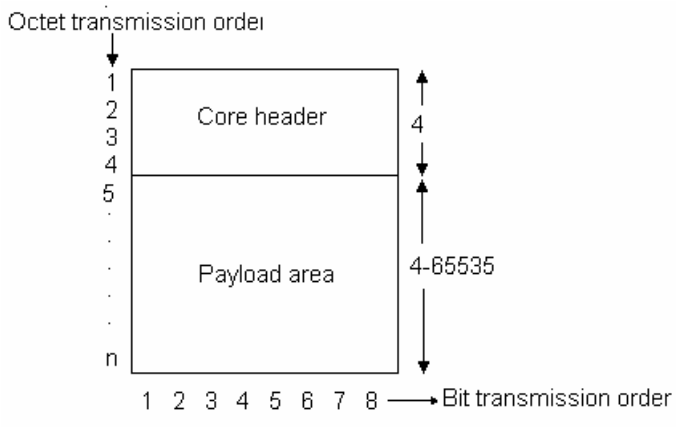
The two-octet PLI field contains a binary number representing the number of octets in the GFP payload area. The absolute minimum value of the PLI field in a GFP client frame is 4 octets. PLI values 0-3 are reserved for GFP control frame usage.

Core HEC (cHEC) Field

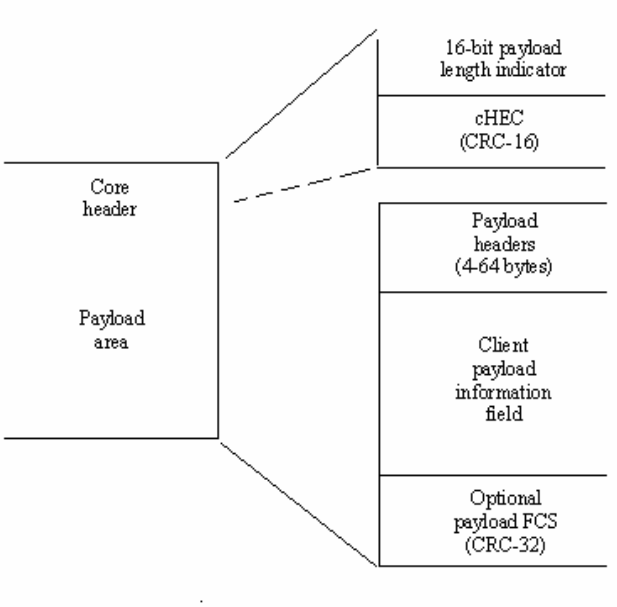
The two-octet Core Header Error Control (cHEC) field contains a CRC-16 error control code that protects the integrity of the contents of the core header by enabling both single bit error correction and multi-bit error detection.

Core Header Scrambling

The core header is scrambled for DC balance by an exclusive-OR operation (modulo 2 addition) with the hexadecimal number B6AB31E0. This number is the maximum transition, minimum side-lobe, Barker-like sequence of length 32. The scrambling of the GFP core header improves the robustness of the GFP frame delineation procedure and provides a sufficient number of 0-1 and 1-0 transitions during idle transmission periods.



a) Frame size and transmission order



b) Field format for GFP client frame

Figure 2.1: Frame format for GFP client frames.

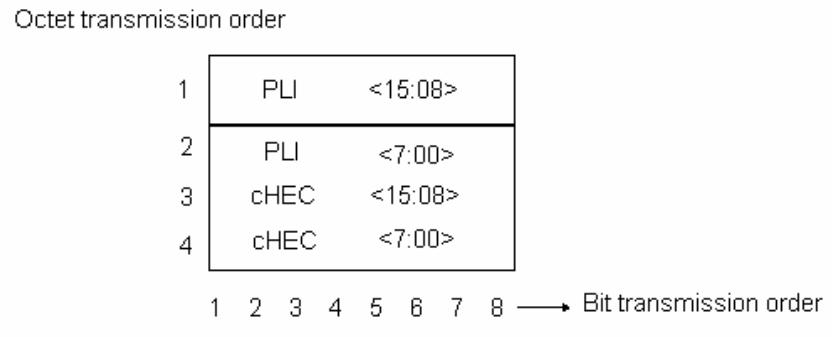


Figure 2.2: GFP core header format.

2.2.2 GFP Payload Area

The GFP payload area, which consists of all octets in the GFP frame after the GFP core header, is used to convey higher layer specific protocol information. This variable length area may include from 4 to 65535 octets. As shown in Figure 2.3, the GFP payload area consists of two common components: a payload header and a payload information field. An optional payload FCS (pFCS) field is also supported [10].

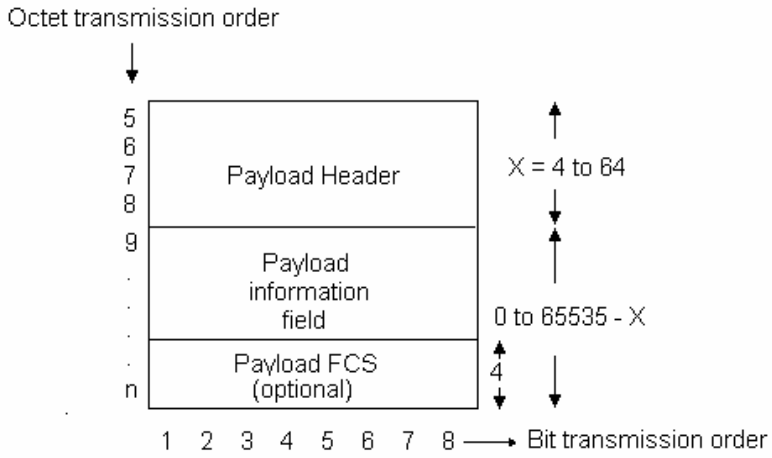


Figure 2.3: GFP payload area format.

Practical GFP Maximum Transmission Unit (MTU) sizes for the GFP payload area are application specific. An implementation should support transmission and reception of

GFP frames with GFP payload areas of at least 1600 octets. By prior arrangement, consenting GFP implementations may use other MTU values. Implementations supporting frame-mapped fiber channel should support GFP payload areas of at least 2156 octets.

Payload Header

The payload header is a variable-length area, 4 to 64 octets long, intended to support data link management procedures specific to the higher-layer client signal. The structure of the GFP payload header is illustrated in figure 2.4. The area contains two mandatory fields, the type and the "tHEC" fields, and a variable number of additional payload header fields. This group of additional payload header fields is referred to as the extension header. The presence of the extension header, and its format, and the presence of the optional payload FCS are specified by the Type field. The "tHEC" protects the integrity of the type field.

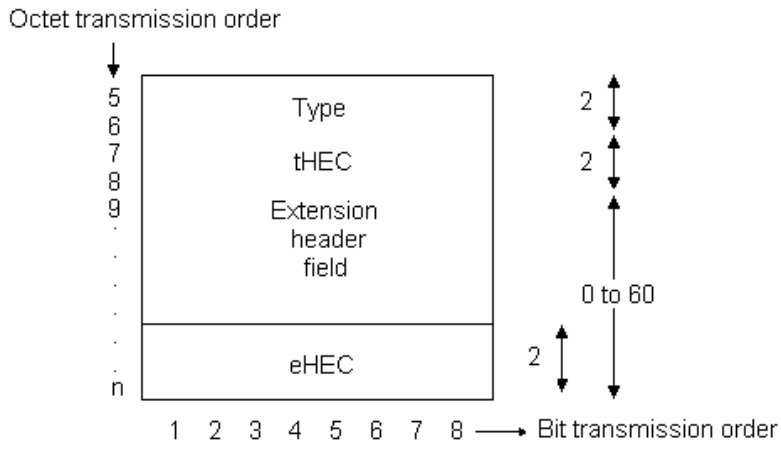


Figure 2.4: GFP payload header format.

An implementation shall support reception of a GFP frame with a payload header of any length in the range 4 to 64 octets.

GFP Type Field

The GFP type field is a mandatory two-octet field of the payload header that indicates the content and format of the GFP payload information field. The type field distinguishes

between GFP frame types and between different services in a multi-service environment. As shown in figure 2.5, the type field consists of a Payload Type Identifier (PTI), a Payload FCS Indicator (PFI), an Extension Header Identifier (EXI) and a User Payload Identifier (UPI).

Payload Type Identifier

The payload type identifier is a three bit subfield of the type field identifying the type of GFP client frame. Two kinds of client frames are currently defined, user data frames (PTI = 000) and client management frames (PTI = 100). PTI code points are given in table 2.1.

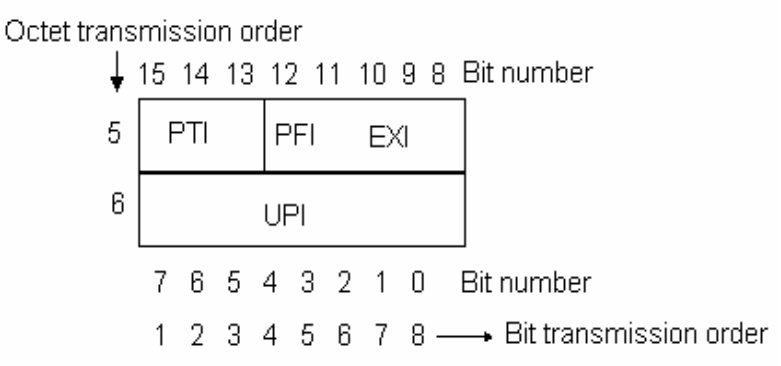


Figure 2.5: GFP type field format.

Payload FCS Indicator (PFI)

A one-bit subfield of the type field indicating the presence (PFI = 1) or absence (PFI = 0) of the payload FCS field.

Table 2.1: GFP payload type identifiers

Type bits <15:13>	Usage
000	Client Data
100	Client Management

Others	Reserved

Extension Header Identifier (EXI)

A 4-bit subfield of the type field identifies the type of extension header GFP. Three kinds of extension headers are currently defined a null extension header, a linear extension header, and a ring extension header. EXI code points are given in table 2.2.

User Payload Identifier (UPI)

An 8-bit field identifying the type of payload conveyed in the GFP payload information field. Interpretation of the UPI field is relative to the type of GFP client frame as indicated by the PTI subfield.

Table 2.2: GFP extension header identifiers

Type bits <11:8>	**Usage**
0000	Null Extension Header
0001	Linear Frame
0010	Ring Frame
Others	Reserved

Type HEC (tHEC) Field

The two-octet type header error control (tHEC) field contains a CRC-16 error control code that protects the integrity of the contents of the type field by enabling both single-bit error correction and multi-bit error detection. The type header consists of the type field and the “tHEC”.

GFP Extension Headers

The payload extension header is a 0-to-60 octet extended field (including the “eHEC”) that supports technology specific data link headers such as virtual link identifiers,

source/destination addresses, port numbers, class of service, extension header error control, etc. The type of the extension header is indicated by the content of the EXI bits in the type field of the payload header.

Three extension header variants are currently defined to support client specific data over a logical ring or logical point-to-point (linear) configurations. The various fields in each extension header are described here. The default value for any undefined fields is 0 unless otherwise stated.

Null Extension Header

The payload header for a frame with a null extension header is shown in figure 2.6. This extension header applies to a logical point-to-point configuration. It is intended for scenarios where the transport path is dedicated to one client signal.

Extension Header for a Linear Frame

The payload header for a linear (point-to-point) frame with an extension header, shown in figure 2.7, is intended for scenarios where there are several independent links requiring aggregation onto a single transport path.

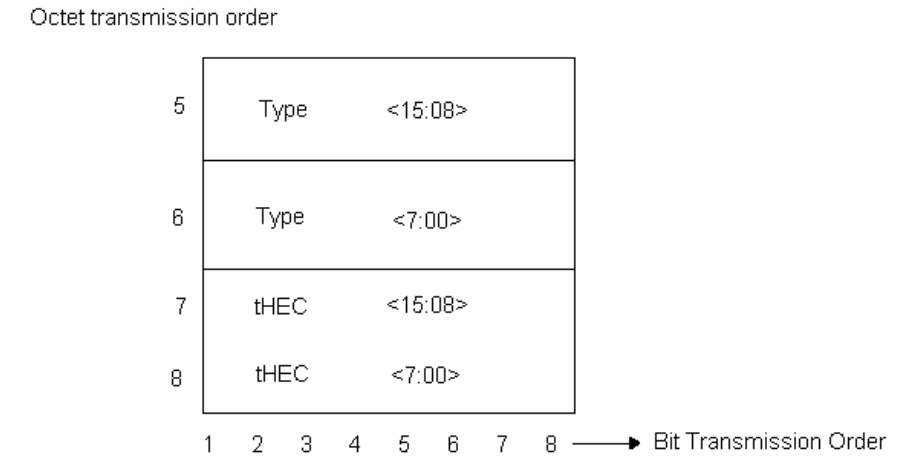


Figure 2.6: Payload header for a GFP frame with a null extension header.

Channel ID (CID) Field

The CID is an 8-bit binary number used to indicate one of 256 communications channels at a GFP termination point.

Spare Field

The 8-bit spare field is reserved for future use.

Extension HEC (eHEC) Field

The two-octet extension header error control field contains a CRC-16 error control code that protects the integrity of the contents of the extension headers by enabling both single-bit error correction (optional) and multi-bit error detection.

Payload Information Field

The payload information field contains the framed PDU for frame mapped GFP or, in the case of transparent GFP, a group of client signal characters. This variable length field may include from 0 to 65535-*X* octets, where *X* is the size of the payload header. This field may include an optional payload FCS field. The client PDU/signal is always transferred into the GFP payload information field as an octet-aligned packet stream.

Payload Frame Check Sequence (pFCS) Field

The GFP payload FCS, as shown in figure 2.8, is an optional, four-octet long, frame check sequence. It contains a CRC-32 sequence that protects the contents of the GFP payload information field. A value of 1 in the PFI bit within the type field identifies the presence of the payload FCS field [10].

