Global Journal of Computer Science and Technology: A
Hardware & Computation
Volume 15 Issue 2 Version 1.0 Year 2015
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Design and Implementation of Data Scrambler & Descrambler System using VHDL

By Naina K. Randive & Prof. G. P. Borkhade
Sant Gadge Baba Amaravati University, India

Abstract- Multimedia data security is very important for multimedia commerce on the internet and real time data multicast. An striking solution for encrypting data with adequate message security at low cost is the use of Scrambler/Descrambler. Scramblers are necessary components of physical layer system standards besides interleaved coding and modulation. Scramblers are well used in modern VLSI design especially those are used in data communication system either to secure data or re-code periodic sequence of binary bits stream. However, it is necessary to have a descrambler block on the receiving side while using scrambling data in the transmitting end to have the actual input sequence on the receiving end. Scrambling and De-scrambling is an algorithm that converts an input string into a seemingly random string of the same length to avoid simultaneous bits in the long format of data. Scramblers have accomplish of uses in today's data communication protocols. On the other hand, those methods that are theoretical proposed are not feasible in the modern digital design due to many reasons such as slower data rate, increasing information, circuit hazards, uncountable hold-up etc. Therefore it is requisite for the modern digital design to have modified architecture to meet the required goal. We will recommend here modified scrambler design which is perfectly suitable for any industrial design.

Keywords: scrambler, descrambler, VHDL, and FPGA.

GJCST-A Classification : B.5.2 B.6.3

DESIGNANDIMPLEMENTATIONOFDATASCRAMBLERDESCRAMBLERSYSTEMUSINGVHDL

Strictly as per the compliance and regulations of:

Design and Implementation of Data Scrambler & Descrambler System using VHDL

Naina K. Randive [α] & Prof. G. P. Borkhade [σ]

Abstract- Multimedia data security is very important for multimedia commerce on the internet and real time data multicast. An striking solution for encrypting data with adequate message security at low cost is the use of Scrambler/Descrambler. Scramblers are necessary components of physical layer system standards besides interleaved coding and modulation. Scramblers are well used in modern VLSI design especially those are used in data communication system either to secure data or re-code periodic sequence of binary bits stream. However, it is necessary to have a descrambler block on the receiving side while using scrambling data in the transmitting end to have the actual input sequence on the receiving end. Scrambling and De-scrambling is an algorithm that converts an input string into a seemingly random string of the same length to avoid simultaneous bits in the long format of data. Scramblers have accomplish of uses in today's data communication protocols. On the other hand, those methods that are theoretical proposed are not feasible in the modern digital design due to many reasons such as slower data rate, increasing information, circuit hazards, uncountable hold-up etc. Therefore it is requisite for the modern digital design to have modified architecture to meet the required goal. We will recommend here modified scrambler design which is perfectly suitable for any industrial design.

Keywords: scrambler, descrambler, VHDL, and FPGA.

I. Introduction

In telecommunications, a scrambler is a device that transposes or inverts signals or otherwise encodes a message at the transmitter to make the message unintelligible at a receiver not equipped with an appropriately set descrambling device. while encryption usually refers to operations carried out in the digital domain, scrambling typically refers to operations carried out in the analog domain. Scrambling is consummate by the addition of components to the original signal or the changing of some important component of the original signal in order to make extraction of the original signal complex .To improve the degree of data security in a conventional Scrambler the number of stages of the shift register needs to be enhanced. This conversely increases error propagation. A uncomplicated method for ensuring security is to encrypt the data. The pseudo-noise (PN) key generation is of paramount importance for any secure communication system. PN sequences base on Linear Feedback Shift Registers (LFSR) and non linear combination based implementations are simplest to give moderate level of security. Chaos base encryption techniques have proved fruitful, but complexity of such systems is important. The complex system generated is used to scramble incoming plain text. At the receiving end, the same code be generated and successfully used to decrypt the transmitted data. The ease of the circuit along with the complexity of the generated codes makes the circuit striking for secure message communication applications.

Author α σ: Dept. of Electronics and Telecommunications Dept. of Electronics and Telecommunications P.R. Pote (Patil) college of Engineering and, Management, Amravati, India.
e-mails: naina0689@gmail.com, gauri.borkhade@gmail.com

II. Proposed Work

The entire operation is proposed using Modelsim and Xilinx blocks goes through three phases.

