Nano Scale Disruptive Silicon-Plasmonic Platform for Chip-to-Chip Interconnection

Interface and plasmonic interconnect models and reports

Deliverable no.:	D2.4
Due date:	10/31/2013
Actual Submission date:	11/15/2013
Authors:	ST
Work package(s):	WP2
Distribution level:	RE[1] (NAVOLCHI Consortium)
Nature:	document, available online in the restricted area of the NAVOLCHI webpage

List of Partners concerned

Partner number	Partner name	Partner short name	Country	Date enter project	Date exit project
1	Karlsruher Institut für Technologie	KIT	Germany	M1	M36
2	INTERUNIVERSITAIR MICRO-ELECTRONICA CENTRUM VZW	IMEC	Belgium	M1	M36
3	TECHNISCHE UNIVERSITEIT EINDHOVEN	TU/e	Netherlands	M1	M36
4	RESEARCH AND EDUCATION LABORATORY IN INFORMATION TECHNOLOGIES	AIT	Greece	M1	M36
5	UNIVERSITAT DE VALENCIA	UVEG	Spain	M1	M36
6	STMICROELECTRONICS SRL	ST	Italy	M1	M36
7	UNIVERSITEIT GENT	UGent	Belgium	M1	M36

[1] **PU** = Public
PP = Restricted to other programme participants (including the Commission Services)
RE = Restricted to a group specified by the consortium (including the Commission Services)
CO = Confidential, only for members of the consortium (including the Commission Services)

Deliverable Responsible

Organization:	STMicroelectronics
Contact Person:	Alberto Scandurra
Address:	Stradale Primosole, 50 – 95121 Catania Italy
Phone:	+39 095 740 4432
Fax:	+39 095 740 4008
E-mail:	alberto.scandurra@st.com

Executive Summary

This document presents the analysis and the characterization of the interface modules placed between the Bi-synchronous FIFO and the analog blocks interacting with plasmonic devices; then, it describes VHDL model of the plasmonic interconnect. It starts with a brief introduction to the general approach used and to the top-level design, and then it analyses the behavioural models of the individual blocks.

Change Records

Version	Date	Changes	Author
0.1 (draft)	2013-11-05	Start	Alberto Scandurra, Valentina Cernuto
1 (submission)	2013-11-15	Final version	Alberto Scandurra, Valentina Cernuto

Contents

1. Introduction

The purpose of the following analysis work is to establish the best arrangement of the digital interface modules in terms of minimization of the number of ‘1’ and transactions between two back-to-back data words. Then the VHDL model of the plasmonic interconnect is described in terms of functionality and structure.

2. Interface modules analysis and characterization

The whole interface module consists of the Optical Bus Inverter Encoder (OBI_Enc), the Serializer (SER) and the Bus Inverter Encoder (BI_ENC).