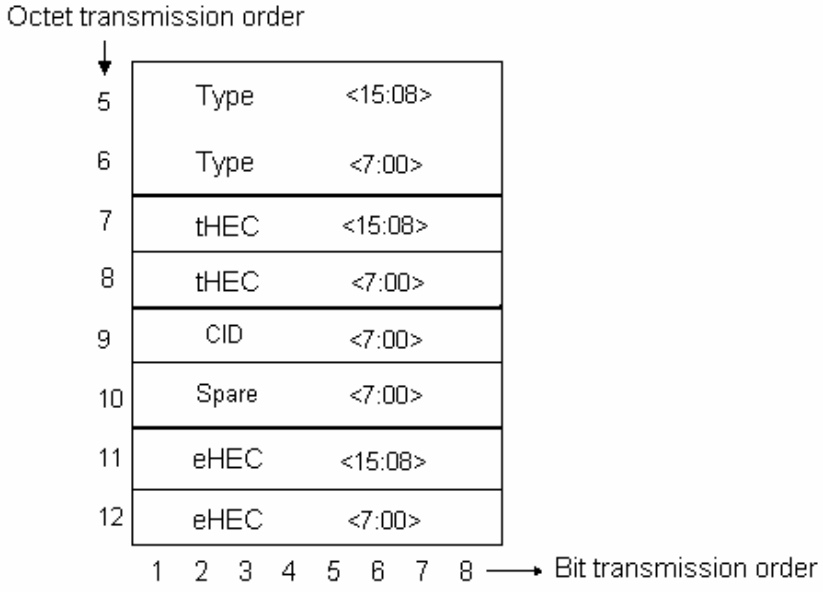


Figure 2.7: Payload header for a linear (point-to-point) frame including the extension header.

Payload Area Scrambling

Scrambling of the GFP payload area is required to provide security against payload information replicating scrambling word (or its inverse) from a frame synchronous scrambler channel. Figure 2.9 illustrates the scrambler and descrambler processes.

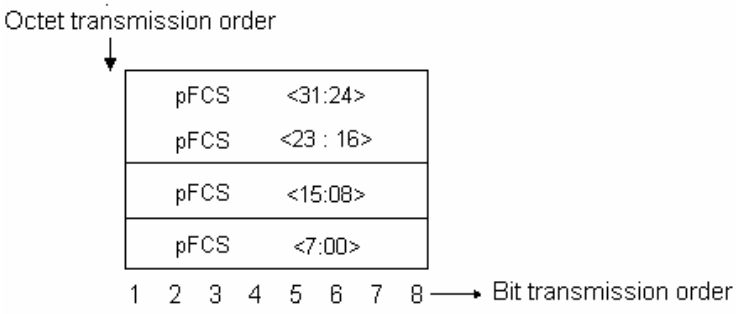


Figure 2.8: GFP payload frame check sequence format.

All octets in the GFP payload area are scrambled using a $1 + x^{43}$ self-synchronous scrambler. Scrambling is done in network bit order.

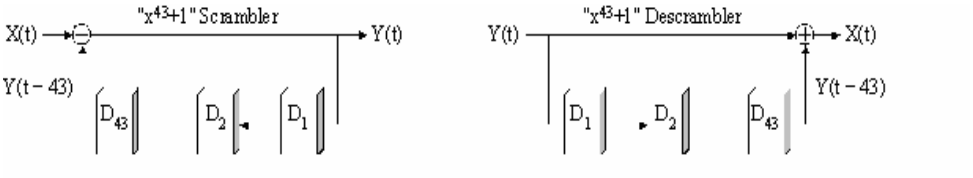


Figure 2.9: $X^{43}+1$ Scrambler and descrambler processes for GFP.

At the source adaptation process, scrambling is enabled starting at the first transmitted octet after the "cHEC" field, and is disabled after the last transmitted octet of the GFP frame. When the scrambler or descrambler is disabled, its state is retained. Hence, the scrambler or descrambler state at the beginning of a GFP frame payload area will thus be the last 43 payload area bits of the GFP frame transmitted in that channel immediately prior to the current GFP frame.

The activation of the sink adaptation process descrambler also depends on the present state of the "cHEC" check algorithm:

a) In the HUNT and PRESYNC states, the descrambler is disabled.

b) In the SYNC state, the descrambler is enabled only for the octets between the "cHEC" field and the end of the candidate GFP frame.

2.2.3 GFP Client Frames

Two types of GFP client frames are currently defined; client data and client management. GFP client data frames are used to transport data from the client signal. GFP client management frames are used to transport information associated with the management of the client signal or GFP connection [10].

Client Data Frames

Client data is transported over GFP using client data frames. Client data frames are GFP client frames consisting of a core header and a payload area. The type field of the client data frames uses the following type subfield values:

- PTI = 000.
- PFI = Payload specific.
- EXI = Payload specific.
- UPI = Payload specific.

The payload FCS indicator (PFI) shall be set as required depending on whether FCS is enabled or not. The Extension Header Identifier (EXI) shall be set consistently with the frame multiplexing and topology requirements for the GFP connection. The User Payload Identifier (UPI) shall be set according to the transported client signal type. Defined UPI values for client data frames are given table 2.3.

GFP Client Management Frames

Client management frames provide a generic mechanism for the GFP client specific source adaptation process to optionally send client management frames to the GFP client specific sink adaptation process. As illustrated in figure 2.10, the client management frames are GFP client frames consisting of a core header and a payload area. The type field of the client data frames uses the following type subfield values:

Table 2.3: User payload identifiers for GFP client frames

PTI = 000	
Type bits < 7:0 >	**GFP frame payload area**
0000 0000 1111 1111	Reserved and not available
0000 0001	Frame-Mapped Ethernet
0000 0010	Frame-Mapped PPP
0000 0011	Transparent Fibre Channel
0000 0100	Transparent FICON
0000 0101	Transparent ESCON
0000 0110	Transparent Gb Ethernet
0000 0111	Reserved for future

0000 1000	Frame-Mapped Multiple Access Protocol over SDH (MAPOS)
0000 1001	Transparent DVB ASI
0000 1010	Framed-Mapped IEEE 802.17 Resilient Packet Ring
0000 1011	Frame-Mapped Fibre Channel FC-BBW
0000 1100	Asynchronous Transparent Fibre Channel
0000 1101 through 1110 1111	Reserved for future standardization

- PTI = 100.
- PFI = Payload specific.
- EXI = Payload specific.
- UPI = Payload specific.

For use as a GFP client management frame, the payload FCS indicator shall be set as required depending on whether FCS is enabled or not. (Note that the use of FCS in GFP Client management frames reduces the amount of 'spare' bandwidth that can be used for such frames.) The Extension Header Indicator (EXI) shall be set as required depending on whether the extension header is employed or not. (Note that the use of extension header in GFP client management frame will significantly reduce the amount of 'spare' bandwidth that can be used for such frames.)

The UPI defines the use of the GFP client management frame payload. In this way the GFP client management frame may be used for multiple purposes. Table 2.4 defines the GFP client management frame payload uses.

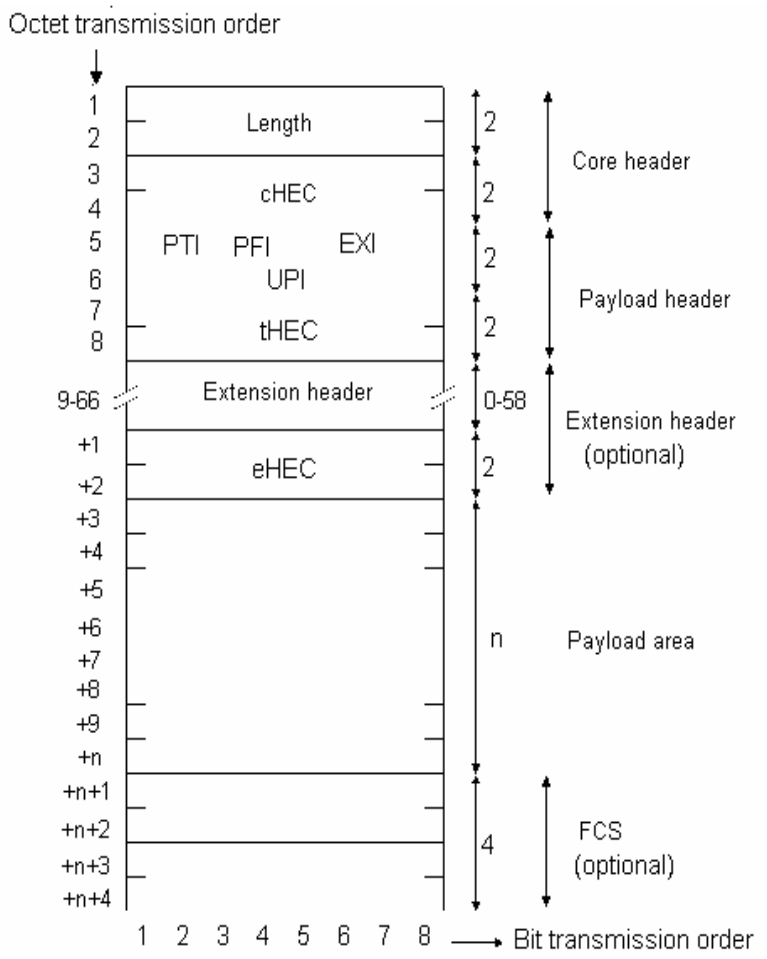


Figure 2.10: GFP client management frame.

2.3 GFP Control Frames

GFP control frames are used in the management of the GFP connection. The only control frame specified at this time is the GFP idle frame.

2.3.1 GFP Ideal Frames

The GFP idle frame is a special four-octet GFP control frame consisting of only a GFP core header with the PLI and 'cHEC' fields set to 0, and no Payload Area. The idle frame is intended for use as a filler frame for the GFP source adaptation process to facilitate the adaptation of the GFP octet stream to any given transport medium where the transport

medium channel has a higher capacity than required by the client signal. The GFP idle frame format is shown in figure 2.11, with the parenthetical values indicating the values after the Barker-like scrambling has been performed.

Table 2.4 – GFP client management frame user payload identifier

PTI = 100	
UPI value	**Usage**
0000 0000 1111 1111	Reserved
0000 0001	Client Signal Fail (Loss of Client Signal)
0000 0010	Client Signal Fail (Loss of Character Synchronization)
0000 0011 through 1111 1110	Reserved for future use

2.4 GFP Frame Delineation Algorithm

GFP frame delineation is performed based on the correlation between the first two octets of the GFP frame and the embedded two-octet "cHEC" field. Figure 2.12 shows the state diagram for the GFP frame delineation method.

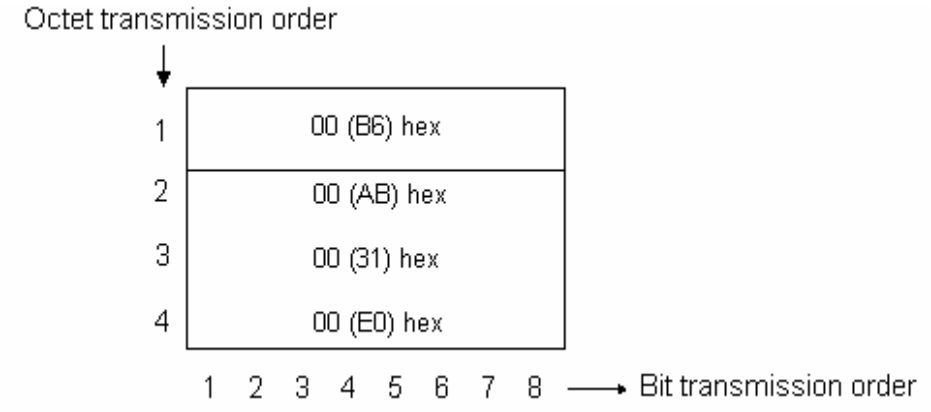


Figure 2.11: GFP idle frame.

The state diagram works as follows:

1) In the HUNT state, the GFP process performs frame delineation by searching, octet-by-octet, for a correctly formatted core header over the last received sequence of four octets. The core header single error correction is disabled while in this state. Once a correct "cHEC" match is detected in the candidate PLI and "cHEC" fields, a candidate GFP frame is identified and the receive process enters the PRESYNC state.

2) In the PRESYNC state, the GFP process performs frame delineation by checking, frame-by-frame, for a correct "cHEC" match in the presumed core header of the next candidate GFP frame. The PLI field in the core header of the preceding GFP frame is used to find the beginning of the next candidate GFP frame. Core header single error correction remains disabled while in this state. The process repeats until DELTA consecutive correct "cHECs" are confirmed, at which point the process enters the SYNC state. If an incorrect "cHEC" is detected, the process returns to the HUNT state. The total number of consecutive correct "cHECs" required to move from the HUNT state to the SYNC state is therefore DELTA + 1.

3) In the SYNC state, the GFP process performs frame delineation by checking for a correct "cHEC" match on the next candidate GFP frame. The PLI field in the core header of the preceding GFP frame is used to find the beginning of the next candidate GFP frame. Single-bit core header error correction is enabled while in this state. Frame delineation is lost whenever multiple bit errors are detected in the core header by the "cHEC". In this case, a GFP loss of frame delineation event is declared, the framing process returns to the HUNT state, and a client Server Signal Failure (SSF) is indicated to the client adaptation process.

4) Idle GFP frames participate in the delineation process and are then discarded.

Robustness against false delineation in the re-synchronization process depends on the value of DELTA. A value of DELTA = 1 is suggested.

Frame delineation acquisition speed can be improved by the implementation of multiple "virtual framers", whereby the GFP process remains in the HUNT state and a separate PRESYNC sub-state is spawned for each candidate GFP frame detected in the incoming octet stream, as depicted in figure 2.12.