1. Architecture of Scrambler & Descrambler
2. Block diagram od Scrambler & Descrambler
3. Overview of Scrambler & Descrambler

a) Architecture of Scrambler & Descrambler

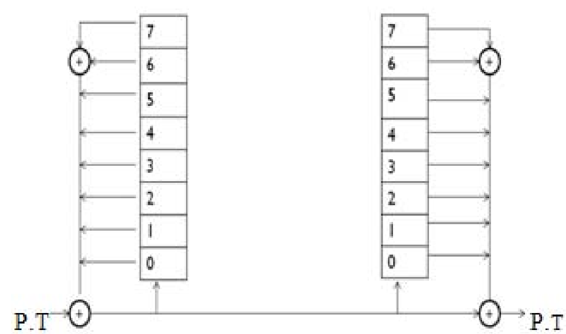

Figure 1 : Architecture of Scrambler and Descrambler

A data scramble & descramble are shown in fig. The scramble operates in the following manner. The initial shift register contents are random but prespecified and fixed to the same in both the scramble and descramble. The initial bit sequence of location 6 & 7 in the shift register X-OR is placed in shift register stage 0.The generated bit sequence is the sum with plain text, then it becomes the bit sequence is crypto word. As this bit is presented to the channel the contents of the shift register are shifted up one stage as follows: 7→out,5→6,4→5,3→4,2→3,1→2.

The descramble operates as follows. The initial shift register contents are random but prespecified and fixed to the same in both the scramble and descramble.

The initial bit sequence of location 6 & 7 in the shift register X-OR is placed in shift register stage 0.The generated bit sequence is the sum with crypto word then it becomes the bit sequence is plain text. As this bit is presented to the channel the contents of the shift register are shifted up one stage as follows: 7→out,5→6,4→5,3→4,2→3,1→2.

b) Block diagram Scrambler & Descrambler

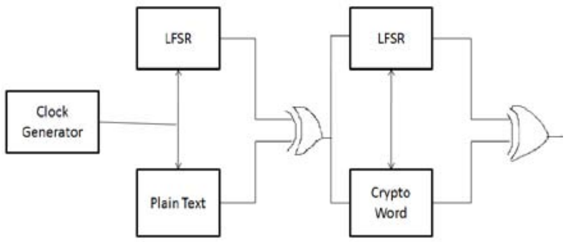

Figure 2 : Block diagram of scrambler And descrambler

Block diagram of scramble & descramble represented in Figure. Scrambler is performed in sequence X-OR the 8-bit plain text (D0-D7) character with the 8-bit (D0-D7) output of the LFSR. An output of the LFSR is XOR with plain text of the data to be processed. The LFSR and data register are then successively advanced and the output processing is repeated for D1 through D7.

Descrambler is performed in order XOR the 8-bit crypt word (D0-D7) character with the 8-bit (D0-D7) output of the LFSR. An output of the LFSR is XOR with crypto word of the data to be processed. The LFSR and data register are then consecutively advanced and the output processing is repeated for D1 through D7.

c) Overview of Scrambler and Descrambler

In the transmitter, a pseudorandom cipher sequence is added (modulo 2) to the data (or control) sequence to produce a scrambled data (or control) sequence.

In the receiver, the same pseudorandom cipher sequence is subtracted (modulo 2) from the scrambled data (or control) sequence to recover the transmitted data (or control) sequence, as illustrated in figure.

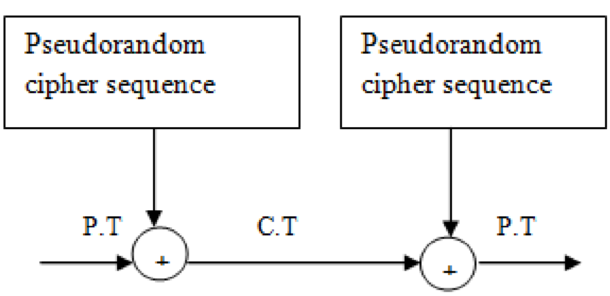

Figure 3 : Overview of scrambler and descrambler

III. Results

The proposed Fpga implementation of various outputs is done using Modelsim and Xilinx. Both hardware and software implementation of various output is tabulated below.

