D1.2.3 FPGA based error generator for PROFIBUS

Report on design and test results

PP3 Yncréa-ISEN team
PP2 KU Leuven team

1

Report outline:

2

1 – Objectives of this (sub)Activity

- To develop a board able to generate an error in a PROFIBUS system *(1) (2)* when a specific (user-configurable) event on the PROFIBUS RS485 physical layer is detected.
- This will be used for training (e.g. *(3)* for measurement tools) and also to replicate potential problems which may be observed on industrial systems. This is very useful for debugging, assessment of diagnostic tools, etc. (Refer to *(5)* and to *(6)* for an overview of errors in industrial field bus systems in OP ArcelorMittal Gent.) It is an addition to work on PROFINET load and error generators (D1.2.2&4), as many industrial systems incorporate a mix of these field bus systems.
- The core of the board is a FPGA for high speed performance and improved versatility. An older error generator – based on low-cost processor technology – proved to have a number of limitations *(4)*. Using FPGA *(7)* technology will remove these limitiations with regard to speed and flexibility, and represents a very good use case for Activity 1.2 of INCASE.
- The ISEN and KU Leuven teams joined forces in this work, combining experience and equipment in these domains. The full design was first done by simulation on recorded (real) PROFIBUS signals, before porting to the real board and the real networks. Please also refer to *(8)*.

3

2 – Design

1) The board must « listen and decode » incoming messages : this will be achieved by FPGA

   A PROFIBUS receiver operates as an **UART** (Universal Asynchrounous Receiver Transmitter), since PROFIBUS characters in a telegram are coded using this format.

   Coding format for PROFIBUS: 1 start bit, 8 data bit + EVEN parity bit, 1 stop bit (thus 1 character = 11 bits)

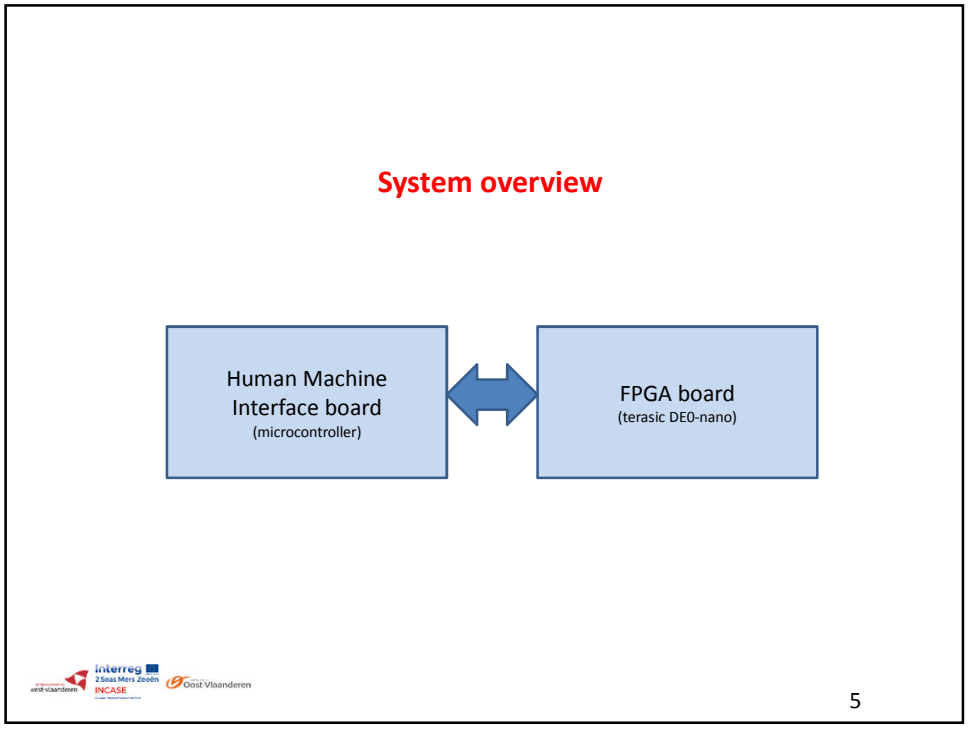
   ⇒ A UART receiver must be implemented in the FPGA

