STUDY AND SIMULATION OF CCS#7 PROTOCOL USING SYSTEM VERILOG

Samvit Bharti  
B.E., North Gujarat University, India, 2006  

Japan Gandhi  
B.E., Pune University, India, 2006

PROJECT

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND ELECTRONIC ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SUMMER 2010
Student: Samvit Bharti

Japan Gandhi

I certify that these students have met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

______________________________, Department Chair  
Suresh Vadhva, Ph.D  
Date

Department of Electrical and Electronic Engineering
Abstract

of

STUDY AND SIMULATION OF CCS#7 PROTOCOL USING SYSTEM VERILOG

by

Samvit Bharti

Japan Gandhi

Common Channel Signaling #7 (also known as CCS#7 and SS#7) is widely used signaling protocol for the telecommunication as well as satellite communication. CCS#7 protocol uses packets instead of signals to minimize the latency between signaling points. CCS#7 comprises of various layers such as Message Transfer Protocol-1 (MTP-1), MTP-2, MTP-3 and User Part in its architecture. The objective of this project is to study and implement this protocol using System Verilog.

This project is divided into various phases. The first phase includes the study of architecture and functionality of the protocol. The second phase includes the study of the error detection and correction methods employed in the MTP-2 layer. The third phase includes the design and simulation of the packet transmission between two signaling points. The implementation of the design is executed using the System Verilog Hardware Description Language (HDL) and uses Synopsys VCS and Design Vision tool for its compilation, debugging and functional verification.

The design is divided into various small modules such as the Transmitter, Receiver, interfaces between User Part and MTP-3 layer as well as interface between two signaling
points. Each part is designed using the Finite State Machines (FSM). The functionality for the design is verified in a test bench environment with different test cases. The results of simulation have been verified to be working accurately according to the specifications of the protocol.

______________________, Committee Chair
John Balachandra, Ph.D

_______________
Date
ACKNOWLEDGMENTS

Firstly, we would like to thank Professor John Balachandra for his constant support and guidance. Without the help provided by him, this project would not have been possible. In addition, he also took great efforts in showing us the right path through his immense industry exposure whenever we were stuck in this project.

In addition, we would like to thank Professor Manish Gajjar for his valuable guidance in designing the protocol and writing the project report. We would also like to share our gratitude to Dr. Suresh Vadhva, graduate coordinator of the Electrical and Electronic Engineering Department, for his valuable feedback, cooperation, and support. Finally, we are thankful to all faculty members of Electrical and Electronic Engineering Department for helping us to accomplish our requirements for graduation at California State University, Sacramento.
# TABLE OF CONTENTS

List of Tables........................................................................................................................................... ix  
List of Figures............................................................................................................................................... x  

Chapter  
1. INTRODUCTION................................................................................................................................. 1  
   1.1 Channel Associated Signaling........................................................................................................ 1  
   1.2 Common Channel Signaling........................................................................................................... 3  
   1.3 Statement of Collaboration ........................................................................................................... 5  
2. SIGNALING NETWORK COMPONENTS.............................................................................................. 6  
   2.1 STP (Signal Transfer Point)........................................................................................................... 7  
   2.2 SCP (Signal Control Point)........................................................................................................... 8  
   2.3 SSP (Service Switching Point) and SP (Service Point).............................................................. 8  
3. ARCHITECTURE OF CCS#7.................................................................................................................... 9  
   3.1 Overview of CCS#7 Architecture.................................................................................................... 9  
   3.2 Message Transfer Part-1 (Signaling Data Link).............................................................................. 11  
   3.3 Message Transfer Part-2 (Signaling Link)...................................................................................... 13  
      3.3.1 Signaling Unit Types ............................................................................................................... 13  
      3.3.2 Description of Fields in Signaling Unit ..................................................................................... 14  
      3.3.3 Initial alignment ....................................................................................................................... 19  
      3.3.4 Error Monitoring in Signaling Link ......................................................................................... 19  
      3.3.5 Signaling Unit Delimitation ..................................................................................................... 19  
      3.3.6 Signaling Unit Alignment ........................................................................................................ 21  
      3.3.7 Error Detection ......................................................................................................................... 21  
      3.3.8 Error Correction ....................................................................................................................... 21  
         3.3.8.1 Basic Error Correction by ACK/NOK Protocol ................................................................. 22  
         3.3.8.2 Preventive Cyclic Retransmission ....................................................................................... 23  
      3.3.9 Actions Taken by Transmitter’s MTP-2 ................................................................................... 24  
      3.3.10 Actions Taken by Receiver’s MTP-2 ..................................................................................... 26  
      3.3.11 Example of Basic Error Recovery ......................................................................................... 27  
      3.3.12 Signaling Link Error Monitoring ............................................................................................ 28  

3.3.13 Flow Control........................................................................................................... 29

4. IMPLEMENTATION OF CCS#7 ................................................................................ 30
   4.1 Transmitter ........................................................................................................... 30
   4.2 Receiver ............................................................................................................... 38
   4.3 Interface from Application to MTP ................................................................. 44
   4.4 Interface from MTP to Application ................................................................. 45
   4.5 Interface between Signaling Points ............................................................... 47
   4.6 Testbench ......................................................................................................... 47

5. SIMULATION OF CCS#7 ......................................................................................... 53
   5.1 Simulation Result from VCS .............................................................................. 53
   5.2 Simulation Results from DVE .......................................................................... 55

6. CONCLUSION ............................................................................................................. 69

Bibliography ...................................................................................................................... 70
LIST OF TABLES

1. Table 1 Difference between CAS and CSS ................................................................. 4
2. Table 2 Status field messages .................................................................................. 15
3. Table 3 Service information ...................................................................................... 16
4. Table 4 Network information .................................................................................... 17
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 1 PCM structure</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2 CAS multi frame</td>
<td>3</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 3 CCS#7 network structure</td>
<td>7</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 4 Hierarchy of signaling link set</td>
<td>8</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 5 Function diagram of CCS#7</td>
<td>9</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 6 Comparison of CCS#7 with OSI model</td>
<td>10</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 7 MTP-1 connection between two SP</td>
<td>12</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 8 Signaling unit types</td>
<td>14</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 9 Complete MSU</td>
<td>18</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 10 Bit stuffing</td>
<td>20</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 11 Pull back method</td>
<td>23</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 12 Transmitter’s action for new signaling unit</td>
<td>24</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 13 Transmitter’s action for retransmission of signaling unit</td>
<td>25</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 14 Receiver’s action for obtaining the signaling unit</td>
<td>26</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 15 Example of basic error recovery</td>
<td>27</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 16 Leaky Bucket for SUERM</td>
<td>28</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

Alexander Graham Bell was the first human on earth to discover the device and methodology through which a person from one part of the world can communicate with the other person residing in the different subcontinent. This historical device was termed as telephone, which brought a revolution on the planet earth. The idea was to convert the human speech into distinguished electrical signals in order to transmit those over a distance and then to decode those into the original message in terms of voice at the receiver end.[10]

In the later years, it was found that the transmission of these analog signals over a larger distance resulted into a distorted output due to the amplification of the unwanted noise along with the original signal. Therefore, communication system shifted towards the digitized format, which leads to the generation of several protocols like Channel Associated Signaling (also known as CAS) and Common Channel Signaling (also known as CCS#7 and SS#7).[1][3][10]

1.1 Channel Associated Signaling:
This protocol uses the digital format for the transmission of signals from the transmitter to the receiver. As the name suggests, the signals are transmitted through dedicated channels and hence uses the specific circuits for the individual signals. In this protocol,
the data to be transferred as well as the control signals for the same are transmitted over the same dedicated channel. This restricts the channel utilization and hence this methodology is inefficient in delivering an optimum throughput. To overcome these disadvantages, a new protocol was discovered called the Common Channel Signaling.[1][10][11]

![PCM structure](image)

1 frame = 125 μsec

<table>
<thead>
<tr>
<th>Bitrate of 1 channel</th>
<th>Bitrate of the link</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \frac{8 \text{ bits}}{125 \mu\text{sec}} = 64 \text{ Kbit/sec} ]</td>
<td>[ 32 \times 64 \text{ Kbit/sec} = 2,048 \text{ Mbit/sec} ]</td>
</tr>
</tbody>
</table>

Figure 1: PCM structure[1]
1.2 Common Channel Signaling:

This protocol as compared to CAS uses the common channel for the transmission of the voice and the control signals. Although, the voice signal and the control signal unlike the CAS, are distinguished with the usage of software in this protocol. This leads to the increase in the bandwidth and overall throughput of the communicating signals in CCS protocol over CAS protocol. The table 1 shows the comparison between the Channel Associated Signaling and the Common Channel Signaling.[1][10][11]
Table 1: Difference between CAS and CCS[1]

