ESIstream

The **E**fficient **S**erial **I**nterface

Protocol Specification

Version 2.1

ESIstream is license-free high efficiency serial interface protocol based on 14b/16b encoding.
Its main benefits are low overhead and ease of hardware implementation.

ESI stream

TABLE OF CONTENTS

FIGURES

Documents Amendment

Issue	Date	Comments
1.0	November 2014	Creation
2.0	October 2016	Precision specified around the boundaries of the protocol.
2.1	October 2020	Terminology, template and figures update.

Terminology

AC	Alternating Coupling
ADC	Analog to Digital Converter
ASIC	Application-Specific Integrated Circuit
CDR	Clock and Data Recovery
DAC	Digital to Analog Converter
ESS	ESIstream Synchronization Sequence
FAS	Frame Alignment Sequence
FPGA	Field Programmable Gate Array
ILA	Integrated Logic Analyzer (a Vivado feature)
LD	Logic Device
LFSR	Linear Feedback Shift Register
PAS	PRBS Alignment sequence
PL	Programmable Logic
PLL	Phase Locked Loop
PRBS	Pseudo-Random Binary Sequence
RX	Receiver
TX	Transmitter
UI	Unit Interval (period of time to send a bit through the serial lane)
Xcvr	Transceiver

1. ESISTREAM PROTOCOL

1.1 Overview

ESIstream protocol is born from a severe need of the following combination:

- Increase rate of useful data, when linking data converters operating at GSps speeds with FPGAs on a serial interface, reducing data overhead on serial links, as low as possible.
- Simplified hardware implementation, simple enough to be built on RF SiGe technologies.

ESIstream provides an efficient High-Speed serial interface based on a 14B/16B encoding using a Linear Feedback Shift Register (LFSR) scrambling unit, a Disparity Bit (DB) to ensure deterministic DC balance transmission and a toggling bit, the Clk Bit (CB), to enable synchronization monitoring.
It is **license-free** and supports in particular serial communication between FPGAs and High-Speed data converters.
However, ESIstream can be used in any system requiring a serial interface. For instance, between two FPGAs or two ASICs.

An ESIstream system is made up of the following elements.

- A transmitter (TX) can be an ADC or any Logic Devices (LD) such as a FPGA or an ASIC.
- A receiver (RX) can be a DAC or any Logic Devices such as a FPGA or an ASIC.
- A number of serial lanes ($L \geq 1$) to transmit serial data.
- A synchronization signal (sync) used to initialize the communication and synchronize the transmitter and receiver. On a single device, only one occurrence of the SYNC signal is necessary between the transmitter and the receiver even if multiple serial links are implemented.

There is no clock lane in a serial interface. For each lane, the receiver should recover the clock from the data.