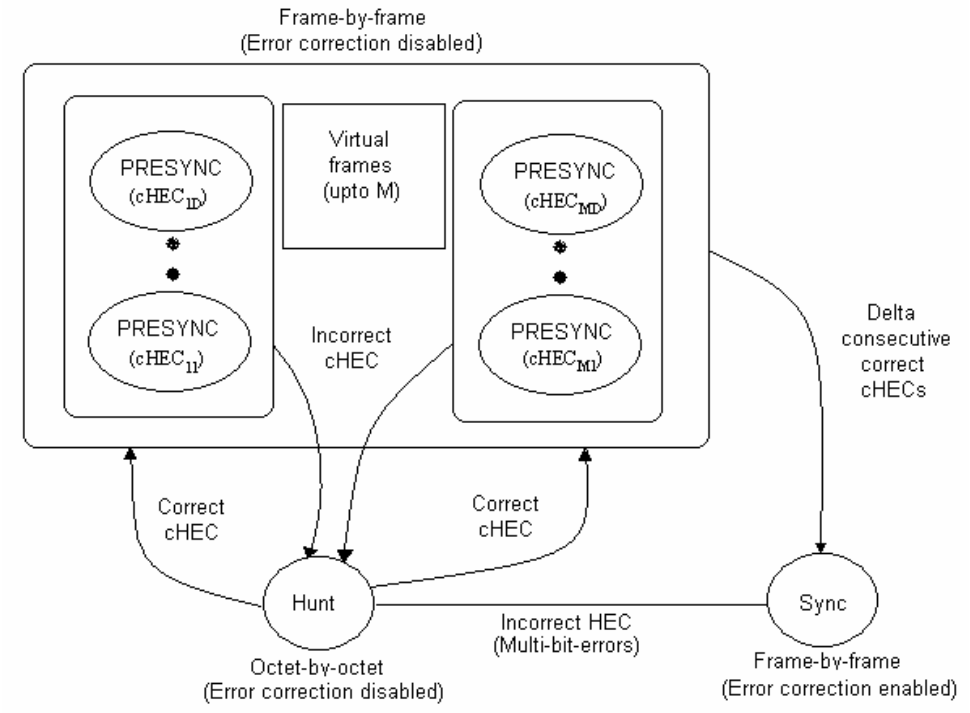


Figure 2.12: GFP frame delineation state diagram.

2.4.1 Frame Multiplexing

GFP frames from multiple ports and multiple client types are multiplexed on a frame-by-frame basis. When there are no other GFP frames available for transmission, GFP idle frames shall be inserted, thus providing a continuous stream of frames for mapping into an octet aligned physical layer [10] [11].

2.4.2 Cient Signal Fail Indication

GFP provides a generic mechanism for a GFP client-specific source adaptation process to propagate a Client Signal Fail (CSF) indication to the far-end GFP client-specific sink-

adaptation process on detection of failure defect in the ingress client signal. Detection rules for client signal fail events are by definition client-specific. Upon detection, a GFP source adaptation process should generate a client management frame (PTI = 100). The PFI subfield is set to 0 (no payload information field FCS), and the EXI subfield is set to the appropriate extension header type as applicable. The two types of CSF use the following UPI field values:

– Loss of Client Signal (UPI = 0000 0001).

– Loss of Client Character Synchronization (UPI = 0000 0010).

Upon detection of the CSF condition, the GFP client-specific source adaptation process should send CSF indications to the far end GFP client-specific sink adaptation process once every 100 ms ≤ T ≤ 1000 ms, beginning at the next GFP frame. Interim frames shall be GFP idle frames. Upon reception of the CSF indication, the GFP client sink adaptation process declares a sink client signal failure.

The GFP client-specific sink adaptation process should clear the defect condition either:

1) After failing to receive N CSF indications in $N \times 1000$ ms (a value of 3 is suggested for N); or

2) Upon receiving a valid GFP client data frame.

Equation 1

Chapter 3

PAYLOAD SPECIFIC ASPECTS FOR TRANSPARENT MAPPING OF 8B/10B CLIENTS INTO GFP

3.1 Introduction

Transparent mapping of 8B/10B payloads into GFP is intended to facilitate the transport of 8B/10B block-coded client signals for scenarios that require very low transmission latency. Examples of such client signals include fibre channel, ESCON, FICON, and Gigabit Ethernet. Rather than buffering an entire frame of the client data into its own GFP frame, the individual characters of the client signal are demapped from the client block codes and then mapped into periodic, fixed-length GFP frames. The mapping occurs regardless of whether the client character is a data or a control character, which thus preserves the client 8B/10B control codes. Frame multiplexing is not precluded with transparent GFP [10] [11].

3.2 Common Aspects of GFP-T

The transparent GFP frame uses the same frame structure as the frame-mapped GFP including the required payload header. The payload FCS is optional. The transparent GFP frame format is depicted in figure 3.1.

3.2.1 Adapting 8B/10B Client Signals via 64B/65B Block Codes

The first step in the client adaptation process is decoding the physical layer of the client signal. For 8B/10B line codes, the received 10-bit character is decoded into its original 8 bit value, if it is an 8B/10B data codeword or into a control character if it is an 8B/10B control codeword. The 8B/10B control code words are mapped into one of the 16 possible 4-bit control code indicators for the 8-bit control characters available in transparent GFP (table 3.1).

The decoded 8B/10B characters are then mapped into a 64-bit/65-bit (64B/65B) block code. The structure of the 64B/65B block code is shown in figure 3.2. The leading bit of the 65-bit block, the Flag bit, indicates whether that block contains only 64B/65B 8-bit data characters or whether client control characters are also present in that block. (flag bit

= 0 indicates data octets only and Flag bit = 1 indicates at least one control octet in the block). Client control characters, which are mapped into 8-bit 64B/65B control characters

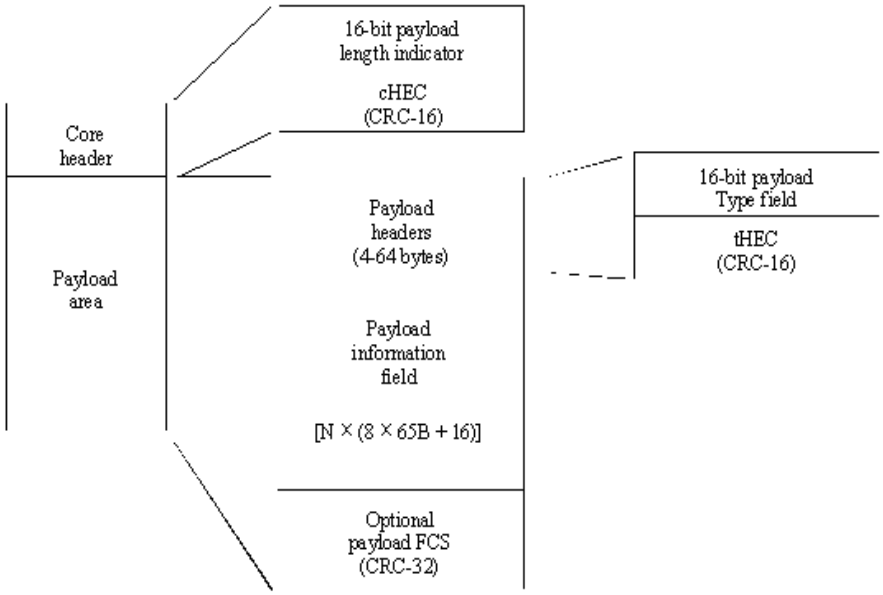


Figure 3.1: Transparent GFP frame format.

are located at the beginning of the 64-bit block payload if they are present in that block. The first bit of the 64B/65B control character contains a Last Control Character (LCC) flag bit which indicates whether this control character is the last one in this block (LCC = 0), or whether there is another control character in the next octet (LCC = 1). The next three bits contain the control code locator, which indicates the original location of the 8B/10B control code character within sequence of the eight client characters contained in the block. The last 4 bits, the control code indicator, give the 4-bit representation of the 8B/10B control code character. The explicit mapping of 8B/10B control code characters into the 4-bit control codes is defined in table 3.1. The control codes are mapped into the payload bytes of the 64B/65B code in the order in which they were received. Note that as a result, the control code addresses aaa-hhh in figure 3.2 will be in ascending order.

Table 3.1: Mapping between 8B/10B control characters and the 64B/65B control code indicators

NAME	Octet value	10B codeword (RD–) abcdei fghj	10B codeword (RD+) abcdei fghj	64B/65B 4-bit mapping
/K28.0/	1C	001111 0100	110000 1011	0000
/K28.1/	3C	001111 1001	110000 0110	0001
/K28.2/	5C	001111 0101	110000 1010	0010
/K28.3/	7C	001111 0011	110000 1100	0011
/K28.4/	9C	001111 0010	110000 1101	0100
/K28.5/	BC	001111 1010	110000 0101	0101
/K28.6/	DC	001111 0110	110000 1001	0110
/K28.7/	FC	001111 1000	110000 0111	0111
/K23.7/	F7	111010 1000	000101 0111	1000
/K27.7/	FB	110110 1000	001001 0111	1001
/K29.7/	FD	101110 1000	010001 0111	1010
/K30.7/	FE	011110 1000	100001 0111	1011
10B_ERR	01	Unrecognized RD–	Unrecognized RD+	1100
65B_PAD	02	N/A	N/A	1101
Spare	03	N/A	N/A	1110
Spare	04	N/A	N/A	1111
NOTE 1 – While all 256 data characters must be supported, only 12 special 8B/10B control code words are recognized and used for 64B/65B control characters in Gigabit Ethernet, Fibre Channel, FICON and ESCON. Hence, the compression of special 8B/10B control code words into 4-bit values is possible				

without restricting client signals, or providing protocol-specific handling of 8B/10B control code words.

NOTE 2 – The re-coding process is entirely unaware of the meaning of control words or ordered sets. It simply generically recodes data and control words into 65B blocks. No knowledge of start-of-frame, end-of-frame, errors, idles, control codes, ordered sets, etc. is required.

For example, if there is a single 64B/65B control character in a block and it was originally located between 8B/10B data code words D2 and D3, the first octet of the 64B/65B block will contain 0.010.C1. The LCC value of 0 indicates that this 64B/65B control character is the last one in that block and the value of aaa = 010 indicates C1's location between D2 and D3. At the demapper, the 64B/65B data characters are remapped as 8-bit data octets and then encoded back into the 8B/10B data code words. For 64B/65B control characters, the four-bit control code indicators are remapped into the appropriate 8B/10B control code words with their positions within the original character stream restored based on the three-bit control code locator.

10 B_ERR Code

Certain client signal defects may produce 8B/10B code words on ingress to the GFP source adaptation process that cannot be recognized by the 64B/65B adaptation process (e.g., a client signal failure, an illegal 8/10B codeword or a legal codeword with a running disparity error, see 3.2). A special 64B/65B control character, the 10B_ERR code, is provided to convey such "unrecognized 8B/10B codeword" client signal defects. When reconstructing the client signal on egress from the transport network, it is recommended that received 10B_ERR codes be typically recoded by the demapper into an invalid transmission character of either 001111 0001 (RD–) or 110000 1110 (RD+) (fixed, illegal 8B/10B code words with neutral disparity, containing a transition within both the first three bits and last three bits of the codeword), depending on running disparity. Although the actual value of the unrecognized 8B/10B codeword is not retained, the occurrence and location of the client signal defect are preserved.

In addition to the recommended invalid transmission character (whose construction minimizes the possible creation of aliased commas when combined with adjacent

characters), 10B_ERR events may also be demapped into alternate invalid transmission characters [10], provided that these alternate invalid transmission characters also meet all 8B/10B coding rules, are of neutral disparity, and contain a minimum of one transition within both the first four bits and last four bits of the codeword.

Input client characters	**Flag bit**	**64-bit (8-Octet) field**							
All data	0	D1	D2	D3	D4	D5	D6	D7	D8
7 data, 1 control	1	0 aaa C1	D1	D2	D3	D4	D5	D6	D7
6 data, 2 control	1	1 aaa C1	0 bbb C2	D1	D2	D3	D4	D5	D6
5 data, 3 control	1	1 aaa C1	1 bbb C2	0 ccc C3	D1	D2	D3	D4	D5
4 data, 4 control	1	1 aaa C1	1 bbb C2	1 ccc C3	0 ddd C4	D1	D2	D3	D4
3 data, 5 control	1	1 aaa C1	1 bbb C2	1 ccc C3	1 ddd C4	0 eee C5	D1	D2	D3
2 data, 6 control	1	1 aaa C1	1 bbb C2	1 ccc C3	1 ddd C4	1 eee C5	0 fff C6	D1	D2
1 data, 7 control	1	1 aaa C1	1 bbb C2	1 ccc C3	1 ddd C4	1 eee C5	1 fff C6	0 ggg C7	D1
8 control	1	1 aaa C1	1 bbb C2	1 ccc C3	1 ddd C4	1 eee C5	1 fff C6	1 ggg C7	0 hhh C8

– Leading bit in a control octet (LCC) = 1 if there are more control octets and = 0 if this payload octet contains the last control octet in that block
– aaa = 3-bit representation of the 1st control code's original position (1st Control Code Locator)
– bbb = 3-bit representation of the 2nd control code's original position (2nd Control Code Locator)
…
– hhh = 3-bit representation of the 8th control code's original position (8th Control Code Locator)
– Ci = 4-bit representation of the ith control code (Control Code Indicator)
– Di = 8-bit representation of the ith data value in order of transmission

Figure 3.2: Transparent GFP 64B/65B code components.

Insertion of 65B_PAD Code and GFP Client Management Frames

Since the transparent GFP application requires that the available path (channel) capacity is at least that of the client signal base (pre-encoding) data rate, the input receive (ingress) buffer at the mapper will regularly approach underflow. For rate adaptation purposes, if a

transparent GFP frame is currently being transmitted and if there are no client characters ready for transmission by the transparent GFP mapper, the mapper shall insert a 65B_PAD padding character. The pad character is mapped into the GFP frame in the same manner as a control character and is recognized and removed by the GFP demapper.

Octet 1, 1							
Octet 1, 2							
Octet 1, 3							
. . .							
Octet 8, 7							
Octet 8, 8							
L1	L2	L3	L4	L5	L6	L7	L8
CRC-1	CRC-2	CRC-3	CRC-4	CRC-5	CRC-6	CRC-7	CRC-8
CRC-9	CRC-10	CRC-11	CRC-12	CRC-13	CRC-14	CRC-15	CRC-16

where: Octet j, k is the kth octet of the jth 64B/65B code in the superblock

Lj is the leading (Flag) bit jth 64B/65B code in the superblock

CRC-i is the ith error control bit where CRC-1 is the MSB of the CRC

Figure 3.3: Superblock structure for mapping 64B/65B code components into the GFP frame.

To minimize latency, the transparent GFP mapper can begin transmitting data as soon as the first 64B/65B code in the group has been formed rather than waiting for the entire superblock to be formed. Client data frames are transmitted with priority over client management frames. If a GFP client management frame is available to transmit, and the ingress buffer is nearly empty (e.g., if a 65B_PAD character has been sent during the current client data frame), then the client management frame may be sent after the current client data frame. In order to maintain low latency, it is recommended that for a right-sized channel only a single client management frame be sent between client data frames. It is also recommended that client management frames used with transparent GFP be

limited to a payload information field of eight bytes or less. Note that low latency may also be maintained by increasing the channel size to allow the exchange of additional client management frames.