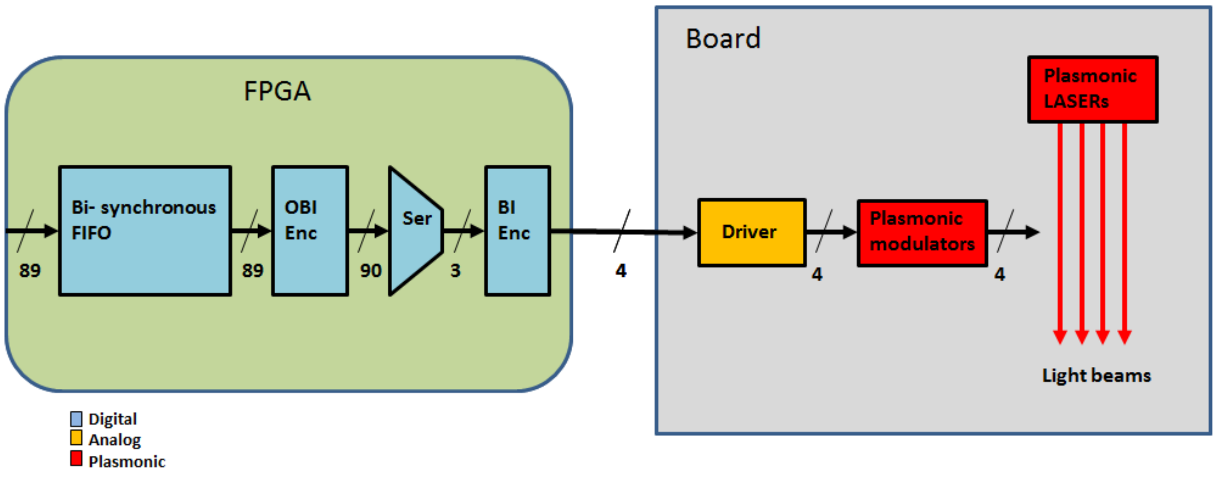

Figure 2-1: NAVOLCHI demonstrator PHY adapter transmitter

The OBI_Enc is responsible for minimizing of ‘1’ in 89-bits data word to be transmitted, in order to keep turned on the minimum number of plasmonic emitters; the SER performs the segmentation of the incoming data according to the selected output data size and the BI_ENC is responsible for minimization of the Hamming distance between two back-to-back 4-bits data words in order to minimize the switching activity of each laser.
We carried out several tests placing the *C language model* of the blocks OBI_Enc, SER and BI_Enc alternately in a different sequence; in each one we used the same input vectors.
The analysis results and the characterization of the different arrangements in terms of area occupation, latency and power consumption are shown in the following table.

Arrangement	Average Gain %	Area $[\mu m^2]$	Latency	Power consumption $[mW]$
OBI_Enc – SER – BI_Enc	60	4320,39	4	2,183
SER – OBI_Enc	41	4244,01	2	2,137
SER – BI_Enc	38	1350,97	2	0,406
OBI_Enc – SER	23	4244,01	3	2,137
BI_Enc – SER – OBI_Enc	20	4320,39	4	2,183
BI_Enc – SER	9	1350,97	3	0,406
SER	0	1274,59	1	0,360

Table 2-1: Analysis Results

The *Average Gain %*, shown in the table, is the arithmetic mean of the percentage gains obtained for each bit vector taken as input sample. This gain was calculated as the ratio between the percentage of '0 'obtained in output compared to that present at the input. Input vectors consisting of all ‘1’ or all ‘0’ have been excluded from the tests, because they represent two limiting cases of zero and infinite gain, regardless of the arrangement of the blocks.
As shown in the previous table, the best arrangement is OBI_Enc-SER-BI_Enc in terms of minimization of the number of ‘1’ and transactions between two back-to-back data words. However, the synthesis reports show that it is the worst solution in terms of area occupation, latency and power consumption.

3. Plasmonic interconnect models

The approach, followed in the VHDL models of the Plasmonic Interconnect, is top-down type, for more details see the MS6 Milestone.
The top-level design consists of the following seven blocks connected in cascade:

- the modulator driver
- the laser
- the plasmonic waveguide
- the plasmonic amplifier
- the plasmonic photodetector
- the TIA (Trans-Impedance Amplifier)
- the CRC (Clock Recovery Circuit)

The block diagram, shown in the following figure, reflects the structural description of the *plasmonic_interconnect* entity as reported in the *plasmonic_interconnect_ent.vhd* file; this entity also includes the analog blocks, present in the transmitter and in the receiver, and the digital Clock Recovery block.
The transmitted data through the plasmonic interconnect is a 30 bits serial data and the proposed VHDL model concerns a single bus of the four that are present in Navolchi demonstrator.

