



2713 North 750 East · Lehi, Utah 84043

1-801-766-0285 · 1-801-280-9168

www.gutzlogic.com

PCI Express 16-bit Parallel Scrambler/De-scrambler

FEATURES

- Verilog RTL source code for scrambling or de-scrambling 2, 8-bit PCI Express symbols in parallel
- Separate skip_dectect, com_detect, and scramble_bypass signals for the two 8-bit symbols
- Simulation and hardware verified

Functional Description and Background

The purpose of the scrambler is to eliminate a repetitive pattern on the data stream. A repetitive pattern on a 2.5Gbs PCI Express data stream (such as 10101010) can generate significant EMI noise. By scrambling the data stream repetitive patterns are eliminated and thus spread the EMI energy over a broader range in the spectrum. This scrambling technique is often referred to as spread spectrum and is an effective way of whitening the noise.

PCI Express uses a Linear Feedback Shift Register (LFSR) to scramble the data stream. The LFSR implements the following polynomial:

$$G(x) = X^{16} + X^5 + X^4 + X^3 + 1$$

Traditionally the LFSR is clocked at the bit transfer rate. The output of the LFSR is XORed with the data to form the scrambled data. Figure 1 shows how the PCI Express scrambler polynomial is implemented serially.

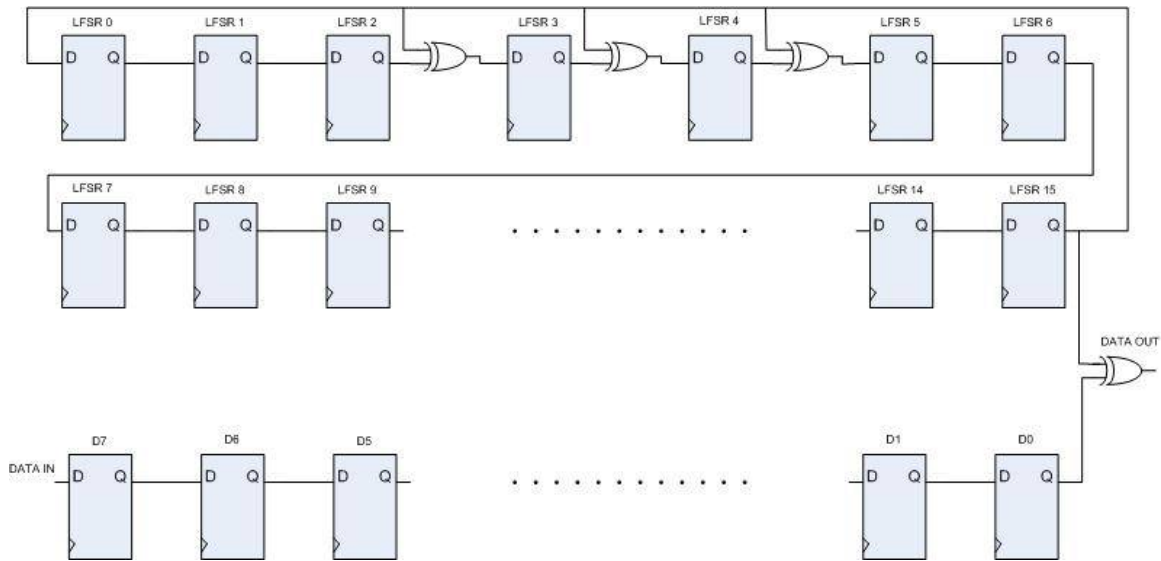


Figure 1 Serial Scrambler

With a data bit rate of 2.0Gbs, serially scrambling the data is not very practical. A method for implementing the LFSR in parallel needs to be developed to make any design practical. The most intuitive method would be to scramble the data 8-bits at a time. This effectively allows the designer to work with a byte rate of 250 MHz rather than the bit rate of 2Ghz. The 8-bit parallel implementation is fairly straight forward, and is shown in the following equations.

```

LFSR0 <= LFSR8;
LFSR1 <= LFSR9;
LFSR2 <= LFSR10;
LFSR3 <= LFSR11 ^ LFSR8;
LFSR4 <= LFSR12 ^ LFSR9 ^ LFSR8;
LFSR5 <= LFSR13 ^ LFSR10 ^ LFSR9 ^ LFSR8;
LFSR6 <= LFSR14 ^ LFSR11 ^ LFSR10 ^ LFSR9;
LFSR7 <= LFSR15 ^ LFSR12 ^ LFSR11 ^ LFSR10;
LFSR8 <= LFSR0 ^ LFSR13 ^ LFSR12 ^ LFSR11;
LFSR9 <= LFSR1 ^ LFSR14 ^ LFSR13 ^ LFSR12;
LFSR10 <= LFSR2 ^ LFSR15 ^ LFSR14 ^ LFSR13;
LFSR11 <= LFSR3 ^ LFSR15 ^ LFSR14;
LFSR12 <= LFSR4 ^ LFSR15;
LFSR13 <= LFSR5;
LFSR14 <= LFSR6;
LFSR15 <= LFSR7;

```

The previous equations essentially advance the LFSR 8 bits every clock cycle. One needs now to implement the following equations to scramble 8-bits and a time.

```
data_out0 <= LFSR15 ^ data_in0;
data_out1 <= LFSR14 ^ data_in1;
data_out2 <= LFSR13 ^ data_in2;
data_out3 <= LFSR12 ^ data_in3;
data_out4 <= LFSR11 ^ data_in4;
data_out5 <= LFSR10 ^ data_in5;
data_out6 <= LFSR9 ^ data_in6;
data_out7 <= LFSR8 ^ data_in7;
```

Scrambling 8-bits at a time in parallel is straight forward, and easy to implement, but the operating frequency of 250 MHz is still rather high and might be hard to work with if designers are using FPGA's.

Implementation

Implementing the PCI Express scrambler/de-scrambler in a 16-bit parallel fashion would essentially halve the frequency designers would be required to work with (125 MHz). This would greatly help designers to meet timing much easier when working with FPGA's.

The implementation of the PCI Express Polynomial in parallel 16-bits at a time is not as trivial as the 8-bit parallel implementation and thus causes it to have greater value. One needs to take into account that one of the 8-bits could be a COM character and the other could be a data character or visa versa. This is one of the many complications that have to be addressed when implementing the scrambler/de-scrambler in parallel 16-bits at a time. Below are some of the Scrambler implementation rules that Gutz Logic's 16-bit parallel Scrambler/de-scrambler addresses.

- Scrambling is applied to Data characters associated with DLLPs and TLPs including logical idles and Data characters TS1 and TS2 ordered sets.
- K characters are not scrambled and bypass the scramble logic.
- Compliance pattern related characters are not scrambled.
- When a COM character exits the scrambler the COM does not get scrambled, but it initializes the LFSR to 16'hFFFF. Similarly on the receive side the COM character initializes the de-scrambler to 16'hFFFF.
- The LFSR does not advance on SKP characters.

The Gutz Logic 16-bit Parallel Scrambler/De-scramblers works very well with TI's XIO1100 X1 PHY chip when running in 16-bit mode.