3.2.2 Adapting 64B/65B Code Blocks into GFP

To preserve the octet alignment of the transparent GFP signal with the transport SDH/ODUk frame, the first step in the adaptation process is to group eight 64B/65B codes into a superblock as shown in figure 3.3. The leading (Flag) bits of each of the eight 64B/65B codes are grouped together into a first trailing octet. The sixteen bits of the last two trailing octets are used for a CRC-16 error check over the bits of this superblock.

Assuming no payload FCS and a null extension header, the resulting GFP frame is $[N \times ((65 \times 8) + 16) + (8 \times 8)]$ bits long, where N is the number of superblocks in the GFP frame. The value of N depends on the base, uncoded, rate of the client signal and on the transport channel capacity. The minimum value of N depends on the data rate of the client signal, the number of GFP frame overhead octets (e.g., 8 with no optional payload FCS and a null extension header), and the size of the payload envelope. Specifically, N_{min} must be chosen such that for the fastest tolerance client clock rate and slowest tolerance SDH/OTN clock rate, the time required to transmit the GFP frame containing the $N \times 8 \times 8$ client characters is less than the time in which the client can deliver these $N \times 8 \times 8$ characters to the GFP mapper.

Error Control with Transparent GFP

The 16-error control bits in a superblock (see figure 3.3) contain a CRC-16 error check code over the 536 bits in that superblock. If the demapper detects an error, it should output either 10B error characters or unrecognized 10B characters in place of all of the client characters contained in that superblock. The 10B error and unrecognized characters are described for disparity errors in the client-specific aspects. This replacement guarantees that the client receiver will be able to detect the presence of the error.

8B/10B codes have built-in error detection capability since a single bit error will always result in an illegal code. The increased bandwidth efficiency gained by decoding the 8B/10B codes and remapping the data into 64B/65B codes comes at the expense of much of this error detection capability. There are four situations in which errors can cause significant problems with 64B/65B codes. The first and most serious problem result if the

leading flag bit of the 64B/65B code is received in error. If the original block contained control codes, these codes will be interpreted as data, and if the original block contained only data, some of these bytes may be interpreted as control codes. The number of data bytes that are erroneously interpreted as control codes depends on the value of the first bit (i.e., the last control code indicator bit position) of the bytes and whether the values of the location address bit positions contain increasing values (which would always be the case for a legal block). Data erroneously converted into control codes could cause the truncation of a client data frame, which in turn can cause error detection problems for the client data since there is a possibility of the truncated client data frame appearing to have a correct CRC value [10]. A similar situation occurs when control characters are present and the last control code indicator bit is affected by an error. Also, errors in the control code location address will cause it to be placed in the wrong sequence by the demapper, and errors in a 4-bit control code value will cause the demapper to generate an incorrect control code. Any error that results in a spurious or incorrect control code has potentially serious consequences.

It is these potential error problems that lead to the addition of a CRC-16 to each superblock. The most reliable mechanism for error control is for the demapper to discard all of the data in a superblock in which an error is detected. The data is discarded by having the demapper output 10B_ERROR 8B/10B codes for those clients that have defined such a code, or another illegal 8B/10B character for all of the characters in that superblock. The CRC-16 optionally allows the possibility of single error correction.

The payload area of the GFP frame is scrambled with a self-synchronous scrambler, and another error control issue concerns the interaction between the GFP payload scrambler and the superblock CRC-16. To understand the issue here, it is helpful to first understand the rationale and implementation behind the payload scrambler.

The reasons for using a self-synchronized payload scrambling process are related to the physical properties of the transport medium and the desire for robustness in public networks. The line code used for SONET/SDH and OTN is non-return-to-zero (NRZ) (after the data has been passed through a SONET/SDH/OTN frame-synchronous scrambler). For NRZ, the laser is turned on for the bit period to represent a 1 and off to represent a 0. The advantage of the NRZ line code is its simplicity and bandwidth

efficiency. The disadvantage of NRZ, however, is that the receiver clock and data recovery circuits can lose synchronization after a long string of either 0s or 1s. The frame-synchronized scrambler, which is reset at regular intervals based on the SONET/SDH/OTN frame, is adequate to defend against normally occurring user data patterns. It would be possible, however, for a malicious user to choose a packet payload that is the same as the frame synchronized scrambler sequence. If this packet lines up in the correct position in the transport frame, an adequately long string of 0s or 1s can be generated to cause a loss of synchronization at the receiver. The resulting loss of synchronization will take down the transport link while the receiver attempts to recover, thus denying the link to other users in the meantime. This problem was originally discovered in ATM networks and is exacerbated by the longer frames used in POS or GFP. In order to guard against such attacks ATM, POS, and GFP use a self synchronous payload scrambler to further randomize the payload data. This self synchronous scrambler uses a polynomial of $x^{43} + 1$, which means that each bit of the ATM/POS/GFP payload area is exclusive ORed with the scrambler output bit that preceded it by 43 bit positions, as shown in figure 3.4. (The scrambler state is retained between successive GFP frames.) The decoder's descrambler reverses this process.

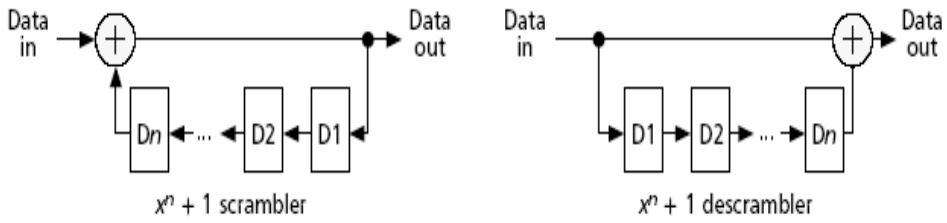


Figure 3.4: Payload self-synchronous scrambler.

In order to use the same payload scrambling technique for both frame-mapped and transparent GFP, all of the GFP payload bits including the GFP-T superblock CRCs must be scrambled. As a result, the superblock CRC has to be calculated over the superblock payload bits prior to scrambling and checked at the decoder after descrambling. The drawback to a self-synchronous scrambler, however, is that each transmission error results in a pair of errors (43 bits apart here) in the descrambled data, which means that

the superblock CRC must cope with this error multiplication. It has been shown [5, 6] that a CRC will preserve its error detection capability in this situation as long as the scrambler polynomial and the CRC generator polynomial have no common factors. Unfortunately, all of the standard CRC-16 polynomials contain $x + 1$ as a factor, which is also a factor in the $x^{43} + 1$ (or any $x^n + 1$) scrambler polynomial. Therefore, a new CRC generator polynomial was required that preserved the triple error detecting capability (which is the maximum achievable over this block size) without having any common factors with the scrambler. In order to perform single error correction, the syndromes for single errors and double errors spaced 43 bits apart must all be unique [6]. The code selected for the superblock is $x^{16} + x^{15} + x^{12} + x^{10} + x^4 + x^3 + x^2 + x + 1$, which has both these desired properties, and hence retains its triple error detection and optional single error correction capabilities in the presence of the scrambler [6–8].

3.3 Transparent GFP Client Management Frames

Client management frames have the same structure as GFP client data frames but are denoted by the payload type code PTI = 100 in the GFP payload header. Like GFP client data frames CMFs have a core header, payload header (both with 2-byte header error checking, HEC) and an optional 32-bit FCS. The total CMF payload size in GFP-T is recommended to be no greater than 8 bytes.

Assuming an 8-byte payload area along with the 8 total bytes for the mandatory core and type headers, the payload efficiency will be 50 percent. Use of FCS and especially extension headers will greatly reduce the efficiency of the CMFs.

As noted above, there is some residual “spare” bandwidth in the SONET/SDH channel for each of the client signal mappings. The amount of this “spare” bandwidth depends on the efficiency of the mapping, which in turn is partially a function of the number of superblocks used in each GFP frame. The residual bandwidth can be used as a client management overhead channel for client management functions. CMFs are also used for downstream indication of client signal fail [11].

3.3.1 Client Signal Fail Indication

GFP uses CMFs to indicate client signal fail (CSF) to the far-end GFP equipment. On detection of a failure defect in the ingress client signal a GFP CMF is transmitted following the current frame. This CMF uses PTI = 100, PFI = 0 (no FCS), appropriate EXI field, and UPI = 0000 0001 (loss of client signal) or 0000 0010 (loss of client character synchronization). Since the payload length indication is transmitted at the beginning of the client data frame and the onset of CSF can occur in the middle of a GFP client data frame, the remainder of the current client data frame will be filled with 10B_ERR codes to give it the required length.
Client management frames indicating CSF are sent every $100 < T < 1000$ ms in order to prevent:

• The receiver from being overwhelmed with frequent CSF indications

• In the case of a frame multiplexed scenario, excessive hogging of the bandwidth for CSF indication of just one channel

Upon reception of the CSF indication, the GFP client sink adaptation process (GFP receiver) declares a sink client signal failure and outputs either 10B_ERROR or other illegal 8B/10B codes on the client egress signal. If the CSF condition is a loss of signal and lasts beyond some time limit, the client egress output signal transmitter (e.g., the laser) may be turned off. The CSF condition at the receiver is cleared by:

• Reception of a valid client data frame

• After failing to receive N CSF indications in $N * 1000$ ms (a value of 3 is suggested for N).

3.3.2 Far-End Performance Reporting

The most universal client management application currently under consideration is the reporting of client-specific performance information from the far end of the GFP link. The GFP receiver can report such client-specific performance statistics as the bit error

rate (BER) or ratios of good to bad client frames either on a periodic basis or upon being queried. Far-end client-specific performance reporting allows both ends of the GFP link to see the status of both directions of the GFP link, which can be valuable if one of the ends is in an unmanned office or the link crosses carrier domains.

Chapter 4

THE GFP-T FRAME MAPPER

4.1 Introduction

This chapter describes the GFP-T frame mapper architecture, which maps eight incoming channels onto GFP-T. Incoming streams can be of type Gigabit Ethernet, Fibre channel (1Gb or 2Gb) or ESCON. The requirement here is to send eight serial channels on a GFP stream.

4.2 The GFP-T Frame Mapper

Xilinx GFP Core can be used for encapsulation of data into a GFP-T frame. But this xilinx GFP core does not performs 8b/10b encoding/decoding. It takes the data after 8b/10b encoding/decoding and forms a GFP-T frame. Thus, before feeding the data to the GFP core there is a need to convert the serial data to the parallel (8-bit). This is done using the Xilinx CPCS Core [12] [13], which performs following steps

1. Serial to 10-bit conversion.

2. 8b/10b encoding/decoding.

Such eight CPCS are instantiated as the goal is to support eight client streams. Each stream can be of type Gigabit Ethernet, Fibre channel (1Gb or 2Gb) or ESCON.

Next step is to connect these two-xilinx cores (CPCS and GFP). There are eight streams coming from CPCS (each stream per CPCS core) and GFP core interface needs single stream. Thus for the whole setup to work it needs some glue logic in between these two cores. This glue logic is **GFP-T Frame Mapper**. The function of the GFP-T frame mapper is to multiplex the incoming streams from the CPCS and feed it to the GFP-T framer as a single stream.

It can be seen in Figure 4.1 that Xilinx CPCS core interface is four-byte wide i.e. after 8b/10b decoding at the receive side it forms a word of 4-bytes [13]. And similarly it receives 4-byte word from the external interface and does 8b/10b encoding for each byte.Xilinx GFP core interface is 64-bits and supports OC192 data rate.

The functions of the GFP-T Frame Mapper are:

1. Converts 4-byte wide data to 8-byte wide and vice versa in the other direction.
2. Stores the data from each channel to be mapped onto GFP-T frame.
3. Prevents loss of data in any of the channels due to different rates at CPCS and GFP-T sides.
4. Provides sufficient data continuously from a channel to form a GFP frame.

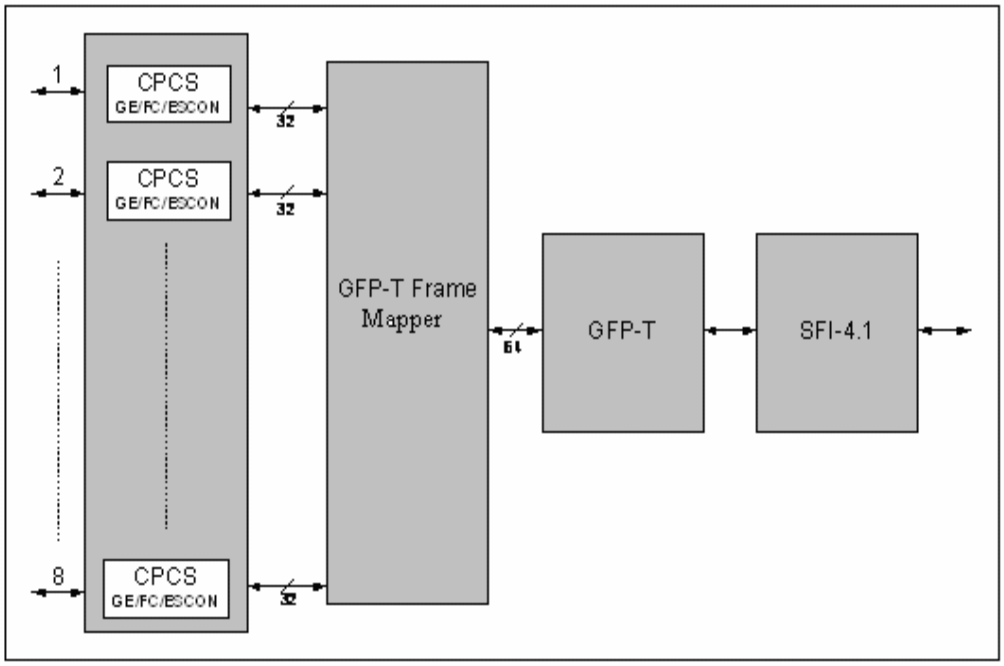


Figure 4.1: Interfacing CPCS and GFP Core.

SFI-4.1 interface provides 10G interface to the back plane.