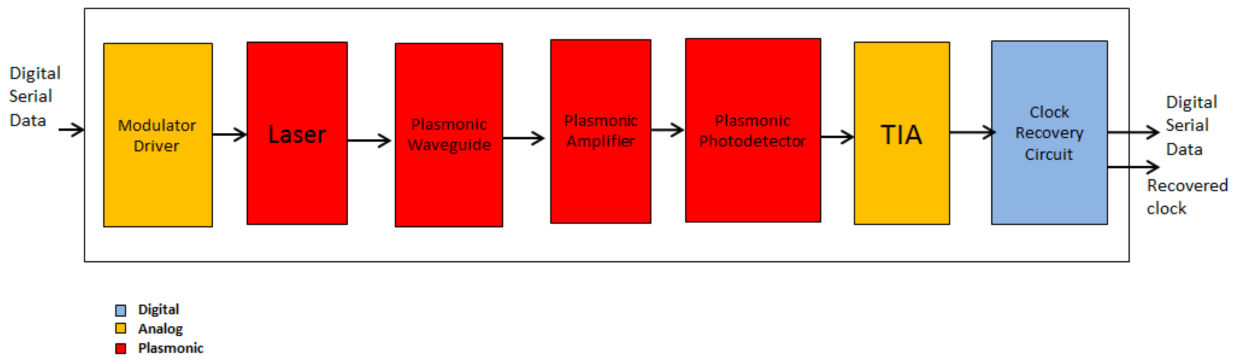

Figure 3-1 : Plasmonic Interconnect Block Diagram (Top-level)

We used the REAL types for the current and optical power representations.

Modulator Driver

This block has the task to generate the current levels in order to modulate properly the light emitted by the laser, starting by the stream of bits which have to be transported through the plasmonic interconnect.
The *modulator_driver* entity has a *std_logic* type as input (named *dig_in*) and a REAL type as output (named *analog_out*). It is shown in the following figure.

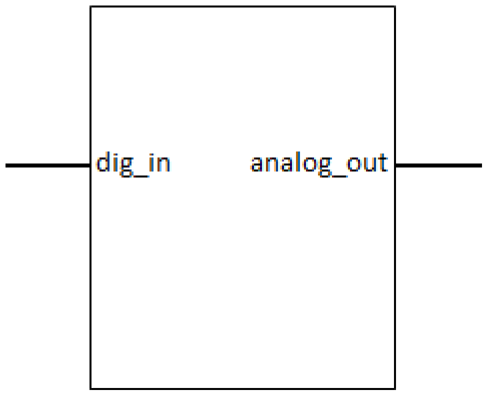

Figure 3-2 : Modulator Driver Entity

In the architecture of *modulator_driver* entity we declared three CONSTANT values which represent the current generated according to the digital input which can be a logic ‘1’, a logic ‘0’ or simple noise, i.e. input signal absence (‘U’).
The *modulator_driver* behavioural model is summarized in the following table.

dig_in	analog_out
‘1’	I_{high}
‘0’	I_{low}
‘U’	I_{dark}

Table 3-1 : Behavioural Table

Metallic nanolaser

This block acts as a current-to-optical power converter, taking the current generated by the modulator driver and generating a proper optical power.
The *laser* entity has a REAL type as input (named *current_in*) and a REAL type as output (named *power_out*). It is shown in the following figure.

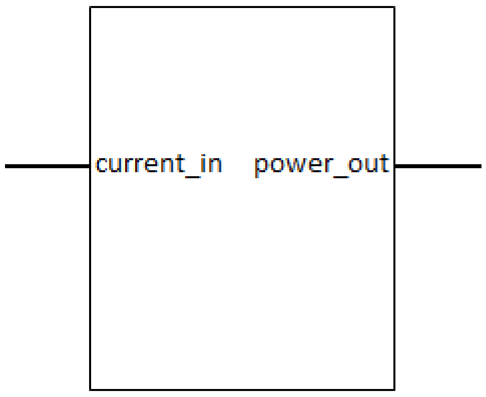

Figure 3-3 : Laser Entity

Also in this case, in the architecture of the *laser* entity we defined the reference values for currents and optical powers as CONSTANT and the behavioural model is described by following table.

current_in	power_out
$\geq I_{high}$	P_{high}
$> I_{dark}$	P_{low}
other values	P_{dark}

Table 3-2 : Behavioural Table

Plasmonic Waveguide

This block has the task to transmit the optical power.
The *plasmonic_waveguide* entity has a REAL type as input (named *power_in*) and a REAL type as output (named *power_out*). It is shown in the following figure.