Limitations:
Although, common channel signaling has overcome several drawbacks of analogue signaling such as limited set of signals, in-band signals due to the dedicated design of the architecture itself and the overall device failure as had a low throughput. On the contrary, common channel signaling also have few drawbacks. This is a software-based protocol and hence always has to be updated with respect to the changes brought into the method adapted for the provision of service. This protocol has a complex hardware. Lastly, this protocol has a loophole in terms of security when operated with the collaboration of the third party.[1][3][9][10]
Application:

Signaling System No. 7 (SS7) is a set of telephony signals protocols, which are used to set up most of the world's public switched telephone network telephone calls. The main purpose is to set up and tear down telephone calls. Other uses include number translation, prepaid billing mechanisms, short message service (SMS), and a variety of other mass-market services.[1][9][10]

1.3 Statement of Collaboration:

This project is entirely studied by both the project partners. We have been through various books and articles specified in the bibliography for studying the architecture and functionality of this protocol. The designing for the protocol is divided into two parts namely the encoder and the decoder. The encoder is designed and implemented by Samvit Bharti while the decoder is designed and implemented by Japan Gandhi. Both of us had an equal contribution towards the writing process for the report.
The communication from a source caller to the destination caller is routed through several points or exchanges. In addition, the data transferred in the form of voice or message follows a specific set of rules called the protocol. In CCS7 network, the calls from the source caller are collected at the local exchanges, which are also called as the Signaling End Point (SEP’s). These Signaling End Points group the signals into the specific Signal Transfer Point (STP), which indeed communicate with one another and then transfer the signal to the corresponding Signaling End Point. This SEP indeed transfers the data to the destination caller. CCS7 protocol is used during the transfer of data from one point to another. This is the basic flow of the routing process as shown in the figure below. The basic components and its functionality are explained later in this chapter.[1][9][10][11]
2.1 STP (Signal Transfer Point):

The basic functionality of these Signaling Transfer Point is to transfer the data to another STP in the channel by adding the configurations of the signal and the amount of data transferred in order to ensure correct transmission of the signals. The signals are transmitted on a specific path between any two Signaling Transfer Points and are called the Signaling Link. These Signaling Links are grouped together to form Signaling Groups which group together to form a Signaling Link Set. In other words, signaling link is a subset of signaling groups, which indeed are subsets of signaling link set as shown in the figure below.[1][2][9][10]
2.2 SCP (Signal Control Point):
SCP’s function in the same manner as the network layer in the TCP/IP protocol. It controls the rate of data transfer between the two points in order to maintain the consistency of transfer. This stops the faster receiving point to starve from the slow transferring point. Also, the error codes and their methodology is maintained by this device.[1][2][10][11]

2.3 SSP (Service Switching Point) and SP (Service Point):
In the currently technology, SSP’s and SP’s are most widely used as the signaling exchanges due to its large channel handling capacity. Their function is to transfer the data from the caller to the local Signaling Transfer Point. The SSP’s service the incoming or outgoing call by forwarding, its data to the STP’s in the network.[1][2][10][11]
Chapter 3

ARCHITECTURE OF CCS#7

To make CCS#7 modular, CCS#7 structures are divided into Message Transfer Part (MTP) and User Part (UP). MTP is divided further into MTP-1, MTP-2 and MTP-3.

Figure 5: Functional diagram of CCS#7[1]

3.1 Overview of CCS#7 Architecture:

1. MTP-1: This layer has similar functionality as Physical Layer in Open System Interconnect (OSI) model[1]. It connects Signaling Points (SP) and transmit/receive bit signals to capture data.
2. MTP-2: This layer has similar functionality as Data Link Layer in OSI model\cite{1}. This layer is responsible for error free reception of data by performing error detection via Cyclic Redundancy Check (CRC) and error correction via ACK/NOK protocol.

![Comparison of CCS#7 with OSI model](image)

3. MTP-3: This layer has similar functionality as Network Layer in OSI model. This layer is responsible for routing, flow control and distribution of data message.
Also this layer is responsible for link configuration and reconfigurations in erroneous situation[1].

4. User Part: User Part sends message to MTP layer and MTP layers ensure delivery of message to destination. Many User Parts exist in CCS#7 like ISUP (ISDN User Part) for ISDN network, TUP (Telephone User Part) for basic telephone services, DUP (Data User Part) for data networks.

3.2 Message Transfer Part-1 (Signaling Data Link):

MTP-1 is a bi-directional differential transmission path consists of two message channels operating in bi-directionally simultaneously at same data rate[1]. MTP-1 is the lowest functional layer also known as physical layer in CCS#7 hierarchy. MTP-1 is described completely in ITU-T’s standard - ITU-T Q.702. MTP. Transport medium is irrelevant in signaling. In old days, copper cables are used as transmission medium. As bandwidth demand increases, now we use optical fibers as transmission medium.[1]

The message channels which carry CCS#7 signals must be exclusively used only for signaling as CCS#7. Voice enhancer equipments used in telecom such as A/u law encoder/decoder, echo cancelling equipments, etc must not be used for signaling channel.[1]

The standard channel 16 is used for signaling between two SP. However if channel 16 is not available other channels can be used, but the channel connection between two SP
must be same. i.e. If channel 15 is used for signaling, both connected SP must use channel 15 for signaling in switch configuration. To maximize reliability, 1+1 configuration (two signaling links) is used in signaling so if one signaling link or PCM link fails other signaling link can take over the traffic.

Many PCM links are connected between two SP depending on traffic. One signaling link can handle about 1000 voice channels (around 32 E1 and 43T1). If the signaling link fails then all the traffic will be affected. To ensure reliability, two signaling links are used. This is illustrated in Figure 7.

Figure 7: MTP-1 connection between two SP[1]
3.3 Message Transfer Part-2 (Signaling Link):

Similar to Data Link Layer in OSI model, MTP-2 ensures reliable signaling transfer between two directly connected SP. MTP-3 transfers Message Signaling Units (MSU) to MTP-2 in variable length and MTP-2 form packet and send it to MTP-1 for transmission. The main functions of MTP-2 are initial alignment, monitoring in signaling link, signaling unit alignment and bit stuffing, error detection and correction, and flow control[1]. Before we continue details on each function, we will have a look at CCS#7 signaling unit types and fields of CCS#7.[1]

3.3.1 Signaling Unit Types:

There are three types of signaling units:

a. Fill-in State Unit (FISU)

When there is no payload to transfer or no link status messages to transfer to destination signaling point, empty signaling units are sent which are called as FISU[1]. By sending FISU, signaling points monitor the errors.

b. Link Status Signal Unit (LSSU)

LSSUs are used for maintaining signaling link alignment and it is used for flow control.

c. Message Signal Unit (MSU)

MSU transfers signaling information from signaling point’s user part to another signaling point’s user part[1].
3.3.2 Description of Fields in Signaling Unit:

a. F (Start Flag):

Signaling unit always starts with an 8-bit start flag “01111110” which is used to separate two signaling units[1]. There is no end flag in the system.

b. BSN (Backward Sequence Number) and FSN (Forward Sequence Number):

BSN is 7-bit sequence number, which indicates previously accepted signaling unit. FSN is the sequence number, which indicates current signaling unit’s sequence number.[1]

c. BIB (Backward Indicator Bit) and FIB (Sequence Number):
BIB and FIB are one bit field which controls the error flag. Transmitter sends packet with current FIB bit. BIB indicates the signaling unit with BSN is whether accepted or retransmission is requested. When a packet is received unsuccessfully receiver send previous BSN with inverted BIB. For example, when a packet with FSN=5 and FIB=1 is received with error, the receiver side will send BSN=4 and BIB=0. In summary, if BIB and FIB are same then the packet is accepted and if BIB and FIB is not same then the packet is requesting retransmission.

d. LI (Length Indicator):

Length indicator is 6-bits field, which indicates number of bytes after LI field and before CRC field.[1][9]

i. LI = 0 for FISU[1]

ii. LI = 1 or 2 for LSSU[1]

iii. LI ≥ 3 for MSU[1]

e. SF (Status Field): LSSU contains 8 or 16 bits status field, which indicates link status of present link[1][9].