4.2.1 Overview of GFP-T Frame Mapper

The main function of the GFP-T frame mapper is to MUX incoming eight channels onto GFP-T channel and.GFP-T frame mapper has following modules:

- CPCS interface
- FIFO
- GFP interface

CPCS interface will separate CPCS (xilinx CPCS core) and FIFO logic. This module helps CPCS and FIFO to exchange the data between them.
FIFO stores the data from eight different ports and maps onto a single GFP channel. This is actually an **8:1 MUX** in its complex form.
GFP interface facilitates data exchange between FIFO and GFP-T framer (xilinx GFP core).

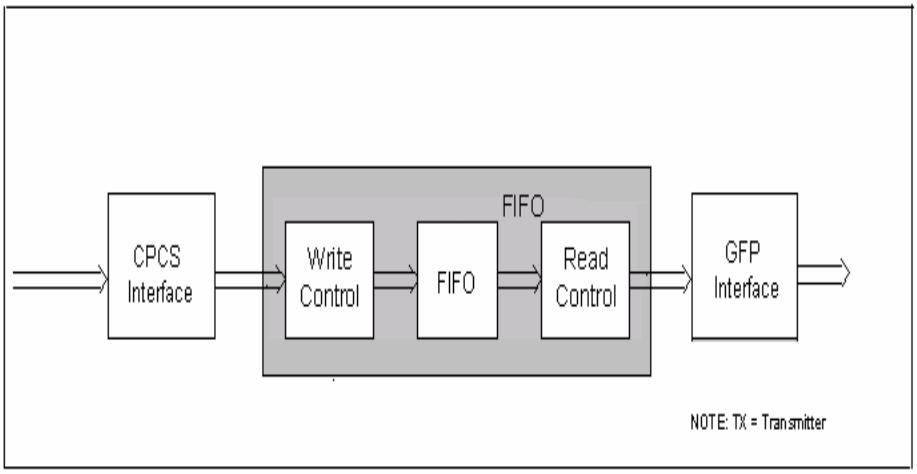


Figure 4.2: GFP-T Frame Mapper.

4.3 Requirements of the Design

The requirements of the design are:

1. Number of incoming channels and their type.
2. Rate of the incoming channels.
3. Carrier of GFP.

4.3.1 Number of Incoming Channels and Their Types

The number of incoming client channels is limited to eight. The client data stream type can only be of the type listed below.

1. Gigabit Ethernet (GE)
2. Fibre channel - 1Gb (FC1)
3. Fibre channel - 2Gb (FC2)
4. ESCON

4.3.2 Rate of the Incoming Channel

Xilinx GFP core supports approximately 10Gbps rate. Thus rate of the data stream going to the GFP core should not exceed 10Gbps after forming a GFP frame, i.e.

$$\text{(Client data + GFP overhead + GFP management frames)} < 10\text{Gbps} \quad (4.1)$$

Here one thing which must be considered is that where this GFP channel will be mapped. This design will be mapped on to the OTN (mapping GFP-T directly on 10G-OTN). The payload rate of the 10G-OTN frames is 9.953280Gbps. Then,

$$\text{(Client data + GFP overhead + GFP management frames)} < 9.953280\text{Gbps} \quad (4.2)$$

In other words, data rate (allotted on the line) of the all the eight channels put together should not exceed 9.953280Gbps. If it exceeds, then client data will be lost. Thus while configuring the port types (Gigabit Ethernet, Fibre channel (1Gb or 2Gb) or ESCON) care should be taken not to exceed 9.953280Gbps.

4.3.3 Carrier of the GFP

Carrier of the GFP-T frame can be

- Sonet
- SDH

- WDM
- **OTN**
- Can be directly mapped on to fibre.

GFP-T frames can be mapped onto the payload area of these frames listed above. The payload capacity varies for each of them. That means the channel bandwidth allotted for each of the client varies depending on the GFP-T carrier.

Channel bandwidth is inversely proportional to the number of super blocks to be transmitted in a GFP-T frame, i.e.

$$\text{Channel bandwidth } \alpha \ \frac{1}{\text{Number of superblocks}} \tag{4.3}$$

$$\alpha \ \frac{1}{\text{Number of Block RAMs}} \tag{4.4}$$

Thus as the channel bandwidth increases the number of Block RAMs required for the design decreases.As the GFP-T carrier changes the total Block RAMs needed for the design will also vary. This design is mainly intended for OTN.

4.4 CPCS and GFP Interface

CPCS interface communicates between the GFP-T Frame Mapper and the CPCS core. Data bus from CPCS to be passed to the GFP core i.e. data to be stored in the FIFO includes

1. Actual data (4 bytes)
2. Special character indication per byte (1bit/byte, therefore for 4bytes of data, 4-bits of special character)
3. Error indication per byte (1bit/byte, therefore for 4bytes of data, 4-bits of error indication)

The actual data is put in a GFP frame. Special character and error indication is used for 64B/65B encoding in the GFP core.

Instead of storing the Error indication, it is transmitted as an invalid special character, which is decoded before feeding to GFP core. In the CPCS interface when error is indicated, it is replaced with the invalid special character by asserting special character indication and replacing the data with an equivalent of invalid special character or 10BERR. This approach decreases the number of bits to be stored in the FIFO. At the GFP interface it is decoded back as error indication and fed to the GFP core.

GFP-T interface communicates between the GFP-T Frame Mapper and the GFP core.It passes the data stored in the FIFO to the GFP core when the GFP-T framer is ready to accept the data.

4.5 FIFO Design

It can be said that GFP-T frame mapper consists of FIFOs for each channel and an 8:1 multiplexer as shown in the figure 4.3.

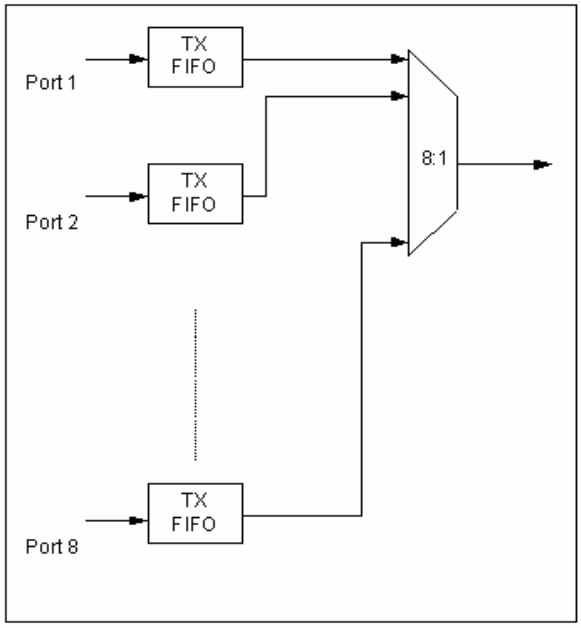


Figure 4.3: GFP-T Frame multiplexer and demultiplexer FIFO.

4.5.1 MUX FIFO (8:1)

Each channel FIFO is filled at constant rate depending upon the type data stream in that channel (1000Mbps for Gigabit Ethernet, 850Mbps or 1700Mbps for FC and 160Mbps in case of ESCON). FIFOs are read out at 10Gbps rate i.e. combined rate (i.e reading rate of all the eight FIFOs) at which the data is supplied to the GFP-T core. FIFO depths should be calculated precisely to avoid FIFO overflows, which will result in client data loss.

Round robin kind of approach is used to read the data from the FIFO. In this approach, FIFOs are visited in a cyclic manner and are read out whenever FIFO level reaches to the threshold value, i.e. next turn of each FIFO comes after reading data from all the other FIFOs. In other words,

In the first clock, 1st channel FIFO level is checked. If it is less than the threshold, in the next clock 2nd channel FIFO level is checked against the threshold for that FIFO. Else 1st channel FIFO is read out to form a GFP frame containing data of that channel. Thus the minimum threshold should be at least one GFP frame worth of data in the FIFO. Then only we can form one GFP frame carrying a specific port's data.

After completing the 1st channel (i.e. after FIFO level goes below the threshold) it moves to the 2nd channel. And this cycle repeats.

In this way number of GFP-T frames sent for particular channel purely depends on the level of the FIFO. The data (in full GFP-T frames) from each channel is sent out continuously till the FIFO level is less than the threshold. This is shown in the the flow chart in figure 4.4.

4.5.2 FIFO Requirement

Eight channels are coming at different data rates. These eight channels have to be mapped on to GFP-T channel. This is basically an 8:1 mux. The need for the FIFO arises because of

1. The different data rates of each of these eight incoming channels.
2. Optimal GFP frame lengths that have to be transmitted per channel as per standard.

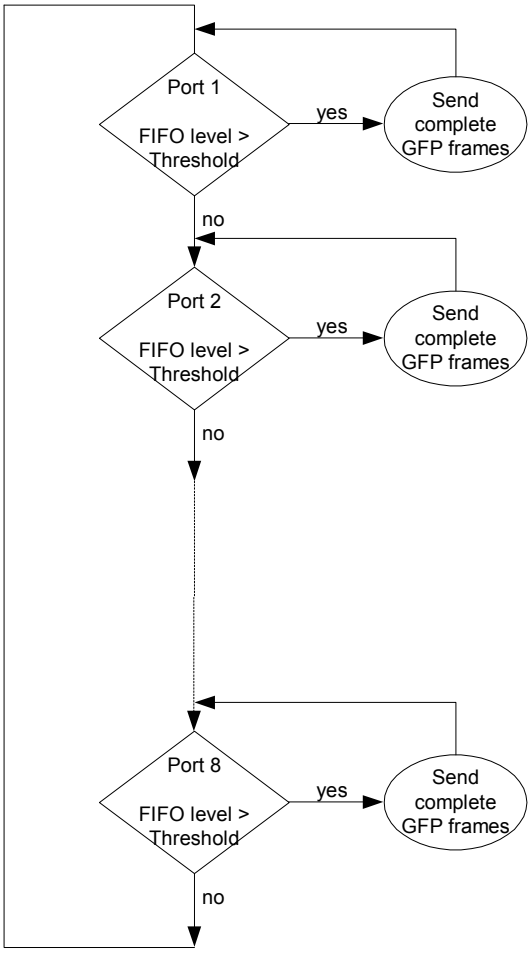


Figure 4.4: Flow diagram for MUX FIFO read Logic.

4.6 FIFO Size Investigation

The optimal length of the GFP-T frame decides the size of the FIFO. The formulae for length of the GFP-T frame are discussed in the following two sections.

4.6.1 Number of Superblocks per Port

GFP-T frame contains fixed size of blocks known as superblocks. Number of superblocks in a GFP-T frame decides the length of the frame.

The minimum number of superblocks is given by

$$N_{min} = \left\lceil \frac{(CSBW_{max})(GFPOH)}{(512)(ChBW_{min}) - (536)(CSBW_{max})} \right\rceil \quad (4.5)$$

where

$ChBW_{min}$ = Transport channel bandwidth with slowest end of the transport clock tolerance;

$CSBW_{max}$ = Client signal data rate with fastest end of the client clock tolerance; and

GFPOH = The number of GFP overhead bits.

The optimal number of superblocks is given by

$$N_{opt} = \frac{(CSBW_{max})[GFPOH + (m)(CMFL)]}{(512)(ChBW_{min}) - (536)(CSBW_{max})} \quad (4.6)$$

where

CMFL = CMF (Client Management Frame) frame length,

m = The number of CMFs that can be transmitted between client data frames

4.6.2 FIFO Size Computation

The requirement is to send N (ranges from N_{min} to N_{opt}) number of superblocks in a GFP frame. Thus in each FIFO there is a need to store minimum of N superblocks (minimum FIFO threshold, which is dependent on the port type) to start the transmission to the GFP. Write to all the FIFOs will happen at constant rate depending upon the respective port type. Since the round robin approach is used to read the FIFOs, while calculating the size of the FIFO the total time spent to empty all the FIFOs in the worst case must be considered.

4.6.3 Time Spent in a FIFO to Send a Single GFP Frame

Size of the super block = 67 bytes (64 bytes of data + 1 byte flag + 2 byte CRC).

In the worst case

$$T = \frac{\{(N_{SB} \times 67 \times 8) + (GFPOH + (N_{CMF} \times CMFL))\}}{R_{OTN}} \quad (4.7)$$

where

T = Time spent in a FIFO to send one GFP frame (us).

N_{SB} = Number of superblocks.

GFPOH = GFP frame overhead.

N_{CMF} = Number of client management frames.

CMFL = Length of client management frame.

R_{OTN} = Actual data rate of the OTN channel which is available for GFP frame which is equal to 9.953280 Gbps.

4.6.4 Weighted Time Spent Per Channel

In the worst case

$$T = T_{\max} \quad (4.8)$$

where

T_{max} = maximum of the time spent in a FIFO to send one GFP frame(T).

During maximum of the "Time spent in a FIFO to send one GFP frame", one or more than one GFP frames can be sent depending on the client data rate. This number of GFP frames sent from a FIFO during this time is rounded up to the nearest integer value. This is because once transmission from a FIFO starts, it will stop only after completing the transmission of the whole GFP frame worth of data.

Now

$$(N_{GFP})_{WT} = T_{\max} \Big/ T \tag{4.9}$$

where

$(N_{GFP})_{WT}$ = Number of GFP frame sent during the weighted time.

Thus

$$T_{WT} = (N_{GFP})_{WT} \times T \tag{4.10}$$

If a fractional value is obtained with this formula then it is rounded upto the nearest upper integer value.

4.6.5 Superblocks Accumulated Per FIFO

Total weighted time spent is given by

$$(T_{WT})_{TOT} = \sum T_{WT} \tag{4.11}$$

where

$(T_{WT})_{TOT}$ = Total weighted time spent.

During the total weighted time spent, the number of superblocks accumulated in each channel FIFO is calculated. This gives the superblocks accumulated in the channel when all the FIFOs have data to transfer.

$$(N_{SB})_{FIFO} = \frac{\{(T_{WT})_{TOT} \times R_{CLIENT}\}}{(D_{SB} \times 8)} \tag{4.12}$$

where

$(N_{SB})_{FIFO}$ = Superblocks accumulated per FIFO.

R_{CLIENT} = Data rate of the client.

D_{SB} = Data bytes of the superblock.

4.6.6 Block RAMs Per FIFO

In Xilinx virtex-4, size of the Block RAM = 2048 bytes.