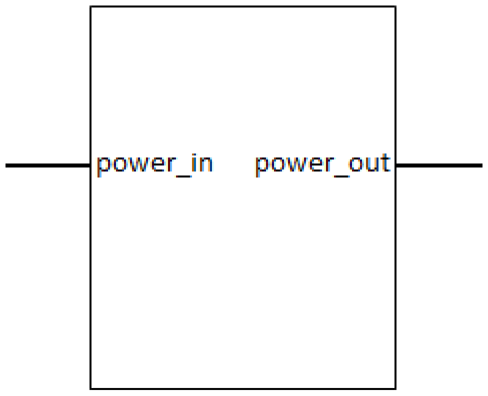

Figure 3-4 : Plasmonic Waveguide Entity

Considering the signal delay and the energy dissipation per transmitted bit introduced by the plasmonic waveguide, we defined a *loss* (G < 1) and a *delay* as CONSTANT values and so the output signal is reduced in amplitude and retarded compared to input signal.

Plasmonic Amplifier

Plasmonic devices are lossy and the task of the amplifier block is to amplify the input optical power.
The *plasmonic_amplifier* entity has a REAL type as input (named *power_in*) and a REAL type as output (named *power_out*). It is shown in the following figure.

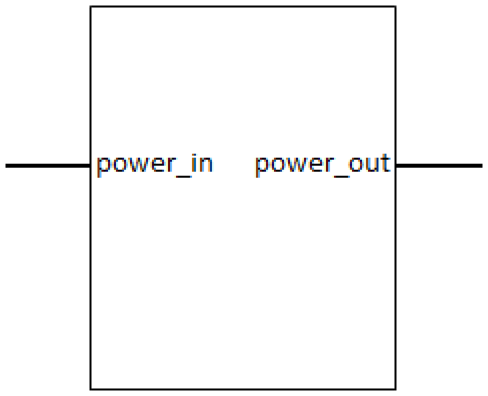

Figure 3-5: Plasmonic Amplifier Entity

In the architecture of *plasmonic_amplifier* entity we defined a *gain* (G >1) as CONSTANT value.

Plasmonic Photodetector

This block acts as an optical power-to-current converter, taking the optical power generated by the plasmonic amplifier and generating a current representing a high level optical power, a low level optical power or simple noise (i.e. input signal absence).
The *plasmonic_photodetector* entity has a REAL type as input (named *light_power_in*) and a REAL type as output (named *current_out*). It is shown in the following figure.

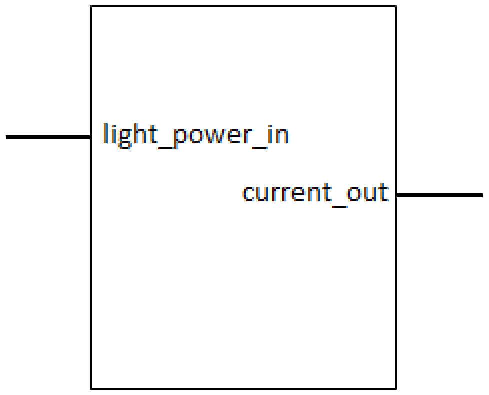

Figure 3-6 : Plasmonic Photodetector Entity

In the architecture of *plasmonic_photodetector* entity we defined the reference values for currents and optical powers as CONSTANT.
The plasmonic photodetector behavioural model is summarized in the following table.

light_power_in	current_out
$\geq P_{high}$	I_{high}
$> P_{dark}$	I_{low}
other values	I_{dark}

Table 3-3 : Behavioural Table

TIA (Trans-Impedance Amplifier)

This block acts as a current-to-voltage converter, taking the photocurrent generated by the plasmonic detector and generating a proper voltage, representing a logic ‘0’, a logic ‘1’ or simply noise (i.e. no detection).
The *TIA* entity has a REAL type as input (named *current_in*) and a *std_logic* type as output (named *voltage_out*). It is shown in the following figure.