<table>
<thead>
<tr>
<th>Bits</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Out of Alignment</td>
</tr>
<tr>
<td>001</td>
<td>Normal Alignment</td>
</tr>
<tr>
<td>010</td>
<td>Emergency Alignment</td>
</tr>
<tr>
<td>011</td>
<td>Out of Service</td>
</tr>
</tbody>
</table>

Table 2: Status field messages[9]
f. SIO (Signaling Information Octet):

SIO is a 8-bit field which is a part of MSU containing information about 4-bit Service Indicator (SI) and 4-bit Network Indicator. Service Indicator contains information about User Part (Application) protocol used for signaling unit.[1][9]

<table>
<thead>
<tr>
<th>Bits</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Process Outage</td>
</tr>
<tr>
<td>101</td>
<td>Busy</td>
</tr>
</tbody>
</table>

Table 2: Status field messages[9]

<table>
<thead>
<tr>
<th>Bits</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Out of Alignment</td>
</tr>
<tr>
<td>001</td>
<td>Normal Alignment</td>
</tr>
<tr>
<td>010</td>
<td>Emergency Alignment</td>
</tr>
<tr>
<td>011</td>
<td>Out of Service</td>
</tr>
<tr>
<td>100</td>
<td>Process Outage</td>
</tr>
<tr>
<td>101</td>
<td>Busy</td>
</tr>
</tbody>
</table>

Table 3: Service information[9]
<table>
<thead>
<tr>
<th>Bits</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>00XX</td>
<td>International</td>
</tr>
<tr>
<td>01XX</td>
<td>International</td>
</tr>
<tr>
<td>10XX</td>
<td>National</td>
</tr>
<tr>
<td>11XX</td>
<td>National</td>
</tr>
</tbody>
</table>

Table 4: Network information[9]

g. SIF (Signaling Information Field) or Message: SIF are actual signaling messages along with routing information in MSU.[1][9]

h. Check bits or CRC (Cyclic Redundancy Check): A 16-bit CRC is used to detect error in signaling unit.[1][9]

i. Spare fields and do not care fields are filled with 0.
Figure 9: Complete MSU[1]

- **FLAG**: Message Info
- **FCS**: Message Type Code
- **CIC**: Trunk Nr. Channel
- **OPC**: SIO
- **DPC**: LI
- **FSN**: BSN
- **BSN**: FLAG

**Signalling Information Field**

- First Transmitted Bit
- **Routing Label**: n x 8, 4, 4, 12

**Network Indicator**

- **International**: 00XX
- **National**: 10XX, 11XX
- **Local**: 01XX

- **Flag**: 0111110
- **FISU**: LI=0
- **LSSU**: LI=1 or 2
- **Normal MSU**: 2<LI<63

**MTP-2**
1. Recognizes and Generates Flags
2. Bit Stuffing
3. Error Check
4. Receiver & Transmitter Control

**MTP-3**
- Message Routing
- Message Distribution

**User Part**

- **BSN**: Backward Sequence Number
- **BIB**: Backward Indication Bit
- **FSN**: Forward Sequence Number
- **FIB**: Forward Indication Bit
- **LI**: Length Indicator
- **SIO**: Service Information Octet
- **DPC**: Destination Point Code
- **OPC**: Originating Point Code
- **SLS**: Signalling Link Selector
- **CIC**: Circuit Identification Code
- **FCS**: Frame Check Sequence
- **0000**: Sign, Network Mngt.
3.3.3 Initial Alignment:

This procedure is used for both first time alignment and link restoration after link failure. During initial alignment, status information is exchanged for proving period to ensure error-free link. Between two signaling point, only the required signaling link will be aligned and no other signaling links are take out of service.[1][9]

3.3.4 Error Monitoring in Signaling Link:

Errors are monitored in signaling link via two functions:

a. Alignment error rate monitor: While the link is in proving state (initial alignment), this function will check for errors and if errors are higher than threshold value then link is taken out of service.[1][9]

b. Signaling unit error rate: While the link is in service, this function will check for errors and if errors are higher than threshold value then link is taken out of service.[1][9]

3.3.5 Signaling Unit Delimitation:

The signaling unit is always started with 8-bit pattern also known as start flag (01111110)[1]. Please note that there is no end flag. Receiver will continuously search for start flag. If six consecutive 1s are detected then the receiver will treat it as start of signaling unit[1]. Problem may arise if other fields in signaling unit may contain six consecutive 1s pattern. This problem is solved by bit stuffing. In bit stuffing procedure, if in signaling unit five consecutive 1s are detected then it will add one zero[1]. So all the
field except start flag will not consist consecutive six 1s pattern. Bit stuffing technique is illustrated in figure 10.[1][9]
3.3.6 Signaling Unit Alignment:
When there are no 1s density violation and all signaling units received is in sequence then signaling link is considered as normal aligned[1]. Total length of SIO must be less than 272 bytes. If any error occurs then the erroneous signaling unit is discarded. Total length of SIO must be less than 272 bytes. When these errors exceed the threshold of alignment, link will be taken out of service and signaling link enters into re-alignment of signaling link[1].[9]

3.3.7 Error Detection:
A 16-bit CRC is used to detect error in signaling unit. CRC is calculated from first bit of BSN to last bit of message. CCITT 16-bit polynomial - $x^{16} + x^{12} + x^{5} + 1$ is used to calculate the CRC[1]. CRC is calculated before bit stuffing. Check bits are included at the end of packet. At receiver end, when signaling unit is received after bit stuffing, CRC is evaluated and if it matches with check bits message will get accepted (if $FSN_{received}$ is $FSN_{expected}$).[1]

3.3.8 Error Correction:
Error correction in CCS#7 is maintained by two methods.

1. Basic error correction by ACK/NOK protocol
2. Preventive cyclic retransmission

3.3.8.1 Basic Error Correction by ACK/NOK Protocol:

In this method, error correction relies on positive/negative acknowledgements to sequence number[1]. Upon successful received signaling unit, positive acknowledgement is sent back to transmitter (ACK) and retransmission of errored signaling units is requested by negative acknowledgement (NOK). Main components of basic error corrections are: retransmission buffer, FSN/BSN and FIB/BIB for ACK/NOK protocol.[1]

Retransmission buffer is a FIFO buffer in each CCS7 transmitter, which stores every signaling unit in retransmission buffer. When signaling unit is received successfully at other end, it will send ACK for received signaling unit and transmitter will delete that signaling unit from retransmission buffer. And upon receival of NOK for signaling unit, the signaling unit will be retransmitted from retransmission buffer. In contrast, to selective retransmission method where selected signaling unit is sent; if a retransmission is requested, all the signaling units in the retransmission buffer will be retransmitted in original order, hence this method is also known as Pull Back Method[1].
3.3.8.2 Preventive Cyclic Retransmission:

Preventive cyclic retransmission method is normally used in higher propagation delay medium (>15msec)[1]. If the propagation delay is very high, the round trip delay will be very much higher so waiting for ACK/NOK will reduce bus utilization. MSUs in retransmission buffer will be sent automatically during idle time when no more new MSUs are being transmitted. This method will increase the bus utilization and also improves
performance because receiver might get erroneous MSU before it sends NOK or before the transmitter gets NOK. This method is used in addition to retransmission of NOK MSUs[1].

3.3.9 Actions Taken by Transmitter’s MTP-2:

![Flowchart](image-url)

Figure 12: Transmitter’s action for new signaling unit[1]
Figure 13: Transmitter’s action for retransmission of signaling unit[1]
3.3.10 Actions Taken by Receiver’s MTP-2:

Figure 14: Receiver’s action for obtaining the signaling unit[1]
3.3.11 Example of Basic Error Recovery:

Let’s take an example of basic error recovery. Transmitter has sent two packets with sequence number 51 and 52, and are stored in retransmission buffer waiting for the ACK/NOK from receiver. Now third packet with sequence number 53 is transmitted to receiver and at the same time will be stored in the retransmission buffer.[1]

![Diagram of basic error recovery](image.png)

> Figure 15: Example of basic error recovery[1]

When receiver successfully receives packet 53, it will check for CRC and if the CRC check is true then the receiver will give ACK for 53 packet and transmitter will delete all three packets from retransmission buffer.[1]
Now if the receiver unsuccessfully receives packet 53, the CRC check will fail. So the receiver will send BSN 53 and inverted BIB to transmitter. So transmitter will know that up to 52 sequence number packets are received successfully and packet 53 is erroneous. Hence, transmitter will delete 51 and 52 from retransmission buffer and re-transmits the packet 53.[1]

3.3.12 Signaling Link Error Monitoring:

There are two error monitors in CCS#7 MTP.