a) For 8 bit Scrambler

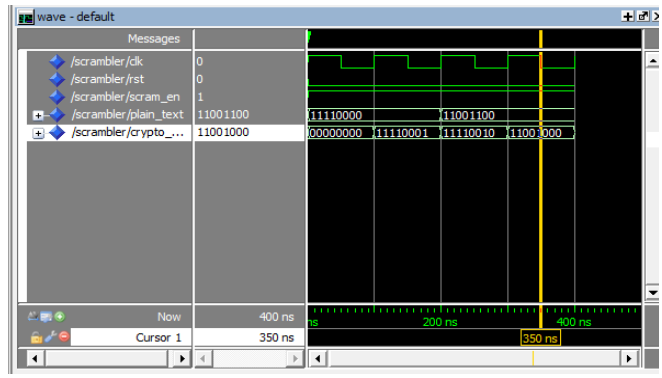

Figure 4 : Wave output for Scrambler

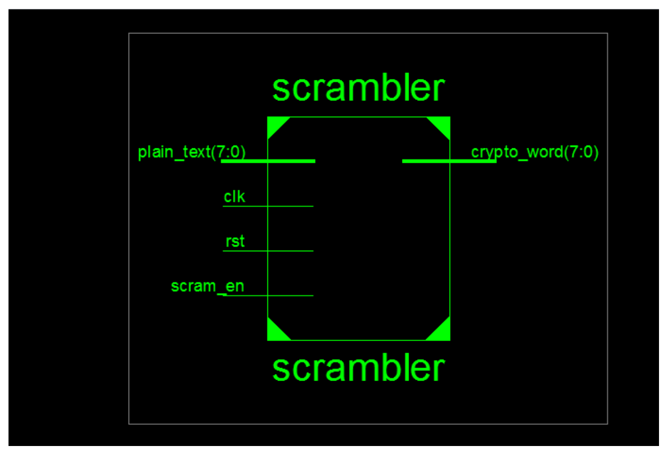

Figure 5 : RTL Schematic for Scrambler

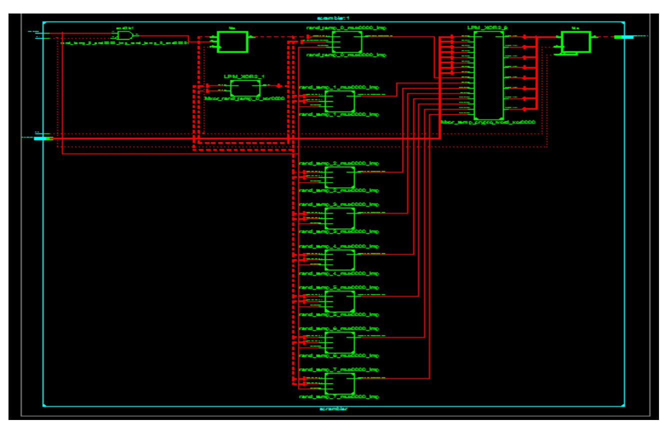

Figure 6 : Internal View of RTL Schematic for Scrambler

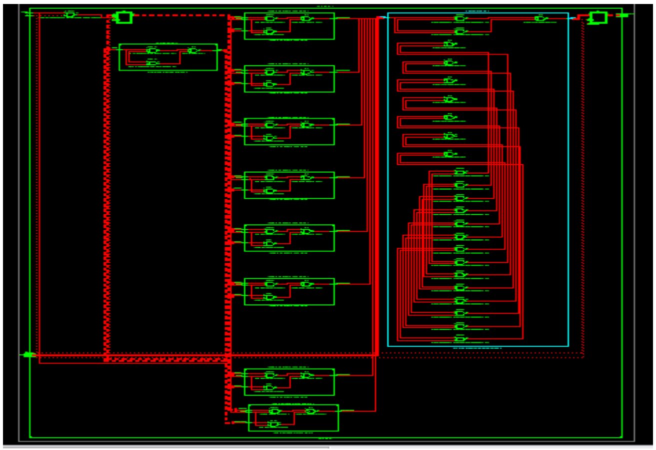

Figure 7 : Internal View of RTL Schematic for Scrambler

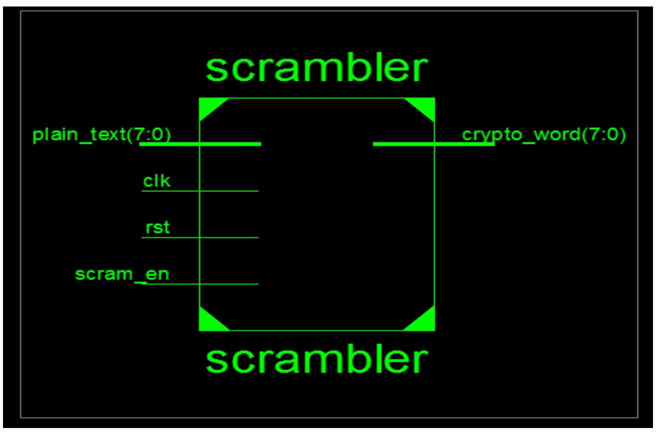

Figure 8 : View Technology Schematic for Scrambler

Area for Scrmabler

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                        9  out of   4656     0%
 Number of Slice Flip Flops:             16  out of   9312     0%
 Number of 4 input LUTs:                 10  out of   9312     0%
 Number of IOs:                          19
 Number of bonded IOBs:                  19  out of    232     8%
 Number of GCLKs:                         1  out of     24     4%

---------------------------
```

Figure 9 : Area for Scrambler

Timing summary for Scrambler

Minimum period= 3.424ns(Maximum Frequency 292.056MHz)