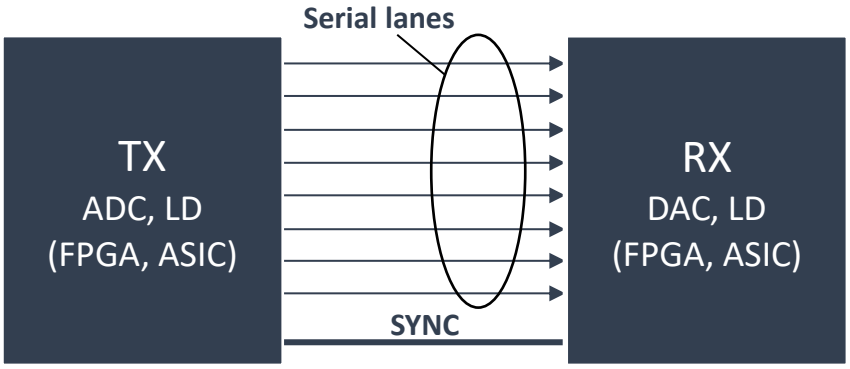

Figure 1: Basic ESIstream system

1.2 TX and RX architecture

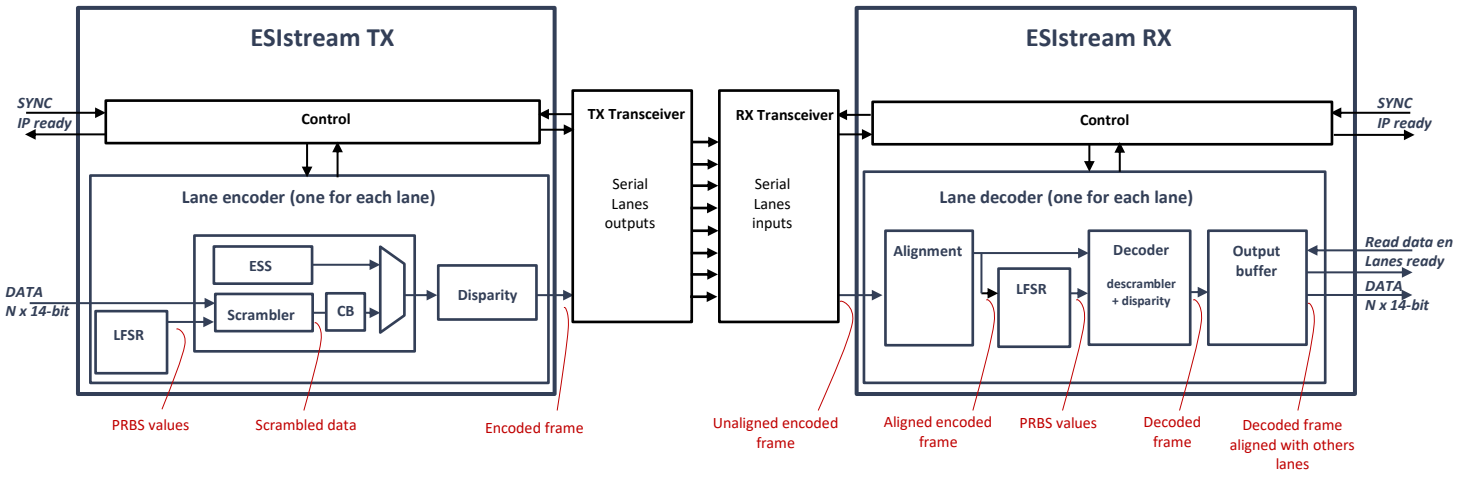

Figure 2: ESIstream TX and RX architecture overview

Different stages of encoding/decoding are implemented to manage the limitations of a serial interface.
An AC coupled interface requires a DC balance, otherwise the AC coupling link will tend to drift and the received data will be corrupted.
The Clock Data Recovery (CDR) in the RX transceiver stage, which usually contains a Phased Locked Loop (PLL), specifies a minimum number of transitions over a period of time. Otherwise, the PLL can lose its lock.

Scrambling, Clk bit and Disparity processing ensure a deterministic DC balance transmission between +/- 15 and a maximum run length of 32 Unit interval (UI).
The run length defines a number of UI without transitions on the serial lane.

1.3 Frame

An ESIstream frame is 16-bit wide. The frame is transmitted, LSB first, between the transmitter and the receiver.
ESIstream uses a 14B/16B encoding, 14-bit of useful data are scrambled and then 2-bit of overhead are concatenated to create the 16-bit ESIstream frame.
The overhead is composed of:

- The Clk Bit (CB), which is toggling between each consecutive frame sent through a single serial lane. The Clk bit can be used to monitor the synchronization of a single lane.
- The Disparity Bit (DB), which is the result of a calculation, the disparity processing, done on the 14-bit data and on the Clk Bit.