Now, the minimum number of block RAMs per FIFO is given by

$$N_{BRMIN} = \frac{\{((N_{SB})_{FIFO} + N_{SBCH})(D_{SB})\}}{S_{BR}} \tag{4.13}$$

where

N_{BRMIN} = Minimum number of block RAM per FIFO.

N_{SBCH} = Number of superblocks to be sent on the GFP frame for the particular channel.

S_{BR} = Size of the Block RAM in bytes.

If a fractional value is obtained with this formula it needs to be rounded up to the nearest upper integer value.

The requirement is to read 8bytes at a time, as GFP-T interface is 8 byte wide. But with a single Xilinx Block RAM, only a maximum of 4 bytes can be read at a time. Thus 2 Block RAMs must be arranged in parallel and both of them must be read out at once so that 8 bytes are read out while reading. Thus the number of Block RAMs per FIFO has to be even number for each FIFO.

Thus the number of Block RAMs per FIFO is equal to "minimum number of Block RAMs per FIFO (N_{BRMIN})" rounded up to next even number only if it is an odd number. Otherwise it is same as the 'Minimum number of Block RAMs per FIFO.

In the figure 4.5 Block RAMs arrangement for a FIFO with 6 Block RAMs is shown. When writing 32-bits one of the six Block RAMs is enabled. While reading, two Block RAMs in a row is enabled and 64-bit data is readout.

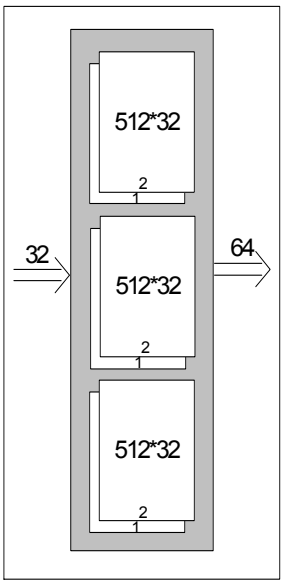


Figure 4.5: Block RAMs arrangement for a FIFO with 6 Block RAMs.

4.7 Data to be Stored in the FIFO

The actual client data is encapsulated onto a GFP-T frame. But for this encapsulation there is a need to pass some indications to the GFP core.GFP core needs special character indication for 64B/65B encoding.

This says that the data byte is a special control character or a data byte. Thus in the FIFO there is a need to store

- Actual data (32)
- Special character indication per byte (4)

All the calculations determining the FIFO size in the section 4.6 are done for the actual data.

Along with the data, special character indication needs to be transferred to the GFP core. To store these special characters memory bytes are required. This is done using the parity bits of the xilinx Block RAMs. In the xilinx Block RAM, there is a parity bit for each byte. These parity bits are used to store the special character indication per byte. By doing this total number of Block RAMs required for the design remains unchanged.

4.8 Challenges in the Design

The challenges in the design are:

4.8.1 Asynchronous Clocks

The frequencies of the client data streams are:

- 62.5 MHz for gigabit Ethernet
- 53.125 MHz for Fibre channel – 1G
- 106.25 MHz for Fibre channel – 2G
- 80 MHz for ESCON

The frequency of the GFP-T framer is 155.52 MHz. It is evident from the clocks listed above that they are asynchronous. If the clocks are synchronous we can design with single clock and clock enables. This makes design and implementation much easier. Asynchronous clocks involve FIFOs and also introduce design and timing issues.

4.8.2 Number of Block RAMs

Table 4.1 gives the number of Block RAMs for any client stream combination. But to design with those numbers of Block RAMs there is a need to dynamically allocate the Block RAMs. This type of design leads to implementation and timing complications. The implementation includes clock muxing (8:1 mux, we are getting one clock per port from CPCS) per Block RAM.

4.9 Implementation

Depending upon the combination of the port stream types, table 4.1 gives the number of Block RAMs required for each port. This means that this table gives the number of Block RAMs with which the incoming channel can be mapped onto the GFP-T channel without losing any data.Taking the maximum number of superblocks required for a channel for the worst-case combination as ‘B-max’. If for all the ports ‘B-max’ number of Block

RAMs is allocated, then any port can manage all types of data streams. This implementation is ideal.

The number of Block RAMs is equal to maximum number of Block RAMs (B-max) multiplied by the number of ports.

Example

For ESCON, N_{opt} = 1**;** for GE, N_{opt} = 32**;** for FC1 & FC2, N_{opt} = 26;

Allotted bandwidth for each channel is same as that of the standard.

With the N_{opt} values listed above maximum number of Block RAMs required per channel is equal to 4 Block RAMs.

Thus total number of Block RAMs required = 4 x 8

= 32 Block RAMs.

Table 4.1: Number of Block RAMs for any client stream combination.

Port	Type	CSBW (Mbps)	CHBW (MBps)	Nopt	T (us)/GFP frame	Tweighted (us)	N acc/port	BRAM Per Port
1	GE	1000	1088.64	32	1.790766461	1.790766461	37	4
2	GE	1000	1088.64	32	1.790766461	1.790766461	37	4
3	GE	1000	1088.64	32	1.790766461	1.790766461	37	4
4	GE	1000	1088.64	32	1.790766461	1.790766461	37	4
5	FC1	850	933.12	26	1.467656893	2.935313786	32	2
6	FC1	850	933.12	26	1.467656893	2.935313786	32	2
7	FC2	1700	1866.24	26	1.467656893	2.935313786	63	4
8	FC2	1700	1866.24	26	1.467656893	2.935313786	63	4
			9953.28					28

Note: Fields in orange are editable.

Chapter 5

INTERFACE AND MODULE DESCRIPTION

5.1 Introduction

This chapter discusses the different entities which are used in the designing of the GFP-T Mapper. Each entity is modeled using VHDL and the interface description of each entity is described in this chapter. Also the reset provision adopted, clocking scheme and technology specific dependencies are introduced.

5.2 Design Globals

5.2.1 Reset Provision

One asynchronous reset is used which resets all the eight channels as well as the outgoing GFP string.

5.2.2 Clocking Scheme

Design has two clocks, namely

- User clock (user_clk): 62.5 / 53.125 / 106.25 / 80 Mhz
- System clock (rxsysclk and txsysclk): 155.52 Mhz

Signals from CPCS are with respect to user_clk and signals to GFP Core are with respect to sysclk.

5.3 Technology Specific Dependencies

The following aspects of the tools/target platforms are depended upon for the successful implementation of this design. If this design is retargeted between tools or target technologies these may need redesign.

5.3.1 Synthesizer Dependent

The design is dependent upon the synthesis tool to infer LUTs for distributed RAM.

5.3.2 Xilinx Dependent

Xilinx dependent components used in the design are: RAMB16_S36_S36.

5.4 Mapper Top

5.4.1 Interface Description

Table 5.1 gives the interface description for the mapper top. It specifies the port names and direction, width, source/destination and a brief description for each port.

Table 5.1: Interface description for mapper top

Port Name	Dir	Width	From/to	Description
rst	in	1	GFP_CORE	Global asynchronous reset
sysclk	in	1	GFP_CORE	GFP side clock
user_clk	in	8	CPCS	CPCS channelwise clock
rx_src_rdy	in	8	CPCS	Data enable for rx_data
rx_er	in	[7…0][4]	CPCS	Error mask for rx_data
rx_data	in	[7…0][32]	CPCS	Data from CPCS
rx_has_k	in	[7…0][4]	CPCS	K charater mask for rx_data
framed	in	8	CPCS	Indicates framed or transparent mode
sync_status	in	[7…0][2]	CPCS	Synchronisation status indication

Port Name	Dir	Width	From/to	Description
pcs_mode_out	in	[7…0][2]	CPCS	PCS mode for each channel
m_sys_src_rdy_n	out	1	GFP_CORE	Data valid for m_sys_data
m_sys_dst_rdy_n	in	1	GFP_CORE	Destination ready signal from GFP core
m_sys_data	out	64	GFP_CORE	Read data from FIFO
m_sys_rem	out	3	GFP_CORE	Valid bytes in the m_sys_data from FIFO
m_sys_src_dsc_n	out	1	GFP_CORE	Asserted when PCS synchronisation is lost
m_sys_cid	out	8	GFP_CORE	Indicates channel id for linear frames
m_sys_upi	out	8	GFP_CORE	Indicates type of management frame when m_sys_mgmt_n is asserted
m_sys_charisk_n	out	8	GFP_CORE	Indicated if m_sys_data has k character
m_sys_10berr_n	out	8	GFP_CORE	Indicates if m_sys_data has

Port Name	Dir	Width	From/to	Description
				8b_10b encoding error
loss_of_sig	in	8	OE	Loss of signal indication
m_GFP frame_length	in	[7:0] [15:0]	TOP	length of the frame to be transmitted (in superblocks) per channel.
m_cid	in	[7:0] [7:0]	TOP	Channel identification byte per channel to map into the GFP-T frame.
m_los	out	[7:0]	TOP	Loss of SYNC. When channel_p losses of 10B synchronization, LOS[p] is set 1.

Port Name	Dir	Width	From/to	Description
m_fifo_ef	out	[7:0]	TOP	M_FIFO_EF is 1, when the FIFO is empty in the MAP direction. Default value is 0.
m_fifo_ff	out	[7:0]	TOP	M_FIFO_EF is 1, when the FIFO is full in the MAP direction. Default value is 0.

5.5 CPCS Interface

5.5.1 Interface Description

Table 5.2 gives the interface description for the CPCS interface. It specifies the port names and direction, width, source/destination and a brief description for each port.

Table 5.2: Interface description for CPCS interface

Port Name	Dir	Width	From/to	Description
rst	in	1		Global asynchronous reset.
clk_wr	in	1	TOP	CPCS channelwise clock .
tx_src_rdy	in	1	TOP	Data enable for incoming data.

tx_data	in	32	TOP	Data from CPCS.
tx_has_k	in	4	TOP	K charater mask for incoming data.
tx_err	in	4	TOP	Error mask for incoming data.
framed	in	1	TOP	Indicates framed or transparent mode.
pcs_tx_data	out	32	FIFO_TOP	Data from CPCS.
pcs_tx_data_en	out	2	FIFO_TOP	Data enable for outgoing data.
pcs_has_k_err	out	4	FIFO_TOP	K and error mask for outgoing data.

5.5.2 Module Overview

CPCS interface communicates between the GFP-T frame multiplexer and the CPCS core. Data bus from CPCS i.e. data to be stored in the FIFO includes

4. Actual data (4 bytes)
5. Special character indication per byte (1bit/byte. For 4bytes of data, 4-bits of special character)
6. Error indication per byte (1bit/byte. For 4bytes of data, 4-bits of error indication)

Formatted: Bullets and Numbering

The actual data is put in a GFP frame. Special character and error indication is used for 64B/65B encoding in the GFP core.

Instead of storing the Error indication it is transmitted as an invalid special character, which is decoded before feeding to GFP core.

In the CPCS interface when error is indicated, it is replaced with the invalid special character by asserting special character indication and replacing the data with an equivalent of invalid special character or 10BERR. This approach decreases the number of bits to be stored in the FIFO. At the GFP interface it is decoded back as error indication and fed to the GFP core.

5.6 FIFO Top

5.6.1 Interface Description

Table 5.3 gives the interface description for the FIFO top. It specifies the port names and direction, width, source/destination and a brief description for each port.

Table 5.3: Interface description for FIFO top.

Port Name	Dir	Width	From/to	Description
rst	In	1		Global asynchronous reset
clk_wr	In	8	TOP	Write domain clock
clk_rd	In	1	TOP	Read domain clock
m_sys_dst_rdy_n	In	1	GFP_CORE	GFP core ready signal to receive data
cpcs_tx_data	In	[7…0][32]	CPCS_INTF	Data from CPCS
cpcs_tx_data_en	In	8	CPCS_INTF	Data enable for cpcs_tx_data
cpcs_has_k_err	In	[7…0][4]	CPCS_INTF	K charater/error mask for cpcs_tx_data
superb_per_frame	In	[7…0][8]	CPU_REG	Superblock per frame for each channel
loss_of_sig	In	8	TOP	Loss of signal indication

Port Name	Dir	Width	From/to	Description
sync_status	In	8	TOP	Synchronisation status indication
fifo_dv	Out	1	GFP_INTF	Data valid from FIFO
fifo_data	Out	72	GFP_INTF	Retreived data from FIFO
fifo_ch_id	Out	3	GFP_INTF	Channel id from which the data is retreived
fifo_m_sys_src_dsc	Out	1	GFP_INTF	Indicates source discontinue for the current channel
fifo_empty	Out	1	CPU_REG	Provides Empty status of all channels
fifo_full	Out	1	CPU_REG	Provides Full status of all channels

5.6.2 Module Overview

This block implements multiplexer for 8 incoming channels. It has read and write domain which work on different clock frequencies. There are two controllers in this block.

1. **Write controller** for each channel keeps counting for number of data accumulated in the FIFO and when the count reaches the number of superblock per frame(configured) value it sends out an indication to read domain that for respective channel threshold has been reached and one frame can be read.

2. **Read Controller** reads out frames from each channel depending on how many frames are ready to read. Number of frames stored for each channel would be incremented when write controller sends an indication of threshold reached. It follows Round-Robin type for selection of next channel. It implements a state machine in which there are two states.

