Design and FPGA Implementation of 100Gbit/s Scrambler Architectures for OTN Protocol
Chethan Kumar M¹, Praveen Kumar Y G², Dr. M. Z. Kurian³.
¹PG Student, Sri Siddhartha Institute of Technology, Tumkur.
²Asst. Prof., dept. of E&C, Sri Siddhartha Institute of Technology, Tumkur.
³HOD, dept. of ECE, Sri Siddhartha Institute of Technology, Tumkur.

ABSTRACT:
This paper describes design and FPGA (Field programmable gate array) implementation of two scrambler architectures, with applications in 100Gbit/s optical transport network (OTN) systems. And comparing these two scrambler architectures using some parameters like resource utilization and performance. The scrambler architectures are modeled using Verilog HDL to verify the correct operation. The programmable logic synthesis and simulation were performed using the tools provided by Xilinx Inc. (ISE 13.4 and Modelsim6.3c) with a Virtex-6 as target device.

Keywords: OTN, FPGA, Scrambling.

[1] INTRODUCTION

The optical transfer networks (OTN) are standards for data transmission over fiber optic links. Due to long sequences of consecutive digits from incoming data streams, the clock signals become low and lead to delay in clock signal. This decrease the data transfer rate in OTN system. Hence complexity increase in OTN system and leads to over consumption of logic resources. Hence a need for clock recovery at the receiver, which in turn requires a guaranteed minimum number of transitions in the incoming serial data stream. The mechanism to achieve this transition density is known as scrambling. This paper presents the comparison between two scrambler architectures and indicates the most suitable solution for 100Gbit/s application.

Section II briefly describes the OTN scrambler and basic scrambler architecture. Section III and IV present scrambler architectures implemented with combinational logic (logical scrambler) and registers (registered scrambler), respectively. The comparison of these two architectures is presented in section V. Finally, conclusions are shown in section VI.

[2] OTN SCRAMBLER

The scrambling process can be carried by implementing an exclusive –OR operation between the transmitted information and the pseudorandom bit sequence (PRBS) generator.
The OTN scrambler can be described using serial scrambling architecture shown in Fig.1 uses the polynomial $x^{16} + x^{12} + x^3 + x + 1[5]$. This architecture depends on transmission frequency or data transfer rate. And the architecture can operate at frequency of 150 MHz for single input bit.

![Fig.1 Serial scrambler block diagram [5]](image)

The scramblers architectures are compared in this document have their PRBS circuits implemented in two different ways each circuit from now onwards treated as a “logical scrambler” and “registered scrambler”. These both cases are described in the following sections.

[3] **LOGICAL SCRAMBLER**

Fig.2 describes a general block diagram of a logical scrambler. The architecture is implemented using register set which are feedback through a combinational circuit [2] [3].

The pseudorandom signal generated by the circuit feed the output bus called $prbs[(L-1)..0]$, where L represents the number of output bits. The simplest example for this type generator shown in Fig. 1, where the output of the random sequence has only one bit.

In this serial scrambler, the initialization block forces the reset of all registers that keep a high logic level in their outputs. As per observation, in each clock cycle the register data are shifted to the right and the least significant bit is calculated from the combined output of the register $reg(0), reg(2), reg(11)$ and $reg(15)$.

![Fig.2. Logical scrambler block diagram [3] [7].](image)

This process continues until $2^{16}-1$ values are generated, and the sequence is then restarted. Hence FPGA operates at 724.638MHz clock frequency, the PRBS produces one bit for each clock cycle, hence, the data rate is 724.638Mbit/s [3][7].
An efficient way to double the data rate of the PRBS generator is to modify the PRBS circuit to generate two bits simultaneously instead of just one. In other words, working with the two output bits in the PRBS module, the data rate reaches 1500Mbit/s for the same 724.638MHz frequency clock shown in Fig.3.

In order to achieve 100Gbit/s throughput, it is necessary to generate 150 bits simultaneously instead of two bits. In other words, working with the 150 output bits in PRBS modules shown in Fig.4. (725MHz*150bits = 108Gbit/s).