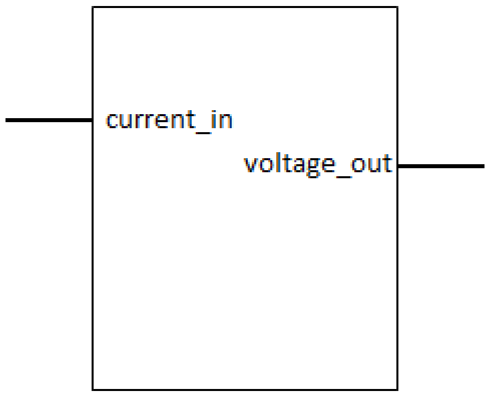

Figure 3-7 : TIA Entity

In the architecture of *TIA* entity we defined the reference currents as CONSTANT values. The digital output is ‘1’, ‘0’ or ‘U’ in accordance with the following table which summarizes the behavioural model of the TIA.

current_in	voltage_out
$\geq I_{high}$	‘1’
$> I_{dark}$	‘0’
other cases	‘U’

Table 3-4 : Behavioural Table

CRC (Clock Recovery Circuit)

Serial data stream could have multiple interpretations in the Receiver as shown in the following figure.

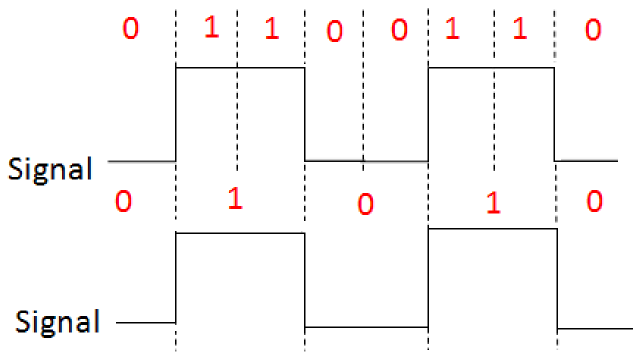

Figure 3-8: Example of double interpretation of one and only signal

There is need to recover the clock from the data stream and to use it to sample the data in order to extract the individual data bits. This task is carried out by a *Clock Recovery Circuit*. It allows the Receiver to regenerate serial data with the fewest bit errors and to obtain a correct interpretation of itself.
To describe *crc* entity we used a structural model in which some subcomponents, that perform smaller operations of the complete model, are instantiate.
This description uses three lower-level components to model the behavior of the CRC:

- an edge detection circuit component;
- a counter component;
- a D-flip flop component.

The *crc* top-level is shown in the following figure.