1. Signaling Unit Error Rate Monitor (SUERM)

While signaling link is in service SUERM is used. SUERM is a counter, which is incremented when erroneous SU is received and decremented at fixed rate[1]. So if at given period, increments of counter are much higher than decrement count, excessive error rate indication is sent to MTP-3[1]. The counter starts from 0 after reset.[1]

![Figure 16: Leaky bucket for SUERM][1]
This phenomenon is also called as “Leaky Bucket”. Erroneous SU are the inputs of bucket and buckets are emptied at constant flow. When threshold T reaches, the excessive error rate indication is sent to MTP-3.[1]

2. Alignment Error Rate Monitor (AERM)

This monitor is used during alignment procedure of signaling link. It is a simple counter mechanism. At proving stage when signaling link is getting alignment, this counter starts from zero and the counter is incremented for every erroneous SU[1]. When counter reaches certain threshold, the link is taken out of service.[1]

3.3.13 Flow Control:

CCS#7 also implements flow control for congestion over network. Upon receiving the congestion from receiver all MSUs and both positive and negative acknowledgements are suspended. LSSU with B (Busy) state sent to receiver[1]. This will initiate rerouting of messages.[1]
4.1 Transmitter:

```verilog
module transmitter (
    app_mtp bus,     //Signals from Application to MTP
    layer
    sp1_sp2.sp1 t_packet  //Physical layer signals
);

parameter start_flag=8'b01111110;

typedef struct packed {
    logic [15:0] fcs;    //fcs=packet Check Sequence
    logic [0:63][7:0]message;  //message
    logic [7:0]mtc;    //mtc=message Type Code
    logic [15:0]cic;    //cic=Circuit Identification Code
    logic [3:0]sls;    //sls=Signaling Link Selector
    logic [13:0]opc;    //opc=Originating Point Code
    logic [13:0]dpc;    //dpc=Destination Point Code
    logic [7:0]sio;    //sio=Signaling Indicator Octet = ssf+si
    logic [7:0]li;     //li=Length Indicator
    logic fib;      //fib=Forward Indicator Bit
    logic [6:0]fsn;    //fsn=Forward Sequence Number
    logic bib;      //bib=Backward Indicator Bit
    logic [6:0]bsn;    //bsn=Backward Sequence Number
    logic [7:0]flag;    //packet
} packet;

packet current_packet;
packet retransmission[0:15];
```
### FSM

```plaintext
eenum {
  reset_state=3'b000,
  idle_state=3'b001,
  packet_formation_state=3'b010,
  packet_seq_state=3'b011,
  packet_crc_state=3'b100,
  packet_transmit_state=3'b101,
  packet_retransmit_state=3'b110
}
cs, ns;
```

### CRC

```plaintext
parameter crc_poly = 16'h1021;
integer i, j;
integer crc_length;
logic [15:0] crc_result;
logic msb;
logic [7:0] current_byte;
logic [15:0] temp;
```

### Bit stuffing

```plaintext
logic [759:0] packet_bitstuffing;
logic [2:0] count;
```

### Sequence Numbers

```plaintext
logic [6:0] bsn, //Backward Sequence Number
  bsn_cnt; //BSN counter to keep track of valid packets received

logic [6:0] fsn, //Forward Sequence Number
  fsn_cnt; //FSN counter

logic bib_prev,bib_next;
logic fib_prev,fib_next;
```

### FIFO

```plaintext
logic [3:0]fifo_read; //FIFO read counter
```
logic [3:0] fifo_write;       // FIFO write counter
logic fifo_full, fifo_empty;  // Full and empty flag for FIFO

///// Other regs /////
logic packet_formation_completed;
logic error;
logic retransmit_flag;

/******************************************************************************
************************ FSM D-FF **************************************/

always_ff @ (posedge bus.clk, negedge bus.reset)
begin
if (!bus.reset)
    cs<=reset_state;
else
    cs<=ns;
end

/******************************************************************************
***************** Input combinational logic *******************************/

always_comb
begin:
    input_comb_block
    unique case (cs)
    reset_state:
        if (bus.valid)
            ns=packet_formation_state;
        else
            ns=reset_state;

    idle_state:
        if (bus.valid)
            ns=packet_formation_state;
        else if (fifo_empty==0)
            ns=packet_retransmit_state;
        else
            ns=idle_state;
packet Formation State:
   ns=packet_seq State;

Packet Seq State:
   ns=packet_crc State;

Packet CRC State:
   ns=packet_transmit State;

Packet Transmit State:
   ns=idle State;

Packet Retransmit State:
   ns=idle State;

default:
   ns=reset State;
endcase
end: input_comb_block

always_comb
begin: output_comb_block
unique case (cs)
reset State:
begin
   /*
   if (bus.valid)
   begin
      //If data is valid then latch data into current packet packet
      current_packet.flag = start_flag;
      current_packet.li = {2'b0,bus.li};
      current_packet.sio = {bus.ssf,bus.si};
      current_packet.dpc = bus.dpc;
      current_packet.opc = bus.opc;
      current_packet.sls = bus.sls;
      current_packet.cic = {4'b0,bus.trunk_no,bus.channel_no};
   */
end: output_comb_block
current_packet.mtc = bus.mtc;
current_packet.message = bus.message;

if ((fifo_write-fifo_read)==4'b1111)
  fifo_full=1;
else if (fifo_write==fifo_read)
  fifo_empty=1;
else
  begin
    fifo_full=0;
    fifo_empty=0;
  end
end

begin //reset all counters and reg to 0
  bsn_cnt=0;
  fsn_cnt=0;
  bib_prev=0;
  bib_next=0;
  fib_prev=0;
  fib_next=0;
  fifo_read=0;
  fifo_write=0;
  fifo_full=0;
  fifo_empty=1;
  retransmit_flag=0;
  packet_formation_completed=0;
end

packet_formation_state:
begin
  if (bus.valid)
    begin
      //If data is valid then latch data into current packet packet
      current_packet.flag = start_flag;
    end
end

idle_state:
begin
end

```haskell
current_packet.li = {2'b0,bus.li};
current_packet.sio = {bus.ssf,bus.si};
current_packet.dpc = bus.dpc;
current_packet.opc = bus.opc;
current_packet.sls = bus.sls;
current_packet.cic = {4'b0,bus.trunk_no,bus.channel_no};
current_packet.mtc = bus.mtc;
current_packet.message = bus.message;

packet_formation_completed=0;
retransmit_flag=0;               //reset retransmit flag

if ( (fifo_write-fifo_read)==4'b1111)
    fifo_full=1;
else if (fifo_write==fifo_read)
    fifo_empty=1;
else
    begin
        fifo_full=0;
        fifo_empty=0;
    end
end

packet_seq_state:
begin
    if (retransmit_flag==0)
    begin
        current_packet.bsn = bsn_cnt;
        current_packet.bib = bib_next;
        current_packet.fsn = fsn_cnt;
        current_packet.fib = fib_next;
        bib_prev = bib_next;
    end

    //Add packet to retransmission buffer (FIFO )
    if (!fifo_full)
    begin
        retransmission[fifo_write]= current_packet;
        fifo_write++;
        fifo_empty=0;
```
fsn_cnt++;  //increment fsn for next transfer
end
else
begin
    error = 1;
end
end

current_packet.li = current_packet.li + 1;
packet_crc_state:
begin

/*************************************************************/
*************  CRC-16 : x16+x12+x5+1(0x1021)  **************
*************************************************************/

    crc_length   = (11+current_packet.li+1)*8;
crc_result=16'hffff;
for (i = 8; i <= crc_length; i = i + 8)
begin
    current_byte = current_packet[i+:8];
    for (j = 0; j < 8; j = j + 1)
begin
        msb = crc_result[15];
crc_result = crc_result << 1;
        if (msb != current_byte[j]) begin
            crc_result = crc_result ^ crc_poly;
crc_result[0] = 1;
        end
    end
end

//Mirror every bit, swap the 2 bytes, and then complement each bit.
for (j = 0; j < 16; j = j + 1)
temp[15-j] = crc_result[j];

//Swap and Complement:
crc_result = ~{temp[7:0], temp[15:8]};
current_packet.fcs = crc_result;

/*************************************************************/
End of CRC  *****************************************************/
packet_bitstuffing[7:0]=current_packet[7:0];

for (i=8,j=8; i<=crc_length+24; i=i+1,j=j+1)
begin
    if (current_packet[i]==1'b1)
        begin
            count++;
            packet_bitstuffing[j]=1'b1;
        end
    else
        begin
            count=0;
            packet_bitstuffing[j]=1'b0;
        end

    if (count==3'b101)
        begin
            j++;
            packet_bitstuffing[j]=1'b0;
            count=0;
        end
end

packet_transmit_state:
begin
    /*******************   Send Packet to MTP-1  ***********************/
    packet_formation_completed=1;
    t_packet.packet = packet_bitstuffing;
end

packet_retransmit_state:
begin
    current_packet=retransmission[fifo_read];
fifo_write--;
retransmit_flag=1;
packet_formation_completed=0;
if (((fifo_write-fifo_read)==0)
    fifo_empty=1;
else
    fifo_empty=0;
end
dcase
end: output_comb_block
endmodule