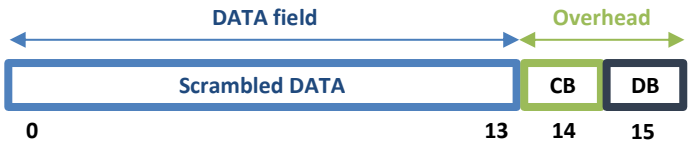

Figure 3: ESIstream encoded frame

These different stages of encoding are realized to manage the limitations of a serial interface. First, an AC coupled interface between transmitter and receiver implies that the transmission be DC balance, otherwise the AC coupling capacitor will drift and the received data will be corrupted. Secondly the CDR in the reception stage usually contains a PLL. This means that there must be transitions in the transmission otherwise this PLL will lose its lock.
The scrambling, clk bit and disparity processing ensure a deterministic transmission with a DC balance between +/- 15 and a max run length of the transmission of 32.
Note: Current CDR can work with max run length of 80 and above.

1.4 Scrambling

Scrambling ensures a statistical DC balanced transmission. It also statistically ensures that there are transitions in the transmission.
ESIstream uses an additive scrambling to avoid error propagation in case of a single bit error. The Linear Feedback Shift Register (LSFR), which generates a Pseudo Random Binary Sequence (PRBS), is based on a Fibonacci architecture and uses the polynomial $X^{17}+X^3+1$. It has a length of 2^{17} - 1. See Annex A for more information.
Instead of using a shift of one bit per operation, it uses shifts of 14 bits per operation to adapt to the size of the data being scrambled.

The ESIstream LFSR is characterized by the following equations:

$$
\begin{aligned}
LFSR_{n+1}(0) &= LFSR_n(14)\\
LFSR_{n+1}(1) &= LFSR_n(15)\\
LFSR_{n+1}(2) &= LFSR_n(16)\\
LFSR_{n+1}(3) &= LFSR_n(0) \text{ xor } LFSR_n(3)\\
LFSR_{n+1}(4) &= LFSR_n(1) \text{ xor } LFSR_n(4)\\
LFSR_{n+1}(5) &= LFSR_n(2) \text{ xor } LFSR_n(5)\\
LFSR_{n+1}(6) &= LFSR_n(3) \text{ xor } LFSR_n(6)\\
LFSR_{n+1}(7) &= LFSR_n(4) \text{ xor } LFSR_n(7)\\
LFSR_{n+1}(8) &= LFSR_n(5) \text{ xor } LFSR_n(8)\\
LFSR_{n+1}(9) &= LFSR_n(6) \text{ xor } LFSR_n(9)\\
LFSR_{n+1}(10) &= LFSR_n(7) \text{ xor } LFSR_n(10)\\
LFSR_{n+1}(11) &= LFSR_n(8) \text{ xor } LFSR_n(11)\\
LFSR_{n+1}(12) &= LFSR_n(9) \text{ xor } LFSR_n(12)\\
LFSR_{n+1}(13) &= LFSR_n(10) \text{ xor } LFSR_n(13)\\
LFSR_{n+1}(14) &= LFSR_n(11) \text{ xor } LFSR_n(14)\\
LFSR_{n+1}(15) &= LFSR_n(12) \text{ xor } LFSR_n(15)\\
LFSR_{n+1}(16) &= LFSR_n(13) \text{ xor } LFSR_n(16)
\end{aligned}
$$

The PRBS is applied to the data with a bitwise XOR binary operation:

DATA[13:0] **XOR** PRBS[13:0] = DATA_SCRAMBLED[13:0]

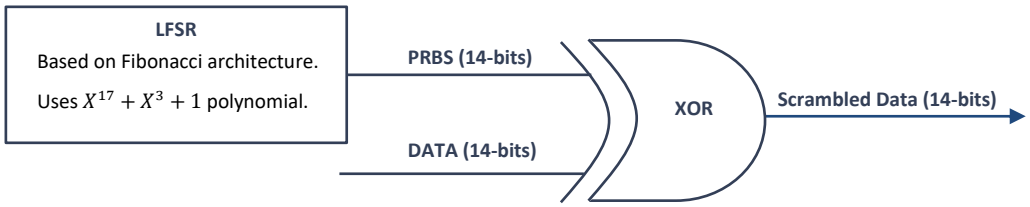

Figure 4: Scrambling principle, bitwise XOR binary operation

In case of a multi-lane interface, in order to reduce correlation between lanes, each lane should have different initial values for the scrambling units.