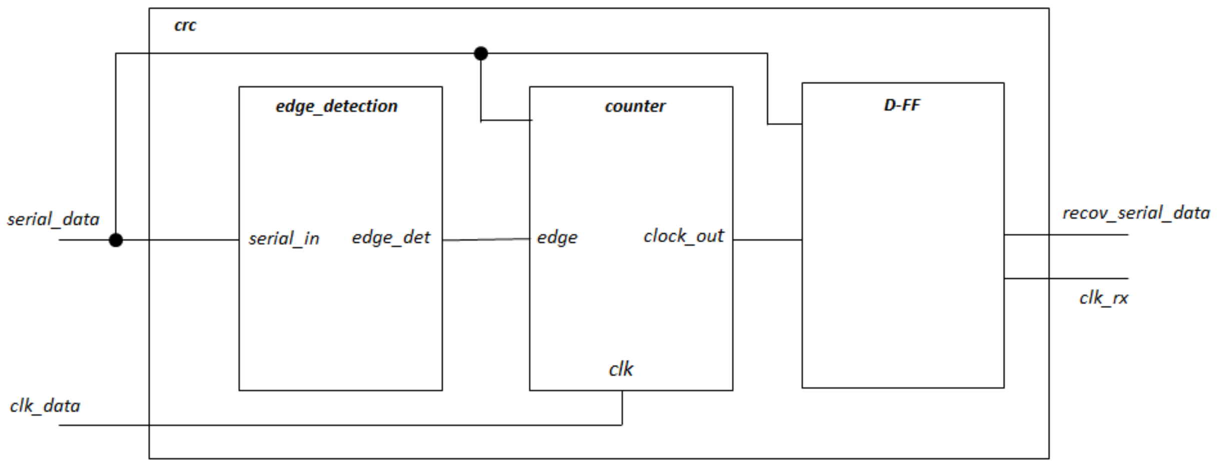

Figure 3-9 : CRC Top-level

Its inputs are:

- *serial_data* declared as *std_logic*;
- *clk_data* declared as *std_logic*. It's the clock of transmitted serial data.

Its outputs are *recov_serial_data* and clk_rx, declared as a *std_logic*.
The *edge_detection* entity has a *std_logic* as input (named *serial_in*) and a *std_logic* as output (named *edge_det*). Its functionality, described in the architecture of *edge_detection* entity, is to generate a pulse when '0' to '1' or '1' to '0' transition of the incoming data occurs.
To describe this block we used a structural model consisting of three INVERTER gates, a NAND gate e two D-flip flop.
The edge detection circuit top-level is shown in the following figure.

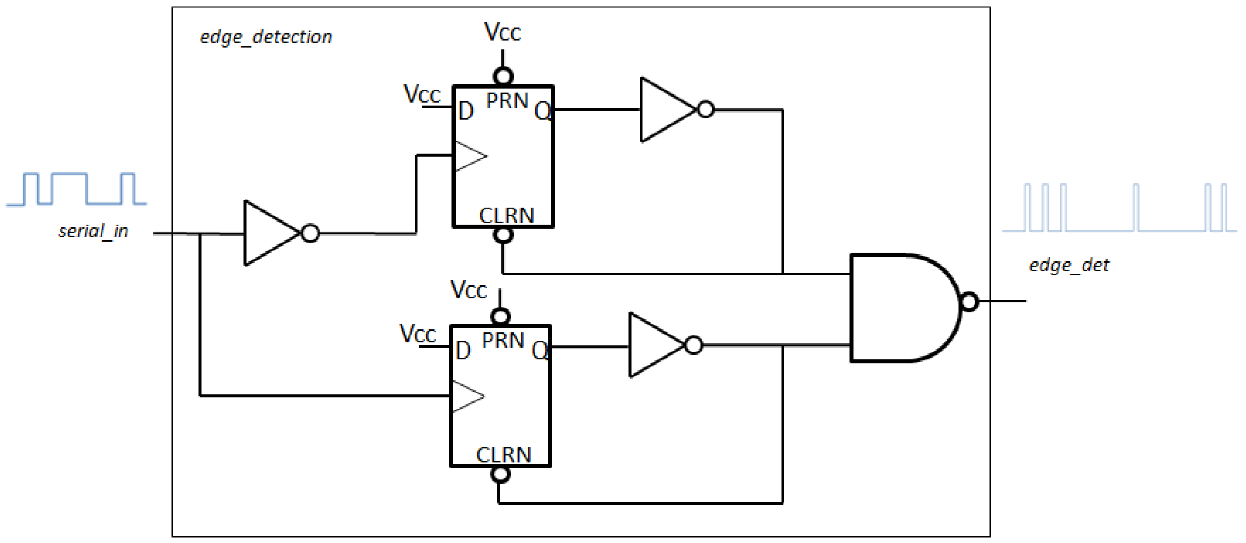

Figure 3-10 : Edge Detection Circuit Entity (Top-level)

The upper flip-flop generates a pulse from a negative transition of the incoming data, while the lower generates a pulse from a positive transition. Pulse width depends on flip-flop and inverter delay.
The second block contains a 3 bit counter whose clock signal is approximately 8 times the data rate and the MBS of its output is used as the Sampling Point of data. This is approximately in the centre of the data bit. Moreover, an input transition gets reset the counter.
The *counter* entity has two *std_logic* as inputs (named *edge* and *clk*), a *std_logic* as output (named *clk_out*).