4.2 Receiver

/****************************** Receiver ******************************/
/****************************** Receiver ******************************/
module receiver (                      //Signals from MTP to Application
    mtp_app bus,                             //Physical layer signals
    layer
    sp1_sp2.sp2 r_packet
);

typedef struct packed {                  //1+11+64+2=78
    logic [15:0] fcs;                     //fcs=packet Check Sequence
    logic [0:63][7:0]message;             //message
    logic [7:0]mtc;                       //mtc=message Type Code
    logic [15:0]cic;                      //cic=Circuit Identification Code
    logic [3:0]sls;                       //sls=Signaling Link Selector
    logic [13:0]opc;                      //opc=Originating Point Code
    logic [13:0]dpc;                      //dpc=Destination Point Code
    logic [7:0]sio;                       //sio=Signaling Indicator Octet = ssf+si
    logic [7:0]li;                        //li=Length Indicator
    logic fib;                           //fib=Forward Indicator Bit
    logic [6:0]fsn;                       //fsn=Forward Sequence Number
    logic bib;                           //bib=Backward Indicator Bit
    logic [6:0]bsn;                       //bsn=Backward Sequence Number
    logic [7:0]flag;                      //packet
} packet;

packet current_packet;

logic [6:0] bsn, fsn;
logic bib, fib;
logic [15:0] fcs;

///// FSM /////
enum {
  reset_state=3' b000,
  idle_state=3' b001,
  packet_received_state=3' b010,
  packet_validate_state=3' b011,
  packet_seq_state=3' b100,
  packet_feedback_state=3' b101
} cs, ns;

///// CRC /////
integer i, j;
parameter crc_poly = 16' h1021;
integer crc_length;
logic [15:0] crc_result;
logic msb;
logic [7:0] current_byte;
logic [15:0] temp;

///// Bit stuffing /////
logic [759:0] packet_bitstuffing;
logic [2:0] count;

///// Sequence Numbers /////
logic [6:0] bsn_cnt;
logic [6:0] bsn_expected;
logic [6:0] fsn_cnt;
logic [6:0] fsn_expected;
logic bib_prev, bib_next;
logic fib_prev, fib_next;
always_ff @ (posedge bus.clk, negedge bus.reset)
begin
    if (!bus.reset)
        cs<=idle_state;
    else
        cs<=ns;
end

always_comb
begin
    case (cs)
        reset_state:
            ns=idle_state;
        idle_state:
            if (r_packet.packet[0]==8'b01111110)
                ns=packet_received_state;
            else
                ns=idle_state;
        packet_received_state:
            ns=packet_validate_state;
        packet_validate_state:
            ns=packet_seq_state;
        packet_seq_state:
            ns=packet_feedback_state;
        packet_feedback_state:
            ns=idle_state;
        default:
            ns=reset_state;
    endcase
end

always_comb
begin
    case (cs)
        reset_state:
            begin
//reset counters to 0
bsn_cnt=0;
fsn_cnt=0;
bib_prev=0;
bib_next=0;
fib_prev=0;
fib_next=0;
bsn_expected=0;
fsn_expected=0;
end

packet_received_state:
begin

/************************   Bit Unstuffing   *********************/

packet_bitstuffing = r_packet.packet;
current_packet[7:0] = packet_bitstuffing[7:0];
for (i=8,j=8; i<=759; i=i+1,j=j+1)
begin
  if (packet_bitstuffing[i]==1'b1)
  begin
    count++;
current_packet[j]=1'b1;
  end
  else
  begin
    count=0;
current_packet[j]=1'b0;
  end
  if (count==3'b101)
  begin
    i++;
count=0;
  end
end

/************************   Bit Unstuffing End  *******************/

/*----------------------------- Bit Unstuffing -----------------------------*/
Assignments to signals

```plaintext
bsn     = current_packet.bsn;
bib     = current_packet.bib;
fsn     = current_packet.fsn;
fib     = current_packet.fib;

bus.li   = current_packet.li;
bus.ssf  = current_packet.sio[7:4];
bus.si   = current_packet.sio[3:0];
bus.dpc  = current_packet.dpc;
bus.opc  = current_packet.opc;
bus.sls  = current_packet.sls;
bus.channel_no  = current_packet.cic[12:5];
bus.trunk_no     = current_packet.cic[4:0];
bus.mtc  = current_packet.mtc;
bus.message      = current_packet.message;

end

packet_validate_state:
begin

fcs     = current_packet.fcs;
```

```
/******************** CRC-16 : x16+x12+x5+1(0x1021) ******************/

```
// Mirror every bit, swap the 2 bytes, and then complement each bit.
for (j = 0; j < 16; j = j + 1)
    temp[15-j] = crc_result[j];

// Swap and Complement:
crc_result = ~{temp[7:0], temp[15:8]};
current_packet.fcs = crc_result;

/**********************    End of CRC   **********************/

if (fcs==crc_result)
    bus.valid = 1;
else
    bus.valid = 0;
end

packet_seq_state:
begin
/******************    Sequence no. check   *******************/
    if(fsn_expected==fsn && fib==bib_prev) // accepted
        begin
            fsn_expected++; // increment fsn for next transfer
            bib_next=fib;
        end
    else // not accepted
        begin
            bib_next=~fib;
        end
end

packet_feedback_state:
begin
/***********************    Feedback   ************************/
    if (bib == fib_prev) // accepted
        begin
        end
    else
        begin
        end
end
begin
end
end
endcase
end
endmodule

4.3 Interface from Application to MTP:

interface app_mtp (input clk, input reset);
logic valid;
logic [5:0]li;
logic [3:0]si;
logic [3:0]ssf;
logic [13:0]dpc;
logic [13:0]opc;
logic [3:0]sls;
logic [4:0]channel_no;
logic [6:0]trunk_no;
logic [7:0]mtc;
logic [0:63][7:0]message;
clocking cb_mtp @ (posedge clk);
  input valid;  //input valid signal
  input li;     //li=Length Indicator
  input si;     //si=Signaling Indicator
  input ssf;    //ssf=Network Indicator
  input dpc;    //dpc=Destination Point Code
  input opc;    //opc=Originating Point Code
  input sls;    //sls=Signaling Link Selector
  input channel_no; //cic=Circuit Identification Code
  input trunk_no; //cic=Circuit Identification Code
  input mtc;    //mtc=Message Type Code
  input message;
endclocking

modport mtp (input clk, input reset, clocking cb_mtp);
modport mtp (  
    input clk,  
    input reset,  
    input valid,  
    input li,  
    input si,  
    input ssf,  
    input dpc,  
    input opc,  
    input sls,  
    input channel_no,  
    input trunk_no,  
    input mtc,  
    input message  
);  
 */

modport app (  
    output valid,  
    output li,  
    output si,  
    output ssf,  
    output dpc,  
    output opc,  
    output sls,  
    output channel_no,  
    output trunk_no,  
    output mtc,  
    output message  
);  
endinterface : app_mtp

4.4 Interface from MTP to Application:

interface mtp_app (input clk, input reset);  
logic valid;  
logic [5:0]li;  
logic [3:0]si;  
logic [3:0]ssf;
logic [13:0]dpc;
logic [13:0]opc;
logic [3:0]sls;
logic [4:0]channel_no;
logic [6:0]trunk_no;
logic [7:0]mtc;
logic [0:63][7:0]message;

clocking cb_mtp @ (posedge clk);
    output valid; //input valid signal
    output li; //li=Length Indicator
    output si; //si=Signaling Indicator
    output ssf; //ssf=Network Indicator
    output dpc; //dpc=Destination Point Code
    output opc; //opc=Originating Point Code
    output sls; //sls=Signaling Link Selector
    output channel_no; //cic=Circuit Identification Code
    output trunk_no; //cic=Circuit Identification Code
    output mtc; //mtc=Message Type Code
    output message;
endclocking

modport mtp (input clk, input reset, clocking cb_mtp);

modport app ( 
    input valid,
    input li,
    input si,
    input ssf,
    input dpc,
    input opc,
    input sls,
    input channel_no,
    input trunk_no,
    input mtc,
    input message
);

endinterface : mtp_app
4.5 Interface between Signaling Points:

interface sp1_sp2;

logic [94:0][7:0] packet;
modport sp1 (  
inout packet  
);

modport sp2 (  
inout packet  
);

endinterface : sp1_sp2

4.6 Testbench:

/********************************************************************
********************************************************************
****** Test Bench **************
********************************************************************
********************************************************************/

//Include all System Verilog files

`include "transmitter.sv"
`include "receiver.sv"
`include "interface_app_mtp.sv"
`include "interface_sp1_sp2.sv"
`include "interface_mtp_app.sv"

module testbench;

//Initialize clock and reset
logic clk,reset;
initial begin
  clk=1;
  reset=0;
  forever #5 clk=~clk;
end
Physical link between Transmitter and Receiver and vice-versa
sp1_sp2 physical_sp1_sp2();
sp1_sp2 physical_sp2_sp1();

App_mtp app_mtp_bus_sp1 (.clk(clk), .reset(reset));
App_mtp app_mtp_bus_sp2 (.clk(clk), .reset(reset));

Mtp_app mtp_app_bus_sp1 (.clk(clk), .reset(reset));
Mtp_app mtp_app_bus_sp2 (.clk(clk), .reset(reset));

//SP1 Transmitter DUT
transmitter DUT_SP1_Trans (.bus(app_mtp_bus_sp1),.t_packet(physical_sp1_sp2));