This particular LFSR was chosen because it reduces the number of necessary gates to be implemented.

1.5 Encoding

Scrambled data (14-bit) are encoded are encoded into a 16-bit frame adding two overhead bits, the Clk Bit (CB) and the Disparity Bit (DB).

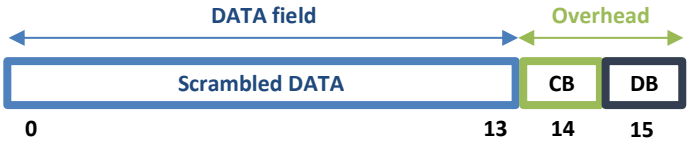

Figure 5: ESIstream encoded frame

1.5.1 Clk Bit (CB)

On each serial lane, the Clk Bit (CB) toggles at every ESIstream frame sent through one serial lane.
The receiver uses the Clk Bit to monitor the link synchronization. If the receiver does not detect that the Clk Bit is toggling properly, then it can state that the link is not synchronous or has lost its synchronization and restart the synchronization process.

Figure 6: Clk Bit toggling properly on one serial lane.

1.5.2 Disparity Bit (DB)

The Disparity Bit ensures deterministically the advantages brought statistically by the scrambling process.

Even with scrambling process, a large running disparity can still occur with very low probability and could produce excessive eye shifts. These eye shifts could be balanced by a more complicated equalization stage in the receiver if the running disparity was still limited. However, a PRBS does not bind the running disparity deterministically, thus the data could be corrupted on the reception end and it could eventually cause the PLL in the CDR to lose its lock. The implementation of the Disparity Bit process prevents from this eventuality.

The transmitter constantly monitors the disparity of the transmission.
For each frame, the running disparity is calculated and 2 cases can occur:

- The running disparity of the transmission **does not** increase above ±16 (+16 and -16 included). In this case, the disparity bit is set to '0' and the 15 bits composed of the scrambled data and of the Clk Bit are transmitted as such.
- The running disparity of the transmission **does** increase above ±16 (+16 and -16 excluded). In this case, the 15 bits composed of the scrambled data and of the Clk Bit are inverted and the disparity bit is set to '1'.

The running disparity is updated with the disparity of the frame. See Annex C for an example.

The disparity bit ensures that the longest possible series of '1' or '0' transmitted is of 48 bits and the Clk Bit reduces the effective value to 32.
The disparity bit also ensures that the running disparity does not exceed ± 16 (included) which satisfies the DC balance condition.

In normal operating mode, the receiver will check the disparity bit first. If it is high then it will invert the received data and descramble them. Otherwise it will only descramble them. From this point the data are available for processing.

1.6 ESIstream Synchronization Sequence (ESS)

Each serial lane must be synchronized to align the frames between the transmitter and the receiver and to synchronize the reception scrambler with the transmission scrambler.

The synchronization is controlled through the synchronization (SYNC) signal sent to the receiver and to the transmitter.
The receiver must receive the SYNC signal prior to receive the ESIstream Synchronization Sequence (ESS).

The transmitter generates the ESS when receiving the SYNC signal. The length of the SYNC pulse should be at least as long as one ESIstream frame period (16-bit).

The ESS is composed of two parts:

- The Frame Alignment Sequence (FAS), 32 frames alternating between 0xFF00 and 0x00FF. This sequence bypasses the scrambling, the Clk Bit and the Disparity Bit processing (the sequence is DC balanced). This alignment pattern (COMMA or FLASH pattern) is used by the receiver to align its data on the transmitter output data.

- The PRBS Alignment Sequence (PAS), 32 additional frames containing the scrambling PRBS alone. These frames contain 14 bits of the PRBS plus the Clk Bit and the Disparity Bit. These frames go through the disparity processing, as the PRBS value will start to impact the running disparity of the transmission.