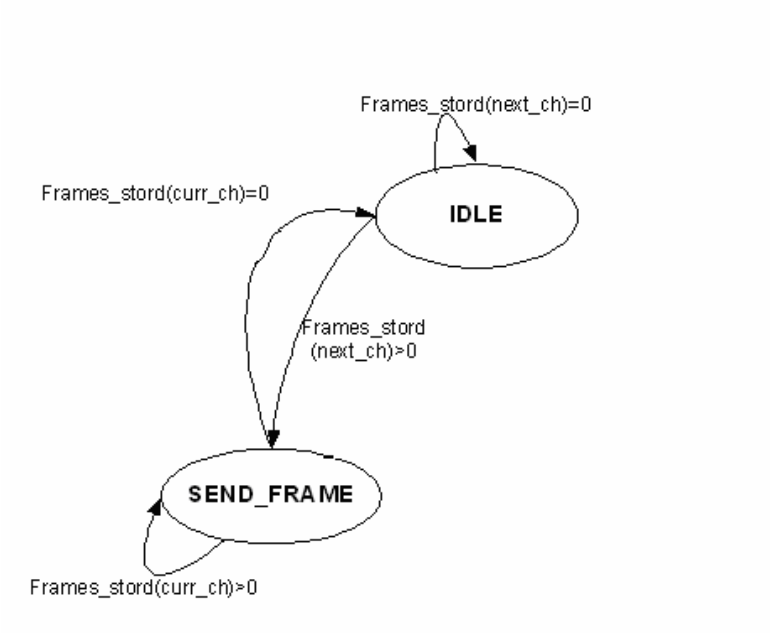


Figure 5.1: MUX FIFO State Machine.

a. **Idle**: This state keeps checking for the next channel selection which has to send a data frame. The order of selection is from 0 to 7^{th} channel.
b. **Send_Frame**: This state sends out data frames retrieved from FIFO.. It goes to Idle state if the above condition is not met to search for next channel. Suppose the synchronisation is lost for currently sending channel in between the data frame, then source discontinue is asserted.

5.7 FIFO Controller

5.7.1 Interface Description

Table 5.4 gives the interface description for the FIFO controller. It specifies the port names and direction, width, source/destination and a brief description for each port.

Table 5.4: Interface description for FIFO controller.

Port Name	Dir	Width	From/to	Description
rst	In	1		Global asynchronous reset
clk_wr	In	1	TOP	Write domain clock
clk_rd	In	1	TOP	Read domain clock
Enable	In	1	FIFO_TOP	Enable to read the data
loss_of_sync_wr	In	1	FIFO_TOP	Loss of synchronisation indication in write mode
loss_of_sync_rd	In	1	FIFO_TOP	Loss of synchronisation indication in read mode
fifo_full	Out	1	CPU_REG	FIFO full status
write_enable_in	In	1	FIFO_TOP	Write enable signal for port B of FIFO
write_data_in	In	36	FIFO_TOP	Data to be written into FIFO
fifo_empty	Out	1	TOP	FIFO empty status

Port Name	Dir	Width	From/to	Description
read_enable_in	Out	1	FIFO_TOP	Read enable signal for port A of FIFO
read_data_out	Out	72	FIFO_TOP	Data read from port A of FIFO

5.7.2 Module Overview

This block writes 36-bits(32b-data and 4b-K/err char) data from CPCS into FIFO and reads 72 bits(64b-data and 8b-K/err char) whenever there is a demand from GFP core. This is implemented using four Block RAMs per channel. On every write, 1 BRAM is written. Write address is incremented when all 4 BRAMs are written. For read, 2 BRAMs are read from 0 to 3rd BRAM in order and increments read address once all BRAMs are read.

Read and write domain signals are reseted when there is a reset from GFP and CPCS side respectively and both reseted when there is synchronisation lost for that channel.It also implements FIFO full and empty status. Full status asserted if read address is leading and gap (between read and write address) is 1 or 2 else deasserted. FIFO empty asserted if write address is leading and gap is less than 2 else deasserted.

5.8 GFP Interface

5.8.1 Interface Description

Table 5.5 gives the interface description for the GFP Interface. It specifies the port names and direction, width, source/destination and a brief description for each port.

5.8.2 Module Overview

GFP interface communicates between the FIFO on the Multiplexer side and the GFPT core.

Data from FIFO is passed to the GFP core.The error indication and K indication is decoded in this module and given as separate signals to the GFP core.

Also the CID and UPI values are sent to GFPT core according to the channel coming from FIFO.

Table 5.5: Interface description for GFP interface.

Port Name	Dir	Width	From/to	Description
rst	in	1		Global asynchronous reset
clk_rd	in	1	TOP	GFP side clock
fifo_out	in	72	FIFO_TOP	Data from FIFO to GFP Interface.
fifo_dv	in	1	FIFO_TOP	Data valid for Incoming data
pcs_mode	in	[1:0][7:0]	TOP	PCS mode for each channel
fifo_ch_id	in	3	FIFO_TOP	Channel ID corresponding to data coming from FIFO.
cid_cfg	in	[7:0][7:0]	TOP	Configured CID for each channel
m_sys_dst_rdy_n	in	1	TOP	Destination ready from GFPT Core
fifo_m_sys_src_dsc	in	1	FIFO_TOP	Source discontinue indication.
m_sys_src_rdy_n_out	out	1	TOP	Source ready out to GFPT core
m_sys_data	out	72	TOP	Data out
m_sys_cid	out	8	TOP	cid corresponding to output data
m_sys_upi	out	8	TOP	UPI corresponding to

				the output data
m_sys_charis_err_n	out	8	TOP	error indication corresponding to output data
m_sys_charis_k_n	out	8	TOP	K indication corresponding to output data
m_sys_rem	out	3	TOP	remainder out
m_sys_src_dsc_n	out	1	TOP	source discontinue indication

5.9 Positive Edge Detect

5.9.1 Interface Description

Table 5.6 gives the interface description for the entity positive edge detect. It specifies the port names and direction, width, source/destination and a brief description for each port.

Table 5.6: Interface description for positive edge detect.

Port Name	Dir	Width	From/to	Description
clk	in	1	TOP	125 Mhz clock for Gigabit Ethernet statistics
rst	in	1	-	Tied to 0
async_sig	in	1	Config_reg	Enable for CRC check and calculation.
sync_pulse	out	1	Config_reg	Bytewise data on which CRC has to be calculated.

5.9.2 Module Overview

This module detects the rising edge of an asynchronous input signal. The asynchronous signal is registered twice to get rid of metastability.The stable version is registered again

for detecting a rising edge and generate a one clock pulse on output signal 'sync_pulse' synchronous to clock 'clk'.

Equation 2

Chapter 6

SIMULATION RESULTS AND HARDWARE IMPLEMENTATION

6.1 Introduction

The GFP-T Frame Mapper was implemented in hardware using VHDL language which is the acronym for Very High Speed Integrated Circuit Hardware description language. The entire architecture may be viewed as a group of subsystems each having its own functionality and signals used for their proper working and synchronization between them.

There are a total of three different entities that were developed and a complete description of these entities along with their test simulations is provided below.

6.2 CPCS Interface

CPCS interface communicates between the GFP-T frame multiplexer and the CPCS core. Data bus from CPCS i.e. data to be stored in the FIFO includes

7. Actual data (4 bytes)
8. Special character indication per byte (1bit/byte. For 4bytes of data, 4-bits of special character)
9. Error indication per byte (1bit/byte. For 4bytes of data, 4-bits of error indication)

Formatted: Bullets and Numbering

The actual data is put in a GFP frame. Special character and error indication is used for 64B/65B encoding in the GFP core.

Instead of storing the Error indication it is transmitted as an invalid special character, which is decoded before feeding to GFP core.

In the CPCS interface when error is indicated, it is replaced with the invalid special character by asserting special character indication and replacing the data with an equivalent of invalid special character or 10BERR. This approach decreases the number of bits to be stored in the FIFO. At the GFP interface it is decoded back as error indication and fed to the GFP core.

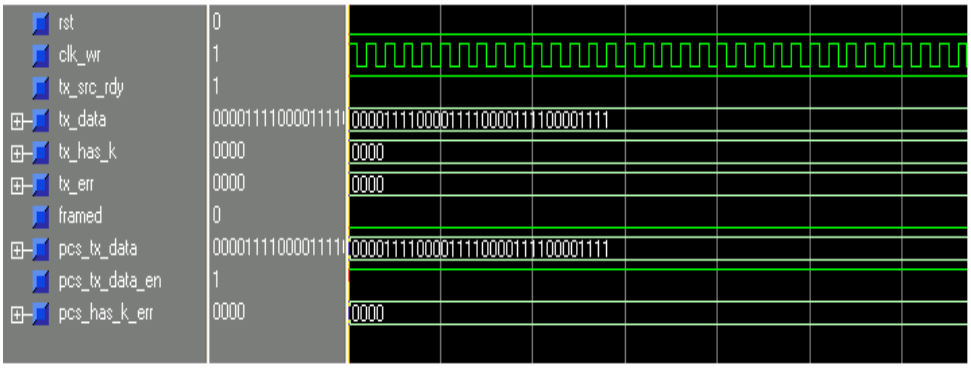


Figure 6.1: VHDL simulation result for entity CPCS interface.

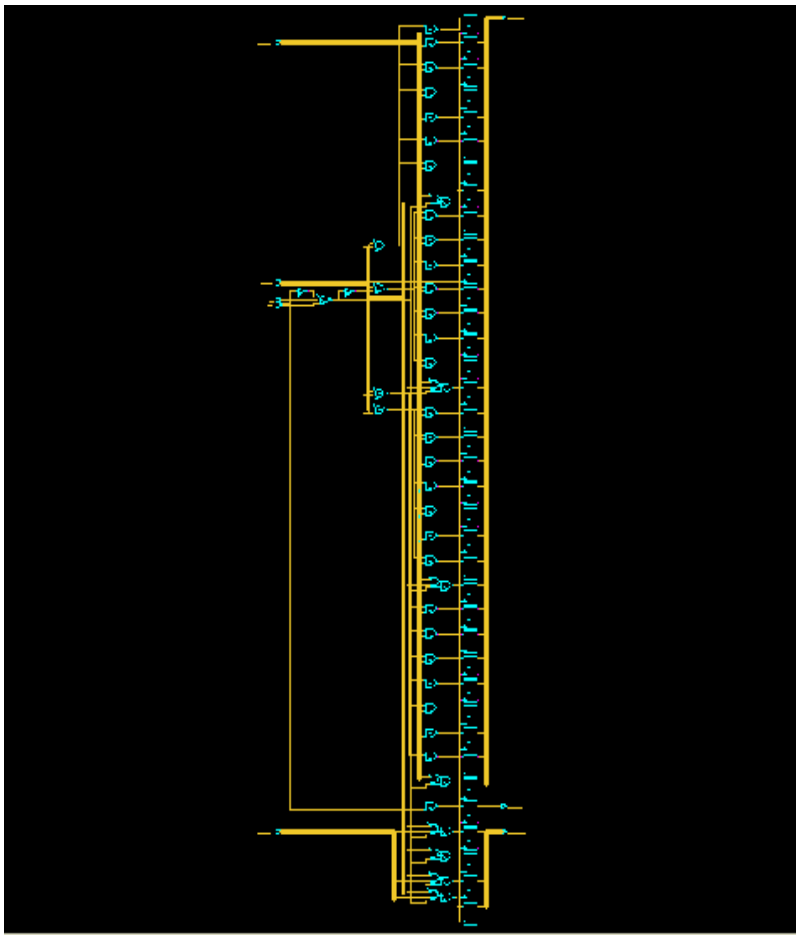


Figure 6.2: Layout of CPCS unit.

6.3 FIFO

This block implements a multiplexer for 8 incoming channels. It has read and write domain which work on different clock frequencies. There are two controllers in this block.

3. **Write controller** for each channel keeps counting for number of data accumulated in the FIFO and when the count reaches the number of superblock per frame(configured) value it sends out an indication to read domain that for respective channel threshold has been reached and one frame can be read.

Formatted: Bullets and Numbering

4. **Read Controller** reads out frames from each channel depending on how many frames are ready to read. Number of frames stored for each channel would be incremented when write controller sends an indication of threshold reached. It follows Round-Robin type for selection of next channel. It implements a state machine in which there are two states.
    a. **Idle**: This state keeps checking for the next channel selection which has to send a data frame. The order of selection is from 0 to 7th channel.
    b. **Send_Frame**: This state sends out data frames retrieved from FIFO.. It goes to Idle state if the above condition is not met to search for next channel. Suppose the synchronisation is lost for currently sending channel in between the data frame, then source discontinue is asserted.

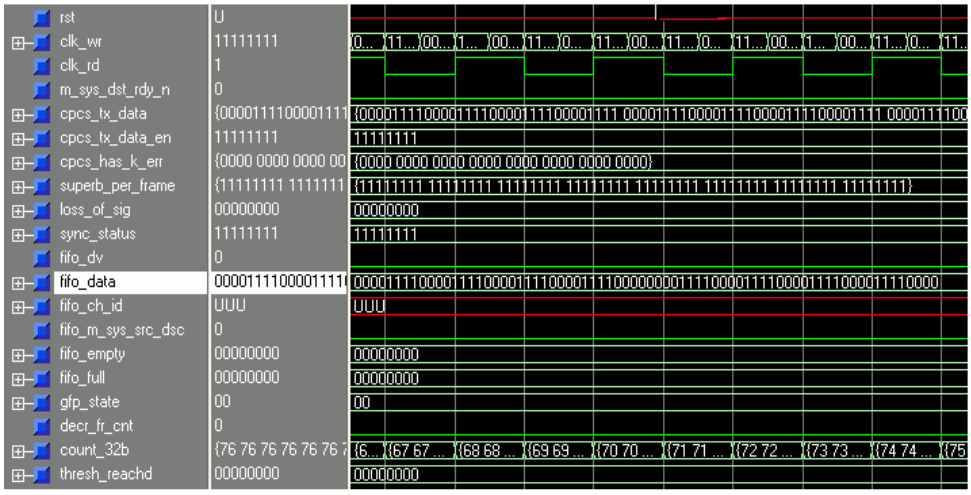


Figure 6.3: VHDL simulation result for FIFO.