Throughput for 8 bit scrambler:- Maximum Freq*No of Bit/No of cycle

=292.056MHz*8/1

=2336.448MHz

~2.4GHz

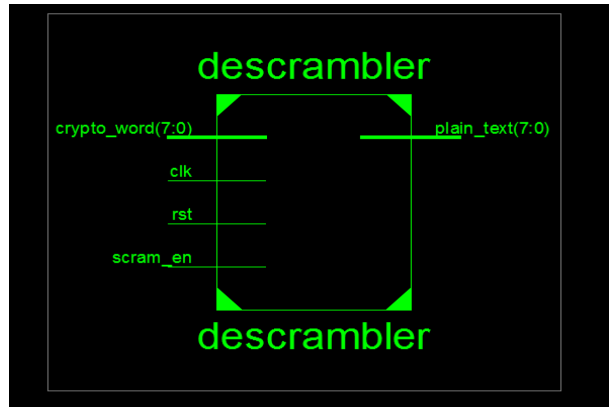

Figure 10 : RTL Schematic for Descrambler

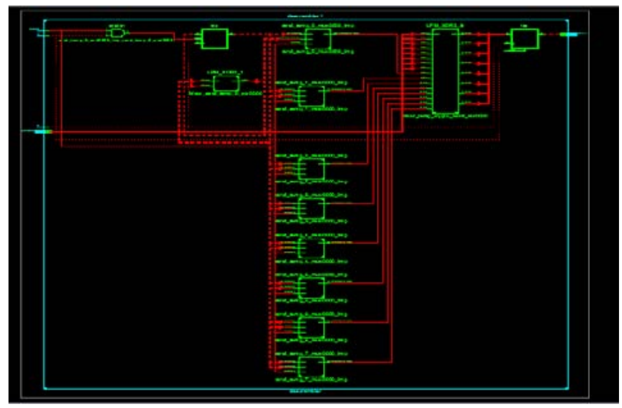

Figure 11 : Internal View of RTL Schematic for Descrambler

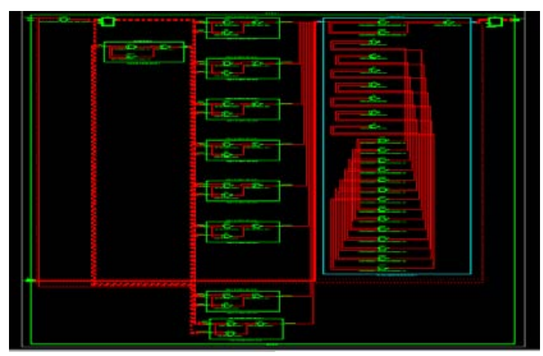

Figure 12 : Internal View of RTL Schematic for Descrambler

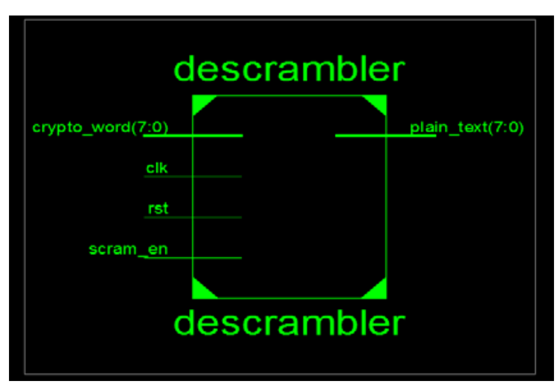

Figure 13 : View Technology Schematic for Descrambler

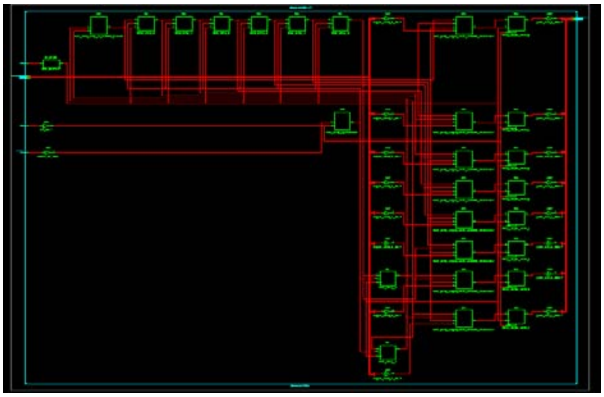

Figure 14 : Internal View Of View technology Schmatic for Descrambler

Area for Scrmabler

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                       9  out of   4656     0%
 Number of Slice Flip Flops:            16  out of   9312     0%
 Number of 4 input LUTs:                10  out of   9312     0%
 Number of IOs:                         19
 Number of bonded IOBs:                 19  out of    232     8%
 Number of GCLKs:                        1  out of     24     4%

---------------------------
```