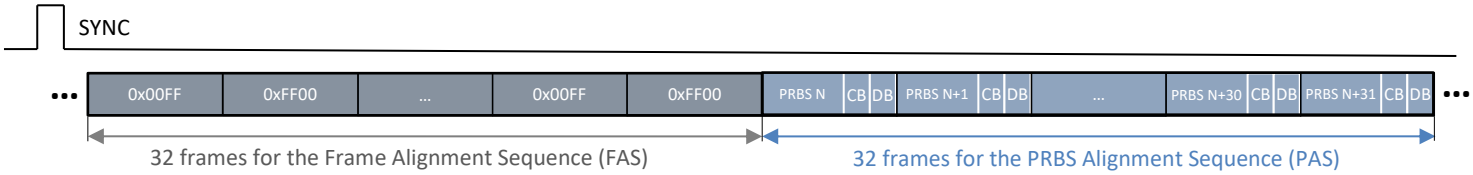

Figure 7: ESIstream Synchronization Sequence (ESS)

When it is received by the transmitter, it will send an alignment pattern which is 32 frames alternating between 0xFF00 and 0x00FF. The sequence bypasses the scrambling and disparity processing (the sequence is DC balanced). This alignment pattern is used by the receiver to align its data on the transmitter output data.
After these 32 frames, the transmitter starts sending 32 additional frames containing the scrambling PRBS alone. These frames contain 14 bits of the PRBS plus the clk bit and the disparity bit. They go through the disparity processing, as the PRBS value will start to impact the running disparity of the transmission.

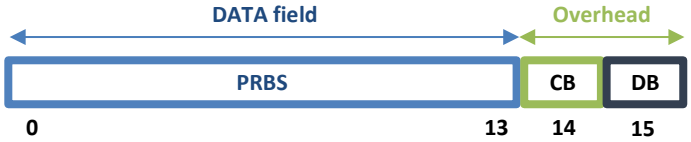

Figure 8: PRBS frames sent during the PRBS Alignment Sequence

The receiver will detect the transition from the alignment pattern to the PRBS alone. The PRBS is reset by the transmitter when receiving the SYNC to avoid the first frame of the PRBS initialization being either 0x00FF or 0xFF00; this to ensure that passive detection is precise to the frame.
The receiver will determine its PRBS initial value after receiving 2 valid frames of PRBS Alignment Sequence. These 2 frames contain 28 bits of the PRBS sequence; the receiver needs 17 bits to determine its PRBS initial value. After that, the synchronization of the link is complete.