Simulation Results:

The simulation results of logical scrambler for 100Gbit/s shown in Fig.5. The results shows 150 inputs in hexadecimal and corresponding 150 output. When reset is ‘1’, output is same as the input. If reset is ‘0’, output changes with each clock cycle. In above result one stage of logical scrambler architecture also shown with 2 bit input and 2 bit output. This behavior continues until $2^{15}-1$ values are generated, and the sequence is then restarted. Likewise above results of 2 bits, the sequence is restarted for 100Gbit/s after some values are generated.
[4] REGISTER SCRAMBLER

The architecture of the registered scrambler is very different from the logical scrambler. Fig.6 shows a block diagram of a generic register scrambler. In this case, the circuit consists basically on L generator blocks (GB), where L is the width of the pseudorandom data bus. Each GB block has an N-bit input and output bus, where N is the order of the polynomial generator. The most significant bit of each generator block is concatenated and registered to generate the pseudorandom output word (prbs_r)[3].

The GB blocks are obtained from the polynomial generation of the pseudorandom sequence. For this scrambler, which the polynomial generator is $x^{16} + x^{12} + x^{3} + x+1$, the GB circuit that generates an output signal is shown in Fig.7 [3]

The internal combinational circuit of the generator block for 100 Gbit/s OTN protocol is shown in Fig.7.
By using 16 generator blocks i.e \( L=16 \) in above architecture the frequency of 1461.561MHz can be achieved for 16 bits. In order to achieve 100Gbit/s throughput in register scrambler architecture, instead of 16 bit input it is necessary to use 80 bits inputs simultaneously and produces 80 bit outputs. The implementation of this architecture can shown in Fig.8. \((1461.5\text{MHz} \times 80 = 116\text{Gbit/s})\).

The below block diagram shown in Fig.8 is the cascading of 16 input register scrambler architecture to achieve 100Gbit/s throughput.
Simulation Results:

The simulation results of register scrambler in Fig. 10 show the 100Gbit/s scrambling architecture. And the results shows 80 bit inputs in hexadecimal and corresponding 80bits output. When reset is ‘1’, output is same as the input. If reset is ‘0’, output changes with each clock cycle. After 16 clock pulses the output becomes same as the input. In above result one stage of register scrambler architecture also shown with 16 bit input and output.

![Fig. 10 Simulation Results of Register Scrambler for 100Gbit/s.](image)

FPGA Result:

The Fig. 11 shows the FPGA result of register scrambler architecture. Here output seen in chip scope pro built-in software for virtex.

![Fig. 11 FPGA Result of Register Scrambler for 100Gbit/s.](image)
[5] SCRAMBLER ARCHITECTURES COMPARISON

The scramblers presented above were modeled and simulated in Verilog HDL to verify the correct operation. The models were synthesized with the necessary time constraints. The programmable logic synthesis and simulations were performed using the tools provided by Xilinx Inc. (ISE 13.4 and Modelsim6.3c) with a Virtex-6 as target device. This kind of FPGA device is designed with configurable logical blocks (CLB) interconnected through a routing matrix. One CLB element contains two slices and each slice has eight logical function generators or look-up-tables (LUT), eight registers, multiplexes and carry logic.

These elements are used by the synthesis process to implement combinational, arithmetic, and ROM functions. After the compilation, the ISE software generates an utilization and performance report. In order to measure the advantages and disadvantages of these architectures, the number of slices, registers and LUTs were utilized for comparison references. Moreover, the maximum acceptable operation frequency was used to compare this performance. Table I present the parameters used to compare the scramblers.

Additionally, a tool called Xilinx XPower Analyzer was used in order to estimate the power consumption of the device according to the implemented logic. The device consumption is an important parameter to be considered because it influences directly to the equipment performance. The estimation results were also included in Table I.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Logical Scrambler</th>
<th>Register Scrambler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Slices</td>
<td>17</td>
</tr>
<tr>
<td>Utilization</td>
<td>Register</td>
<td>163</td>
</tr>
<tr>
<td></td>
<td>LUTs</td>
<td>155</td>
</tr>
<tr>
<td>Performance</td>
<td>Max Frequency</td>
<td>725MHz</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Consumption</td>
<td>3.43W</td>
</tr>
</tbody>
</table>

[6] CONCLUSION

This paper presented two scrambler architectures, denoted by logical scrambler and registered scrambler, with applications in 100 Gbit/s optical transport network (OTN) systems. The results indicate that it is simpler to implement the registered scrambler because of the block generator used in the registered scrambler architecture is identical to any data width of the output bus. However, this leads to a cascade of logic that increases with the width of the output bus. And also register scrambler can achieve higher performance with lower FPGA resources occupation. According to Table I, the register scrambler uses lesser resource utilization than the logical scrambler, which represents a lower cost to the project. Comparing the frequency performance, the register scrambler is two times better than the logical scrambler. Therefore the most suitable architecture for 100 Gbit/s OTN applications is the register scrambler.
REFERENCES