//SP1 Receiver DUT
receiver DUT_SP1_Recev (.bus(mtp_app_bus_sp1), .r_packet(physical_sp2_sp1));

//SP2 Transmitter DUT
transmitter DUT_SP2_Trans (.bus(app_mtp_bus_sp2),.t_packet(physical_sp2_sp1));

//SP2 Receiver DUT
receiver DUT_SP2_Recv (.bus(mtp_app_bus_sp2), .r_packet(physical_sp1_sp2));

//Turn on VCD for DVE
initial
$vcdpluson;

initial
begin

#20 reset=0;  //reset to system

#10
reset=1;  //SP1: Appl to MTP layer signals reset to 0
app_mtp_bus_sp1.valid=0;
ap_mtp_bus_sp2.valid=0;

#10  //SP1: Appl to MTP layer signals
app_mtp_bus_sp1.valid=1;
ap_mtp_bus_sp1.li=6'b111111;
ap_mtp_bus_sp1.si=4'b1111;
ap_mtp_bus_sp1.ssf=4'b1111;
ap_mtp_bus_sp1.dpc=14'b1111_1111_1111_11;
ap_mtp_bus_sp1.opc=14'b1111_1111_1111_11;
ap_mtp_bus_sp1.sls=4'b1111;
ap_mtp_bus_sp1.channel_no=5'b11111;
ap_mtp_bus_sp1.trunk_no=7'b1111_111;
ap_mtp_bus_sp1.mtc=8'b1111_1111;
ap_mtp_bus_sp1.message='b1;

#10
app_mtp_bus_sp1.valid=0;
ap_mtp_bus_sp1.li=6'b0;
ap_mtp_bus_sp1.si=4'b0;
ap_mtp_bus_sp1.ssf=4'b0;
ap_mtp_bus_sp1.dpc=14'b0;
ap_mtp_bus_sp1.opc=14'b0;
ap_mtp_bus_sp1.sls=4'b0;
ap_mtp_bus_sp1.channel_no=5'b0;
ap_mtp_bus_sp1.trunk_no=7'b0;
ap_mtp_bus_sp1.mtc=8'b0;
app_mtp_bus_sp1.message='b0;

#70
app_mtp_bus_sp1.valid=1;
app_mtp_bus_sp1.li=6'b111110;
app_mtp_bus_sp1.si=4'b1111;
app_mtp_bus_sp1.ssf=4'b1111;
app_mtp_bus_sp1.dpc=14'b1111_1111_1111_11;
app_mtp_bus_sp1.opc=14'b1111_1111_1111_11;
app_mtp_bus_sp1.sls=4'b1111;
app_mtp_bus_sp1.channel_no=5'b11111;
app_mtp_bus_sp1.trunk_no=7'b1111_111;
app_mtp_bus_sp1.mtc=8'b1111_1111;
app_mtp_bus_sp1.message='b1;

#10
app_mtp_bus_sp1.valid=0;
app_mtp_bus_sp1.li=6'b0;
app_mtp_bus_sp1.si=4'b0;
app_mtp_bus_sp1.ssf=4'b0;
app_mtp_bus_sp1.dpc=14'b0;
app_mtp_bus_sp1.opc=14'b0;
app_mtp_bus_sp1.sls=4'b0;
app_mtp_bus_sp1.channel_no=5'b0;
app_mtp_bus_sp1.trunk_no=7'b0;
app_mtp_bus_sp1.mtc=8'b0;
app_mtp_bus_sp1.message='b0;

/******************************************************************************/

//SP2: Appl to MTP layer signals

#100
app_mtp_bus_sp2.valid=1;
app_mtp_bus_sp2.li=6'b111111;
app_mtp_bus_sp2.si=4'b1111;
app_mtp_bus_sp2.ssf=4'b1111;
app_mtp_bus_sp2.dpc=14'b1111_1111_1111_11;
app_mtp_bus_sp2.opc=14'b1111_1111_1111_11;
app_mtp_bus_sp2.sls=4'b1111;
appp_mtp_bus_sp2.channel_no=5'b11111;
appp_mtp_bus_sp2.trunk_no=7'b1111_111;
appp_mtp_bus_sp2.mtc=8'b1111_1111;
appp_mtp_bus_sp2.message='b1;

#10
app_mtp_bus_sp2.valid=0;
appp_mtp_bus_sp2.li=6'b0;
appp_mtp_bus_sp2.si=4'b0;
appp_mtp_bus_sp2.ssf=4'b0;
appp_mtp_bus_sp2.dpc=14'b0;
appp_mtp_bus_sp2.opc=14'b0;
appp_mtp_bus_sp2.sls=4'b0;
appp_mtp_bus_sp2.channel_no=5'b0;
appp_mtp_bus_sp2.trunk_no=7'b0;
appp_mtp_bus_sp2.mtc=8'b0;
appp_mtp_bus_sp2.message='b0;

#70
app_mtp_bus_sp2.valid=1;
appp_mtp_bus_sp2.li=6'b111110;
appp_mtp_bus_sp2.si=4'b1111;
appp_mtp_bus_sp2.ssf=4'b1111;
appp_mtp_bus_sp2.dpc=14'b1111_1111_1111_11;
appp_mtp_bus_sp2.opc=14'b1111_1111_1111_11;
appp_mtp_bus_sp2.sls=4'b1111;
appp_mtp_bus_sp2.channel_no=5'b11111;
appp_mtp_bus_sp2.trunk_no=7'b1111_111;
appp_mtp_bus_sp2.mtc=8'b1111_1111;
appp_mtp_bus_sp2.message='b1;

#10
app_mtp_bus_sp2.valid=0;
appp_mtp_bus_sp2.li=6'b0;
appp_mtp_bus_sp2.si=4'b0;
appp_mtp_bus_sp2.ssf=4'b0;
appp_mtp_bus_sp2.dpc=14'b0;
appp_mtp_bus_sp2.opc=14'b0;
appp_mtp_bus_sp2.sls=4'b0;
appp_mtp_bus_sp2.channel_no=5'b0;
appp_mtp_bus_sp2.trunk_no=7'b0;
appp_mtp_bus_sp2.mtc=8'b0;
app_mtp_bus_sp2.message="b0;

#100
$finish;
end

//Monitoring signals
initial
$monitor ($time, "clk=%b reset=%b sp1_li=%b sp2_li=%b successful=%b",
clk,reset,app_mtp_bus_sp1.li,mtp_app_bus_sp2.li,mtp_app_bus_sp2.valid);
endmodule
5.1 Simulation Result from VCS:

Chronologic VCS simulator copyright 1991-2009
Contains Synopsys proprietary information.
Compiler version D-2009.12; Runtime version D-2009.12; Jul 28 15:30 2010
VCD+ Writer D-2009.12 Copyright 2009 Synopsys Inc.
Warning! From $vcdplupon at time 0 in file testbench.sv line 74:
[VCD++-ETINU]:
Enumerated type name "unnamed$$" is not unique. Suffix added to make it unique.

0clk=1 reset=0 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
5clk=0 reset=0 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
10clk=1 reset=0 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
15clk=0 reset=0 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
20clk=1 reset=0 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
25clk=0 reset=0 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
30clk=1 reset=1 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
35clk=0 reset=1 sp1_li=xxxxxx sp2_li=xxxxxx successful=x
40clk=1 reset=1 sp1_li=111111 sp2_li=xxxxxx successful=x
45clk=0 reset=1 sp1_li=111111 sp2_li=xxxxxx successful=x
50clk=1 reset=1 sp1_li=000000 sp2_li=xxxxxx successful=x
55clk=0 reset=1 sp1_li=000000 sp2_li=xxxxxx successful=x
60clk=1 reset=1 sp1_li=000000 sp2_li=xxxxxx successful=x
65clk=0 reset=1 sp1_li=000000 sp2_li=xxxxxx successful=x
70clk=1 reset=1 sp1_li=000000 sp2_li=xxxxxx successful=x
75clk=0 reset=1 sp1_li=000000 sp2_li=xxxxxx successful=x
80clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=x
85clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=x
90clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
95clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
100clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
105clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
110clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
115clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
120clk=1 reset=1 sp1_li=111110 sp2_li=111111 successful=1
125clk=0 reset=1 sp1_li=111110 sp2_li=111111 successful=1
130clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
135clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
140clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
145clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
150clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
155clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
160clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
165clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
170clk=1 reset=1 sp1_li=000000 sp2_li=111111 successful=1
175clk=0 reset=1 sp1_li=000000 sp2_li=111111 successful=1
180clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
185clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
190clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
195clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
200clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
205clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
210clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
215clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
220clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
225clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
230clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
235clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
240clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
245clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
250clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
255clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
260clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
265clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
270clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
275clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
280clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
285clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
290clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
295clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
300clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
305clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
310clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
315clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
320clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
325clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
330clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
335clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
340clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
345clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
350clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
355clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
360clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
365clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
370clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
375clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
380clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
385clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
390clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
395clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
400clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
405clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
410clk=1 reset=1 sp1_li=000000 sp2_li=111110 successful=1
415clk=0 reset=1 sp1_li=000000 sp2_li=111110 successful=1
$finish called from file "testbench.sv", line 200.
$finish at simulation time 420