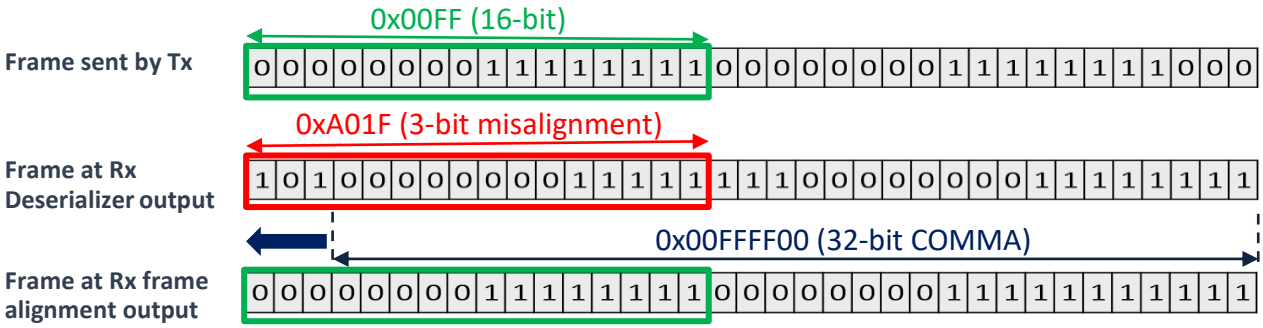

Figure 9: Receiver (RX) Frame alignment principle

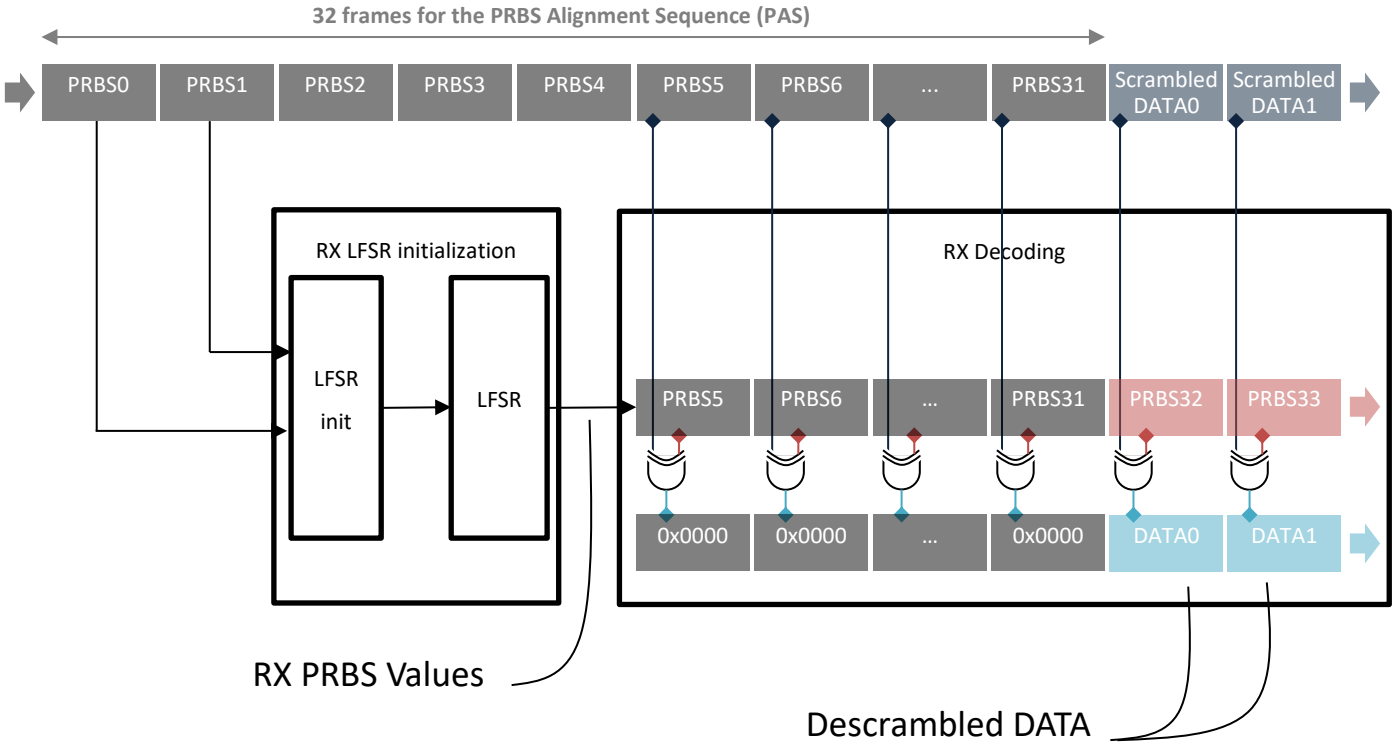

Figure 10: Receiver (RX) descrambling principle

2. ANNEXA: PRBS ARCHITECTURE

2.1 Fibonacci architecture definition:

A LFSR is used as a generator for a pseudo-random binary sequence to scramble the serial link data.

It is defined by a polynomial: $G(X) = g_m X^m + g_{m-1} X^{m-1} + g_{m-2} X^{m-2} + \ldots + g_2 X^2 + g_1 X + g_0$ and an initial value.