Figure 15 : Area for Scrambler

Timing summary for Scrambler

Minimum period= 3.424ns(Maximum Frequency 292.056MHz)

Throughput for 8 bit scrambler:- Maximum Freq*No of Bit/No of cycle

=292.056MHz*8/1

=2336.448MHz

~2.4GHz

b) *Maximum Length polynomial for Scrambler*

For Enhanced Security using polynomial equation

X7+X6+X4+X3+1=0

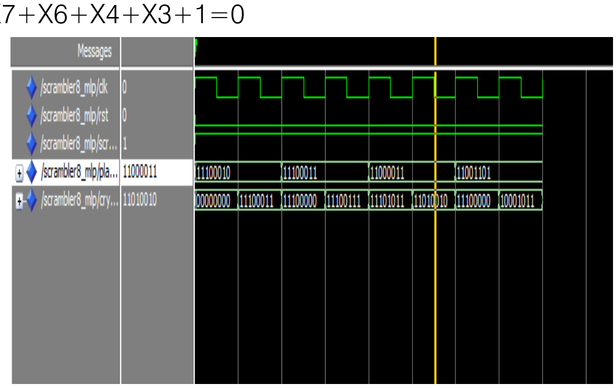

Figure 16 : Wave output for Maximum Length polynomial for Scrambler

c) *Maximum Length polynomial for Desrambler*

For Enhanced Security using polynomial equation

X7+X6+X4+X3+1=0

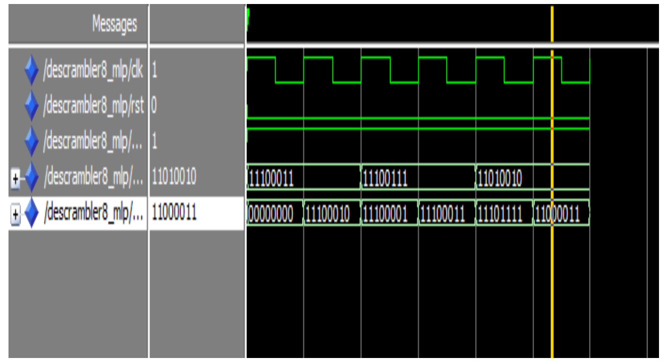

Figure 17 : Wave output for Maximum Length Polynomial for Descrambler

d) *For 16 Bit Scrambler*

Area for 16bit Scrambler

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                      37  out of   4656     0%
 Number of Slice Flip Flops:            64  out of   9312     0%
 Number of 4 input LUTs:                34  out of   9312     0%
 Number of IOs:                         67
 Number of bonded IOBs:                 67  out of    232    28%
 Number of GCLKs:                        1  out of     24     4%
```

Figure 18 : Area for 16Bit Scrambler

Timing Summary:

Minimum period=3.424ns (Maximum Frequency 292. 056MHz)

e) *For 32 Bit Scrambler*

Area for 32 Bit Scrambler

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                      37  out of   4656     0%
 Number of Slice Flip Flops:            64  out of   9312     0%
 Number of 4 input LUTs:                34  out of   9312     0%
 Number of IOs:                         67
 Number of bonded IOBs:                 67  out of    232    28%
 Number of GCLKs:                        1  out of     24     4%
```