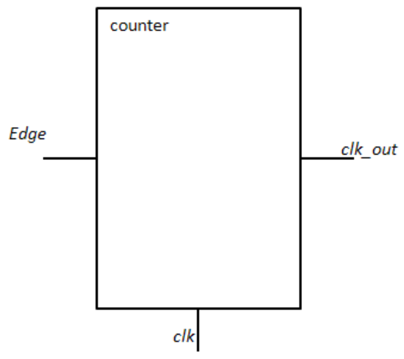

Figure 3-11 : Counter Entity

If an edge signal occurs, the counter is reset early. If there are successive bits of the same states, the counter *free-runs* and continues to sample the data correctly [1]. It is shown in the following figure.

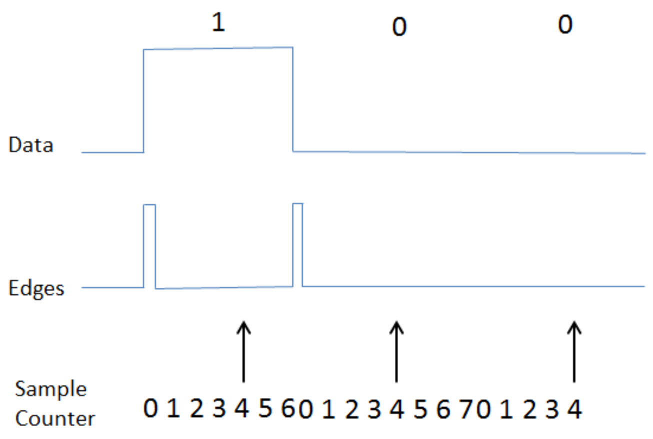

Figure 3-12: Example of the counter functionality

Then the output of Sampling Circuit is the clock signal for a D-flip flop, which is the last block, to regenerate the real content of the data as shown in the following figure [1].

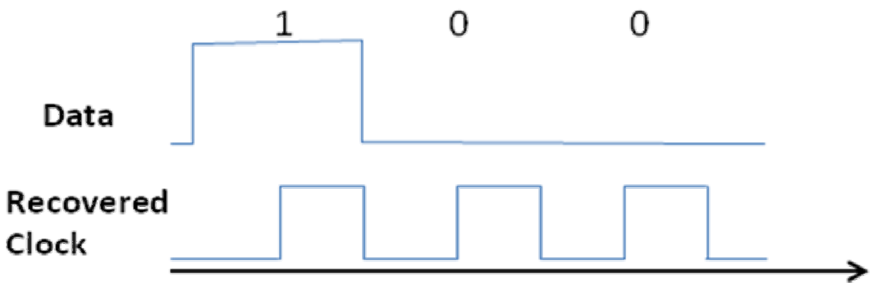

Figure 3-13: Example of a CRC output signal

4. Conclusion

The above document presents the interface modules analysis and characterization and shows that the best arrangement is OBI_Enc-SER-BI_Enc in terms of minimization of the number of '1' and transactions between two back-to-back data words, but it is the worst solution in terms of area occupation, latency and power consumption.

Then, a VHDL behavioural model of a Plasmonic Interconnect is described. Traditionally to recover the clock from a serial data stream uses Phase Locked Loop (PLL) [1]. But, in order to be implemented with FPGA, we proposed a purely Digital Method of Clock Recovery. In particular, it's the same used inside all Universal Asynchronous Receiver Transmitter (UARTs[2]) devices. The next step of the work will be to run simulations of these VHDL models.

[2] They traditionally have a Sampling Clock that is 16 Times the Data Rate [1].

References

[1] http://www.twyman.org.uk/clock_recovery/