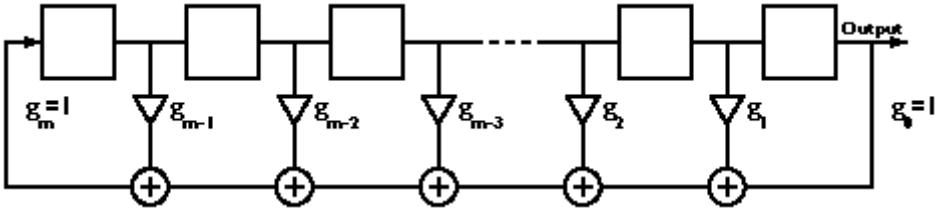

Figure 11: Fibonacci implementation of LFSR

2.2 Fibonacci architecture example:

For a 4 bits LFSR of the polynomial X^4+X^1+1:

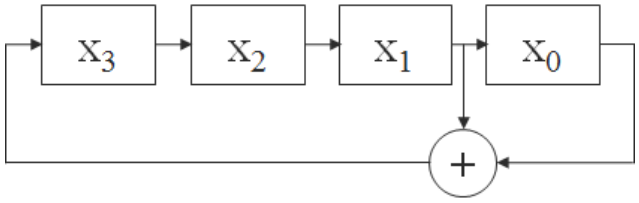

Figure 12: LFSR Fibonacci serial architecture

2.3 Fibonacci parallelized architecture:

For a 4 bits LFSR of the polynomial X^4+X^1+1, with a 4 steps parallel architecture:

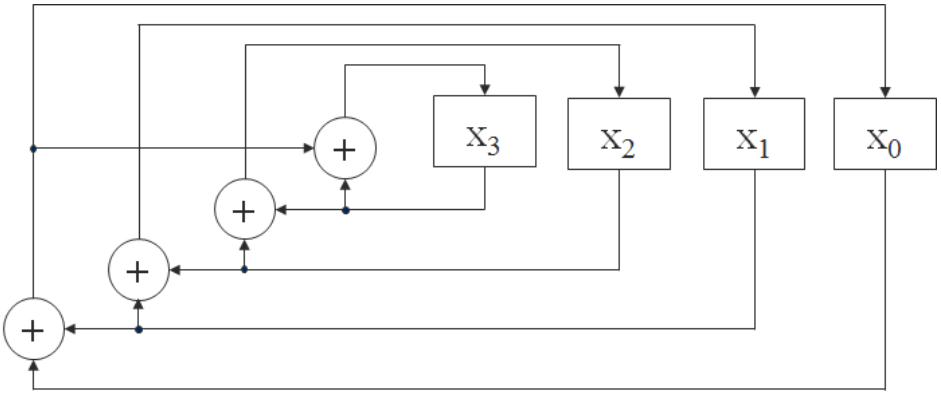

Figure 13: LFSR Fibonacci 4 step parallel architecture

This architecture does in one clock period the same operation that the serial architecture does in 4 clock periods. It keeps the same length.

3. ANNEX B: EXAMPLE OF DISPARITY BIT IMPLEMENTATION

During a transmission, the running disparity (RD) is constantly monitored. At some instant t_i, the running disparity of the transmission is -5. It means that from the beginning of the transmission, there were 5 more '0' sent than '1'.
The next 15 bits of data (14 bits scrambled + clk bit) to be sent are below:

$RD(t_i) = -5$

t_i	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0

$D_w = -7$

The disparity of the word (D_w) is -7. The updated running disparity of the transmission is:

$RD(t_{i+1}) = RD(t_i) + D_w + DB = -5 - 7 - 1 = -13$

DB = +1 if the disparity bit is '1'
DB = -1 if the disparity bit is '0'

The updated running disparity is still between -16 and +16, the data are transmitted as such and the disparity bit is '0':

t_i	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0 ← DB

The next 15 bits of data to be sent are:

t_{i+1}	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1

$D_w = -5$

The updated running disparity of the transmission is:

$RD(t_{i+2}) = RD(t_{i+1}) + D_w + DB = -13 - 5 - 1 = -19$

The running disparity exceeds -16, the data are inverted and the disparity bit is set to '1'.

t_{i+1}	1	1	1	1	1	0	0	0	1	0	1	1	1	1	0	1 ← DB

The running disparity becomes:

$RD(t_{i+2}) = RD(t_{i+1}) + D_w + DB = -13 + 5 + 1 = -7$

The next data are then processed using the same algorithm.