V C S   S i m u l a t i o n   R e p o r t

Time: 420
CPU Time: 0.280 seconds; Data structure size: 0.0Mb
Wed Jul 28 15:31:02 2010

5.2 Simulation Results from DVE:
<table>
<thead>
<tr>
<th>#</th>
<th>Design</th>
<th>Signal</th>
<th>Value</th>
<th>Time: 0 - 437 \times 1s (C10REF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>014</td>
<td>V1</td>
<td>temp[15:0]</td>
<td>16'hxxxx</td>
<td></td>
</tr>
<tr>
<td>015</td>
<td>V1</td>
<td>packet_bitstufing760'h...xxx xxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>016</td>
<td>V1</td>
<td>count[2:0]</td>
<td>3'h1</td>
<td>2838</td>
</tr>
<tr>
<td>017</td>
<td>V1</td>
<td>fsn_expected[6:0]</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>018</td>
<td>V1</td>
<td>bib_prev</td>
<td>1'b0</td>
<td>01</td>
</tr>
<tr>
<td>019</td>
<td>V1</td>
<td>bib_next</td>
<td>1'b0</td>
<td>02</td>
</tr>
<tr>
<td>020</td>
<td>V1</td>
<td>fib_prev</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>021</td>
<td>V1</td>
<td>fib_next</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>022</td>
<td>V1</td>
<td>c/c_poly[15:0]</td>
<td>4129</td>
<td></td>
</tr>
<tr>
<td>023</td>
<td>V1</td>
<td>current_packet</td>
<td>624'h...xxx xxx</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SG**

**DUT_SPI_Trans**

<table>
<thead>
<tr>
<th>#</th>
<th>Design</th>
<th>Signal</th>
<th>Value</th>
<th>Time: 0 - 437 \times 1s (C10REF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>024</td>
<td>V1</td>
<td>cs[31:0]</td>
<td>reset_state</td>
<td>![state diagram]</td>
</tr>
<tr>
<td>025</td>
<td>V1</td>
<td>ns[31:0]</td>
<td>reset_state</td>
<td>![state diagram]</td>
</tr>
<tr>
<td>#</td>
<td>Design</td>
<td>Signal</td>
<td>Value 16'hXXXXX</td>
<td>Value 16'hXXXXX</td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>027</td>
<td>V1</td>
<td>[31:0]</td>
<td>x</td>
<td>036</td>
</tr>
<tr>
<td>028</td>
<td>V1</td>
<td>crc_length[31:0]</td>
<td>x</td>
<td>600</td>
</tr>
<tr>
<td>029</td>
<td>V1</td>
<td>crc_result[15:0]</td>
<td>c7d7</td>
<td></td>
</tr>
<tr>
<td>030</td>
<td>V1</td>
<td>msb</td>
<td>1'b1x</td>
<td></td>
</tr>
<tr>
<td>031</td>
<td>V1</td>
<td>current_byte[7:0]</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>032</td>
<td>V1</td>
<td>temp[15:0]</td>
<td>283e</td>
<td></td>
</tr>
<tr>
<td>033</td>
<td>V1</td>
<td>packet_bitstuffing[760'h...XXXXx]</td>
<td>etbe 500 007e</td>
<td>000 0000 0000 0000 1e8 6be f3ef bef4 etbe 3e01 007e</td>
</tr>
<tr>
<td>034</td>
<td>V1</td>
<td>count[2:0]</td>
<td>3'h1x</td>
<td></td>
</tr>
<tr>
<td>035</td>
<td>V1</td>
<td>tsr_cnt[6:0]</td>
<td>7'h00</td>
<td>00</td>
</tr>
<tr>
<td>036</td>
<td>V1</td>
<td>bib_prev</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>037</td>
<td>V1</td>
<td>bib_next</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>038</td>
<td>V1</td>
<td>fib_next</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>039</td>
<td>V1</td>
<td>fib_next</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>040</td>
<td>V1</td>
<td>fifo_read[3:0]</td>
<td>4'h0</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Desig.</td>
<td>Signal</td>
<td>Value</td>
<td>Time: 0 - 437 × 1 s (C1:0REF)</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>-------------------------</td>
<td>-------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>041</td>
<td>V1</td>
<td>ffo_write[3:0]</td>
<td>4'h10</td>
<td>00100010000000000000000100010001</td>
</tr>
<tr>
<td>042</td>
<td>V1</td>
<td>ffo_full</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>043</td>
<td>V1</td>
<td>ffo_empty</td>
<td>1'b1</td>
<td></td>
</tr>
<tr>
<td>044</td>
<td>V1</td>
<td>packet_formation</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>045</td>
<td>V1</td>
<td>retransmit_flag</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>046</td>
<td>V1</td>
<td>start_flag[7:0]</td>
<td>126</td>
<td>126</td>
</tr>
<tr>
<td>047</td>
<td>V1</td>
<td>crc_poly[15:0]</td>
<td>14129</td>
<td>14129</td>
</tr>
<tr>
<td>048</td>
<td>V1</td>
<td>current_packet</td>
<td>62'h0</td>
<td></td>
</tr>
<tr>
<td>049</td>
<td>V1</td>
<td>retransmission[0]</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>SG</td>
<td></td>
<td>DUT_SP2_Receiv</td>
<td></td>
<td></td>
</tr>
<tr>
<td>050</td>
<td>V1</td>
<td>bsn[5:0]</td>
<td>7'hxx</td>
<td>00</td>
</tr>
<tr>
<td>051</td>
<td>V1</td>
<td>tsn[5:0]</td>
<td>7'hxx</td>
<td>00</td>
</tr>
<tr>
<td>052</td>
<td>V1</td>
<td>bib</td>
<td>1'bX</td>
<td></td>
</tr>
<tr>
<td>053</td>
<td>V1</td>
<td>fib</td>
<td>1'bX</td>
<td></td>
</tr>
</tbody>
</table>

Note: The table and diagram represent signal transitions and values over time.

<table>
<thead>
<tr>
<th>#</th>
<th>Desig.</th>
<th>Signal</th>
<th>Value</th>
<th>Time: 0 - 437 × 1s (C1:0REF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>054</td>
<td>V1</td>
<td>fc5[15:0]</td>
<td>16'hxxxx</td>
<td></td>
</tr>
<tr>
<td>055</td>
<td>V1</td>
<td>cs[31:0]</td>
<td>idle_state</td>
<td></td>
</tr>
<tr>
<td>056</td>
<td>V1</td>
<td>ns[31:0]</td>
<td>idle_state</td>
<td></td>
</tr>
<tr>
<td>057</td>
<td>V1</td>
<td>jf[31:0]</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>058</td>
<td>V1</td>
<td>jf[31:0]</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>059</td>
<td>V1</td>
<td>crc_length[31:0]</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>060</td>
<td>V1</td>
<td>crc_result[15:0]</td>
<td>16'hxxxx</td>
<td></td>
</tr>
<tr>
<td>061</td>
<td>V1</td>
<td>msb</td>
<td>1'hbx</td>
<td></td>
</tr>
<tr>
<td>062</td>
<td>V1</td>
<td>current_byte[7:0]</td>
<td>16'hxx</td>
<td></td>
</tr>
<tr>
<td>063</td>
<td>V1</td>
<td>temp[15:0]</td>
<td>16'hxxxx</td>
<td></td>
</tr>
<tr>
<td>064</td>
<td>V1</td>
<td>packet_bit stuffing</td>
<td>760'hxxxxxxx</td>
<td>2836</td>
</tr>
<tr>
<td>065</td>
<td>V1</td>
<td>count[2:0]</td>
<td>3'hx</td>
<td></td>
</tr>
<tr>
<td>066</td>
<td>V1</td>
<td>bsn_cnt[6:0]</td>
<td>7'h00</td>
<td></td>
</tr>
<tr>
<td>067</td>
<td>V1</td>
<td>bsn_expected[6:0]</td>
<td>7'h00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Desig.</th>
<th>Signal</th>
<th>Value</th>
<th>Time: 0 - 437 × 1s (C1:0REF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>066</td>
<td>V1</td>
<td>tsn_cnt[6:0]</td>
<td>7'h00</td>
<td></td>
</tr>
<tr>
<td>068</td>
<td>V1</td>
<td>tsn_expected[6:0]</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>070</td>
<td>V1</td>
<td>bib_prev</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>071</td>
<td>V1</td>
<td>bib_next</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>072</td>
<td>V1</td>
<td>fib_prev</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>073</td>
<td>V1</td>
<td>fib_next</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>074</td>
<td>V1</td>
<td>crc_poly[15:0]</td>
<td>4129</td>
<td></td>
</tr>
<tr>
<td>075</td>
<td>V1</td>
<td>current_packet</td>
<td>62'h...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DUT_SPZ_Tx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>076</td>
<td>V1</td>
<td>bus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>077</td>
<td>V1</td>
<td>t_packet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>078</td>
<td>V1</td>
<td>cs[31:0]</td>
<td>reset_state</td>
<td></td>
</tr>
<tr>
<td>079</td>
<td>V1</td>
<td>ns[31:0]</td>
<td>reset_state</td>
<td></td>
</tr>
<tr>
<td>080</td>
<td>V1</td>
<td>j[31:0]</td>
<td>x</td>
<td>625</td>
</tr>
</tbody>
</table>