Figure 19 : Area for 32Bit Scrambler

Timing Summary:

Minimum period=3.424ns(Maximum Frequency 292.056 MHz)

f) For 16 Bit Scrambler

Area for 16bit Scrambler

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                      37  out of   4656     0%
 Number of Slice Flip Flops:            64  out of   9312     0%
 Number of 4 input LUTs:                34  out of   9312     0%
 Number of IOs:                         67
 Number of bonded IOBs:                 67  out of    232    28%
 Number of GCLKs:                        1  out of     24     4%
```

Figure 20 : Area for 16Bit Scrambler

Timing Summary:

Minimum period=3.424ns(Maximum Frequency 292.056MHz)

g) For 32 Bit Scrambler

Area for 32 Bit Scrambler

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                      37  out of   4656     0%
 Number of Slice Flip Flops:            64  out of   9312     0%
 Number of 4 input LUTs:                34  out of   9312     0%
 Number of IOs:                         67
 Number of bonded IOBs:                 67  out of    232    28%
 Number of GCLKs:                        1  out of     24     4%
```

Figure 21 : Area for 32Bit Scrambler

Timing summary:

Number Of Bit	Maxmum Frequency
8 Bit	292.056MHz
16 Bit	292.056MHz
32 Bit	292.056 MHz
8 Bit	449.438MHz

Timing Summary:

Minimum period=3.424ns(Maximum Frequency 292.056 MHz)

h) Implementation

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                      37  out of   4656     0%
 Number of Slice Flip Flops:            64  out of   9312     0%
 Number of 4 input LUTs:                34  out of   9312     0%
 Number of IOs:                         67
 Number of bonded IOBs:                 67  out of    232    28%
 Number of GCLKs:                        1  out of     24     4%
```

Figure 22 : Device utilization Summary

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 2.225ns (Maximum Frequency: 449.438MHz)
   Minimum input arrival time before clock: 4.366ns
   Maximum output required time after clock: 4.283ns
   Maximum combinational path delay: No path found
```

Figure 23 : Timing Summary

IV. Conclusion

A new modified scheme for complex PN-code based data scrambler and descrambler has been presented. A scrambler & descrambler accepts information in intelligible form and through intellectual transformation assure data quality with fastest rate without any error or dropping occurrence. We used our proposed and modified design in our present universal serial bus architecture. Moreover, this current design is very efficient, more securable, high speed, low power and lower area used & it has lots of scope to improved.

V. Acknowledgment

I have taken efforts in this project. Though it would not have been possible without the kind support and help of many individuals and organizations. I would like to make bigger my sincere thanks to all of them. I am highly indebted to Prof .G. P. Borkhade for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to convey my pleasure towards my guide & member of Electronics and Telecommunication Engineering for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to institute persons for giving me such attention and time. My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

References Références Referencias

1. Rajib Imranand Monirul Islam, 2013, Indurtial Modified Digital scrambler and descrambler system.
2. Davinder Pal Sharma, 2013, Data scrambler of ultra-wide band communication system.
3. G.M. Bhat, M. Mustafa, Shabir Ahmad, and Javaid Ahmad, 2009, VHDL modeling and simulation of data scrambler and descrambler for secure data communication.

4. Sharma, D. P. Singh J, Simulation and spectral analysis of the scrambler for 56Kbps modem. The Journal of Signal Processing Systems. 67, 269-277 (2012).
5. Sharma, D. P. Singh J.DSP based implementation of scrambler for 56Kbps modem. Signal Processing – An International Journal. 4, 85-96 (2010).
6. Hethan Kumar, M Praveen Kumar Y G, Dr. M volume 3 Issue 4,April 2014, "Design and implementation of Logical Scrambler Architecture for OTN Protocol".
7. Xiao-Bei Liu, Soo Ngee Koh, Chee-Cheon Chui, and Xin-Wen Wu, "A Study of Reconstruction of Linear Scrambler using Dual Words of Channel Encoder" March 2013.