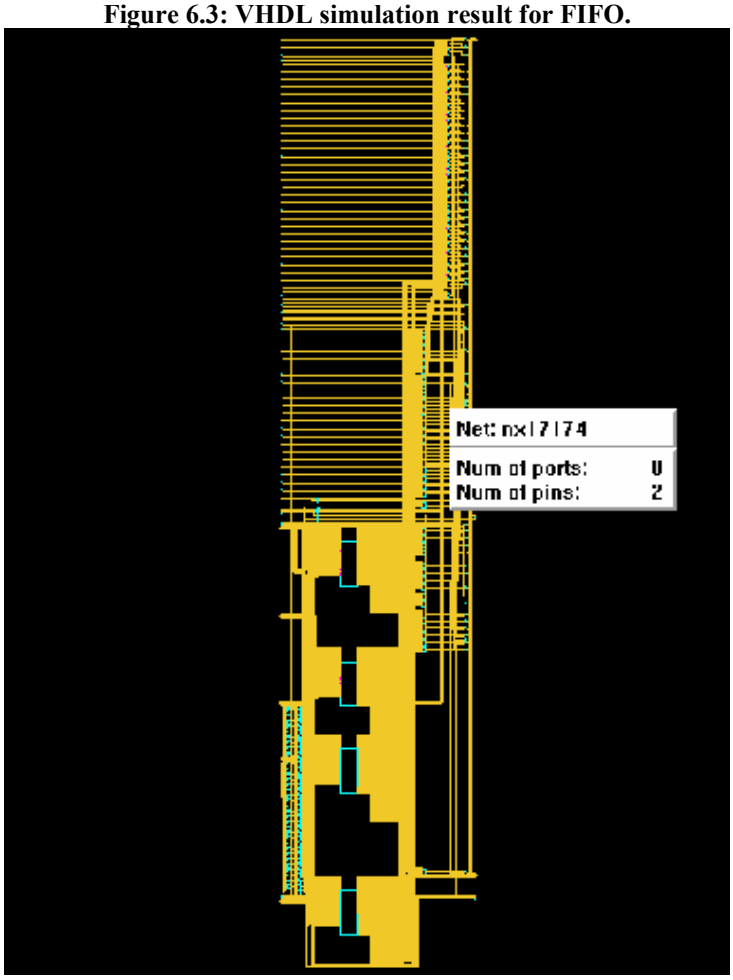


Figure 6.4: Layout of FIFO unit.

6.4 GFP Interface

GFP interface communicates between the FIFO on the Multiplexer side and the GFPT core.

Data from FIFO is passed to the GFP core.

The error indication and K indication is decoded in this module and given as separate signals to the GFP core.

Also the CID and UPI values are sent to GFPT core according to the channel coming from FIFO.

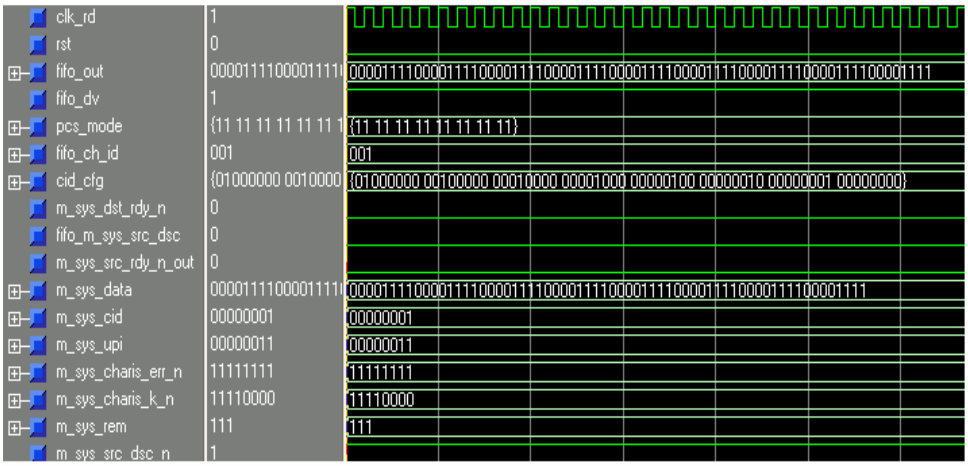


Figure 6.5: VHDL simulation result for GFP interface.

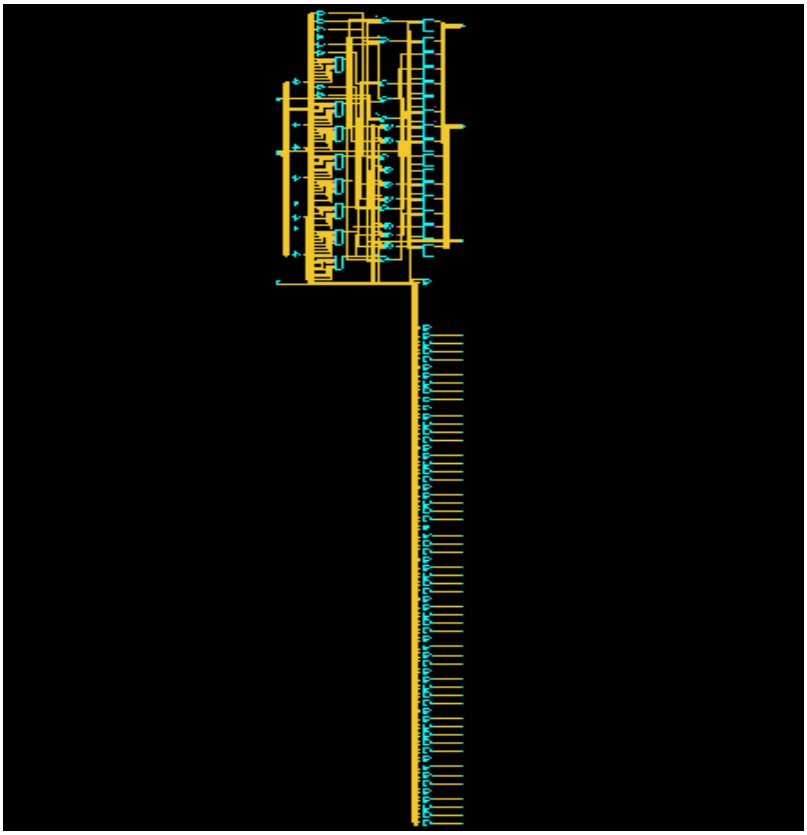


Figure 6.6: Layout of GFP unit.

6.5 GFP-T Mapper Top

This is the structural top for the GFP-T Mapper.

Figure 6.7: Layout of GFP-T Frame Mapper.

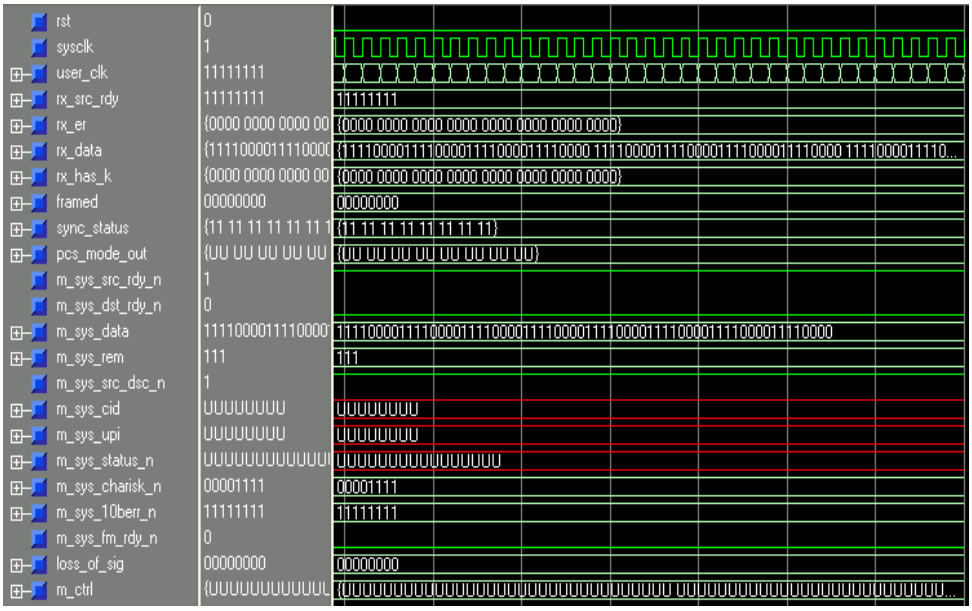


Figure 6.8: VHDL simulation result for GFP-T Mapper.

6.6 Leonardo Spectrum Report

The Leonardo Spectrum report revealed various parameters like number of gates being used in different entities, number of ports & number of instances being used. It also tells about various frequencies that can be given to sys_clk and user_clk which are governed by critical paths present in the circuit.

Table 6.1 specifies the total accumulated area.

Table 6.2 gives the type and number of cells used in the design. It also tells about the number of gates used.

Table 6.1: Total accumulated area

Number of CONZ0 :	10
Number of CONZ1 :	10
Number of gates :	53215
Black Box ramb16_s36_s36	32

Table 6.2: Leonardo spectrum report for cell: gfpt_mux_top

Cell	Library	References	Total	Area
CONZ0	xcl05u	1 x 1	1	CONZ0
CONZ1	xcl05u	1 x 1	1	CONZ1
TS1R1	xcl05u	224 x 5	1142	gates
tx_cpcs_interface	work	8 x 476	381	2 gates
tx_fifo_top	work	1 x 32	32	ramb16_s36_s36
		46342	46342	gates
		8	8	CONZ1
		8	8	CONZ0
tx_gfp_interface	work	1 x 1919	1919	gates
		1	1	CONZ0
		1	1	CONZ1
Number of ports:		1017		
Number of nets:		1217		
Number of instances		236		
Number of references to this view		: 0		

Table 6.3 specifies the clock frequency report.

Table 6.3: Clock frequency report

Clock	Frequency
sys_clk	108.7 MHz
user_clk(7)	172.6 MHz
user_clk(6)	172.6 MHz
user_clk(5)	172.6 MHz
user_clk(4)	172.6 MHz
user_clk(3)	172.6 MHz
user_clk(2)	172.6 MHz
user_clk(1)	172.6 MHz
user_clk(0)	172.6 MHz

Chapter 7

CONCLUSIONS AND FUTURE SCOPE

7.1 Conclusion

To adapt traffic from higher-layer client signals over an octet synchronous transport network encapsulation techniques are required. Generic Framing Procedure (GFP) is such an encapsulation technique which can help to map Ethernet data over SDH/SONET/OTN networks. Xilinx has provided a GFP-T core which can perform the encapsulation and decapsulation of data. The requirement of the present work was to send eight serial channels on a GFP-T stream. Incoming streams were taken to be of type Gigabit Ethernet, Fiber channel (1 GB or 2 GB) or ESCON. The GFP-T core was not able to perform 8b/10b encoding/decoding, thus before feeding the data to the GFP-T core there was a need to convert the serial data to the parallel (8-bit). For this purpose, Xilinx CPCS Core, which performs serial to 10-bit conversion & 8b/10b encoding/decoding, was used. For each incoming channel, one CPCS core was instantiated. Next step was to connect these two-xilinx cores (CPCS and GFP). There were eight streams coming from CPCS (each stream per CPCS core) and GFP core interface required a single stream.

Consequently, in the work undertaken, an architecture was proposed which can be used to interface these two cores. This architecture was “The GFP-T Frame Mapper” which was implemented in hardware using VHDL language. The entire architecture was a group of subsystems each having its own functionality. A total of three entities were developed, namely, the CPCS interface, the FIFO and the GFP interface. The CPCS interface helped the CPCS core and the FIFO to exchange data between them. FIFO stored the data from eight different ports and mapped it onto a single GFP channel. This was actually an 8:1 multiplexing in its complex form. GFP interface facilitated data exchange between FIFO and GFP-T framer (xilinx GFP core). In order to prevent the loss of data there was a need to calculate the number of block RAMs in a FIFO which will help to map the data coming from CPCS core onto GFP-T core, which was successfully done.

The hardware implementations have lead to a successful newer perspective as the implementation was confined to dedicated system architecture. Depending upon the combination of the port stream, the number of Block RAMs required for each port has

been calculated. This is the number of Block RAMs with which the incoming channel can be mapped onto the GFP-T channel without losing any data. Taking the maximum number of super blocks required for a channel for the worst-case combination, required number of Block RAMs has been allocated. With this number of Block RAMs, any port can manage all types of data streams. This implementation is ideal. Table 4.1 gives the number of Block RAMs for any client stream combinations. Thus this thesis provides an approach with which incoming channels can be mapped onto GFP-T core without any loss of data, without the introduction of noise and with optimal number of super blocks per frame.

7.2 Future Scope

For the implementation proposed in this thesis, Block RAM usage is more. The number of Block RAMs directly depends on the optimal number of superblocks (N_{opt}) of all the eight ports. N_{opt} directly depends on the number of GFP overhead bits (GFPOH), number of management frames sent (N_{CMF}), length of the client management frames (CMFL) and Client signal data rate with fastest end of the client clock tolerance ($CSBW_{max}$). N_{opt} is inversely proportional to the transport channel bandwidth with slowest end of the transport clock tolerance ($ChBW_{min}$). Thus by increasing the $ChBW_{min}$, N_{opt} can be decreased, thereby decreasing the number of Block RAMs utilised. So, further improvements can be made by allocating more bandwidth per channel which will decrease the number of superblocks to be transmitted in a frame. If this approach is used, number of Block RAMs required will automatically decrease.

REFERENCES

[1] ITU-T Rec. I.432.1, "B-ISDN User-Network Interface — Physical Layer Specification," 1993.

[2] ISO/IEC 8802-3: 2000 (E), "Information Yechnology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications," 2000.

[3] P802.3ae/D3.1, "Draft IEEE Supplement to ISO/IEC 8802-3 (IEEE 802.3) — Media Access Control (MAC) Parameters, Physical Layer, and Management Parameters for 10 GB/s Operation."

[4] ISO/EIC 3309:1991(E), "Information Technology – Telecommunications and Information Exchange between Systems — High-Level Data Link Control (HDLC) Procedures — Frame Structure," 4th ed., 1991.

[5] "American National Standard for Telecommunications — Synchronous Optical Network (SONET): Physical Interfaces Specifications", ANSI T1.105.06, 2000.

[6] ITU-T Rec. G.707, "Network Node Interface for the Synchronous Digital Hierarchy (SDH)," 1996.

[7] ITU-T Rec. G.709, "Interfaces for the Optical Transport Network (OTN)," 2001.

[8] J. Carlson, P. Langner, J. Manchester, and E. Hernandez- Valencia, "The Simple Data Link (SDL) Protocol," IETF RFC 2823, May 2000.

[9] IEEE Standard 802.3-2002, Information Technology – Telecommunications and Information Exchange Between Systems – LAN/MAN – Specific Requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.

[10] ITU-T Rec. G.7041/Y.1303, "Generic Framing Procedure (GFP)," 2003.

[11] M. Scholten, "Overview of Generic Framing Procedure (GFP)," OIF 2001-172, Apr. 2001.

[12] GFP_ds303.pdf - Xilinx product specification for GFP core.

[13] Xapp759.pdf - Xilinx application notes for CPCS.