---

19
<table>
<thead>
<tr>
<th>#</th>
<th>Design</th>
<th>Signal</th>
<th>Value</th>
<th>Time: 0 - 437 x 1s (C1:CREF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>095</td>
<td>V1</td>
<td>fifo_prev</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>096</td>
<td>V1</td>
<td>fifo_next</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>097</td>
<td>V1</td>
<td>fifo_read[3:0]</td>
<td>4'h0</td>
<td></td>
</tr>
<tr>
<td>098</td>
<td>V1</td>
<td>fifo_write[3:0]</td>
<td>4'h0</td>
<td>0</td>
</tr>
<tr>
<td>099</td>
<td>V1</td>
<td>fifo_full</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>V1</td>
<td>fifo_empty</td>
<td>1'b1</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>V1</td>
<td>packet Formation</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>V1</td>
<td>error</td>
<td>1'bx</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>V1</td>
<td>retransmit_flag</td>
<td>1'b0</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>V1</td>
<td>start_flag[7:0]</td>
<td>125</td>
<td>128</td>
</tr>
<tr>
<td>105</td>
<td>V1</td>
<td>crc_poly[15:0]</td>
<td>4129</td>
<td>4129</td>
</tr>
<tr>
<td>106</td>
<td>V1</td>
<td>current_packet</td>
<td>824'h...xxxxxxx</td>
<td>007e 007e 007e 3e01 007e</td>
</tr>
<tr>
<td>107</td>
<td>V1</td>
<td>retransmission</td>
<td>0: Not Loaded</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Design.</td>
<td>Signal</td>
<td>Value</td>
<td>Time</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>--------------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>SG</td>
<td></td>
<td>app_mtp_bus_sp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>V1</td>
<td>clk</td>
<td>St1</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>V1</td>
<td>reset</td>
<td>St0</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>V1</td>
<td>valid</td>
<td>t'b'x</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>V1</td>
<td>li[5:0]</td>
<td>6'hox</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>V1</td>
<td>si[3:0]</td>
<td>4'hx</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>V1</td>
<td>ss[3:0]</td>
<td>4'hx</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>V1</td>
<td>dp[13:0]</td>
<td>14'hxxx</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>V1</td>
<td>op[13:0]</td>
<td>14'hxxx</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>V1</td>
<td>sl[3:0]</td>
<td>4'hx</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>V1</td>
<td>channel_no[4:0]</td>
<td>5'hxx</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>V1</td>
<td>trunk_no[5:0]</td>
<td>7'hxx</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>V1</td>
<td>mtc[7:0]</td>
<td>6'hxx</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>V1</td>
<td>message[0:53][7:512'hxxxxxxx]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time: 0 - 437 × 1 s (C1:0REF)
<table>
<thead>
<tr>
<th>#</th>
<th>Design</th>
<th>Signal</th>
<th>Value</th>
<th>Time</th>
<th>Time: 0 - 437 x 1 s (C1:0REF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>app_mbp_bus_sp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>V1</td>
<td>clk</td>
<td>St1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>V1</td>
<td>reset</td>
<td>St0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>V1</td>
<td>valid</td>
<td>1'bX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>V1</td>
<td>li[5:0]</td>
<td>6'hXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>V1</td>
<td>si[3:0]</td>
<td>4'hX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>V1</td>
<td>ss[3:0]</td>
<td>4'hX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>V1</td>
<td>dpc[13:0]</td>
<td>14'hXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>V1</td>
<td>opc[13:0]</td>
<td>14'hXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>V1</td>
<td>sls[3:0]</td>
<td>4'hX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>V1</td>
<td>channel_no[4:0]</td>
<td>6'hXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>V1</td>
<td>trunk_no[5:0]</td>
<td>7'hXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>V1</td>
<td>mtc[7:0]</td>
<td>6'hXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>V1</td>
<td>message[0:53][7:512'h...000000]</td>
<td>000000000000000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Design.</td>
<td>Signal</td>
<td>Value</td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>--------------</td>
<td>-------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>V1</td>
<td>clk</td>
<td>St1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>V1</td>
<td>reset</td>
<td>St0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>V1</td>
<td>valid</td>
<td>1'hx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>V1</td>
<td>r1[5:0]</td>
<td>6'h0x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>V1</td>
<td>s1[3:0]</td>
<td>4'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>139</td>
<td>V1</td>
<td>s0[3:0]</td>
<td>4'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>V1</td>
<td>dpc[13:0]</td>
<td>14'hxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>V1</td>
<td>opc[13:0]</td>
<td>14'hxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>V1</td>
<td>s1s[3:0]</td>
<td>4'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>143</td>
<td>V1</td>
<td>channel_no[4:0]</td>
<td>6'h0x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>V1</td>
<td>trunk_no[6:0]</td>
<td>7'h0x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>V1</td>
<td>mtc[7:0]</td>
<td>6'h0x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>146</td>
<td>V1</td>
<td>message[0:53][7:512'h...xxxxxx]</td>
<td>0 0000 0000 0000 0000 0000 0000 0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User: bhatas@titan.ecs.csus.edu  Date: 07/28/2010  Total Time Range: 0 - 437  Page 11 of 12
<table>
<thead>
<tr>
<th>#</th>
<th>Desig.</th>
<th>Signal</th>
<th>Value</th>
<th>Time</th>
<th>Total Time Range: 0 - 437</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG</td>
<td>mtp_app_bus_sp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>V1</td>
<td>clk</td>
<td>St1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>V1</td>
<td>reset</td>
<td>St0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>149</td>
<td>V1</td>
<td>valid</td>
<td>1'bxs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>V1</td>
<td>l[i][0:0]</td>
<td>6'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>V1</td>
<td>s[i][0:0]</td>
<td>4'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>152</td>
<td>V1</td>
<td>ss[i][0:0]</td>
<td>4'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>153</td>
<td>V1</td>
<td>dpc[i][13:0]</td>
<td>14'hxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>154</td>
<td>V1</td>
<td>opc[i][13:0]</td>
<td>14'hxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>V1</td>
<td>sls[i][0:0]</td>
<td>4'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>V1</td>
<td>channel_no[i][4:0]</td>
<td>5'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>157</td>
<td>V1</td>
<td>trunk_no[i][5:0]</td>
<td>7'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>V1</td>
<td>mtc[i][7:0]</td>
<td>8'hxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>V1</td>
<td>message[0:53][7:5]</td>
<td>12'hxxx</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User: bharis@titan.ecs.csus.edu  Date: 07/20/2010  Total Time Range: 0 - 437  Page 12 of 12
<table>
<thead>
<tr>
<th>#</th>
<th>Design.</th>
<th>Signal</th>
<th>Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SG</td>
<td>physical_sp1_sp2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>V1</td>
<td>packet[94:0][7:0]</td>
<td>760'h_...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SG</td>
<td>physical_sp2_sp1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>V1</td>
<td>packet[94:0][7:0]</td>
<td>760'h_...</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 6

CONCLUSION

CCS#7 has revolutionized the communication world with a better and faster means of telecommunication. This protocol has increased the overall throughput of the system as compared to those with the CAS protocol.[1]

A good understanding of concept of CCS#7 protocol has been done. The simulation of Verilog code verified the protocol. Each module of the code was tested successfully through the Verilog simulations.

We have implemented the encoder and the decoder modules in order to test the functionality of the protocol. Other scope of future work is to increase the speed, which restricts the bandwidth in the two-way communication. We have proposed a solution for the same in our study, which we found as the challenging task.
1. Common Channel Signaling System # 7; Alcatel; Document number: 770 00438 0590 VHBE Ed. 07

2. ITU-T Q.118: General Recommendations on Telephone Switching and Signaling


5. ITU-T Q.702: CCS #7: Signaling Data Link.

6. ITU-T Q.703: CCS #7: Signaling Link.