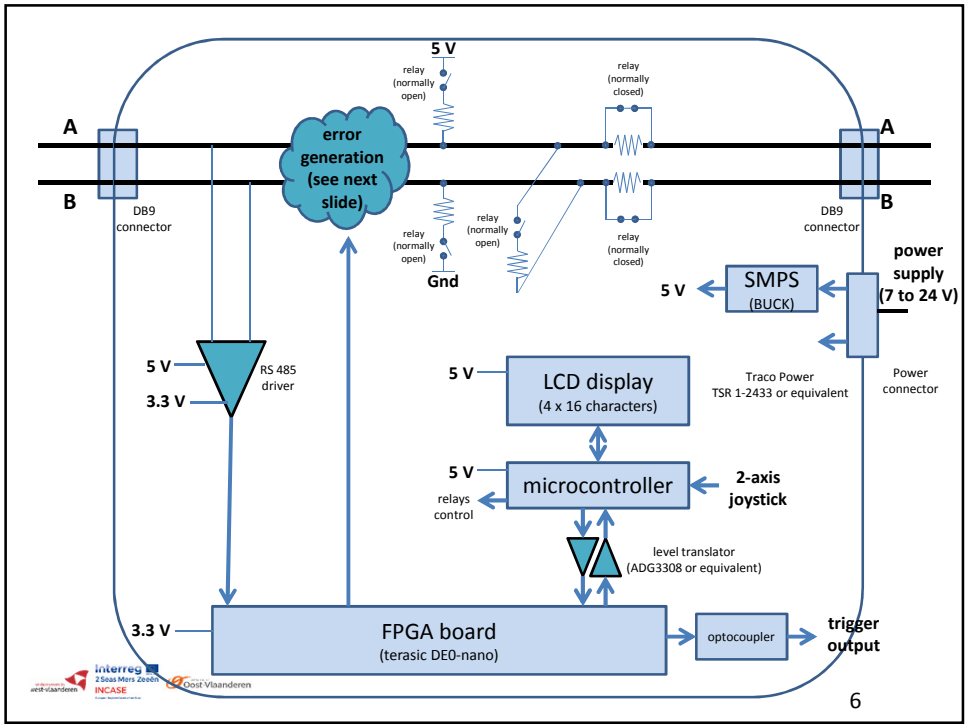
   The bit rate must be configurable (max. speed = **12 Mbps**)

2) The board must generate a error when a specific event is found

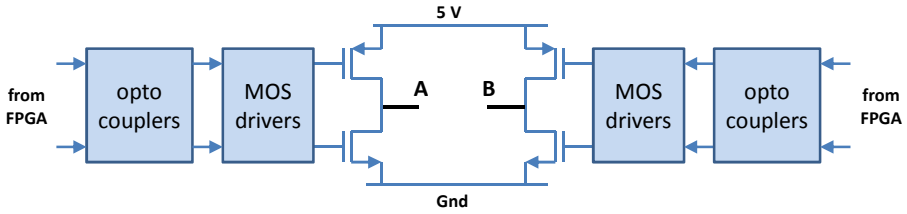
   The type a error will be user-selectable (short-circuit, high or low fixed state, etc.) : this will be achieved by a FPGA and microcontroller

4


System overview
Human Machine Interface board
(microcontroller)
FPGA board
(terasic DE0-nano)
5

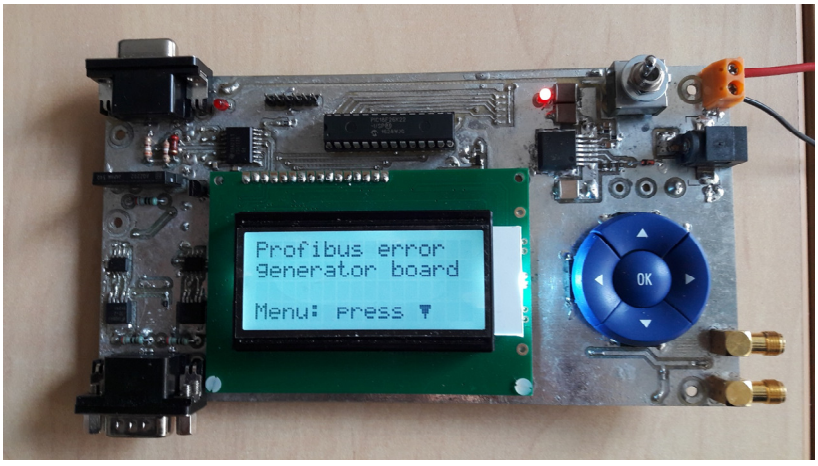

5 V
relay (normally open)
relay (normally closed)
A
B
error generation (see next slide)
DB9 connector
relay (normally open)
relay (normally open)
relay (normally closed)
Gnd
5 V
SMPS (BUCK)
power supply (7 to 24 V)
5 V
3.3 V
RS 485 driver
5 V
LCD display (4 x 16 characters)
Traco Power TSR 1-2433 or equivalent
Power connector
5 V
relays control
microcontroller
2-axis joystick
level translator (ADG3308 or equivalent)
3.3 V
FPGA board (terasic DE0-nano)
optocoupler
trigger output
6

Error generation, 1st design:



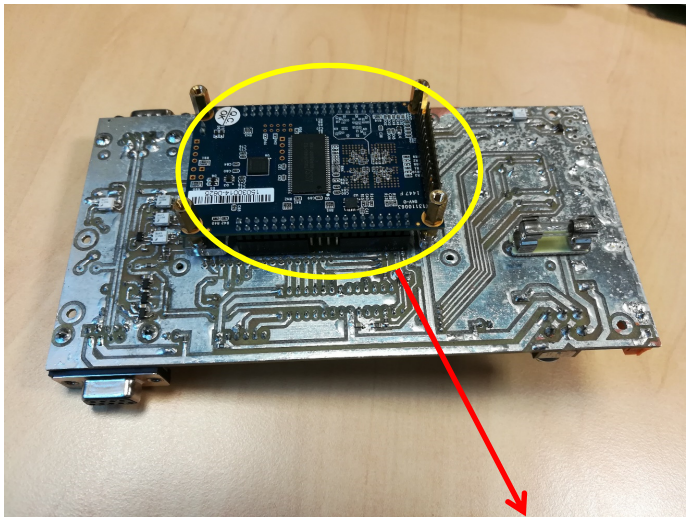
7

Picture of the board (top):



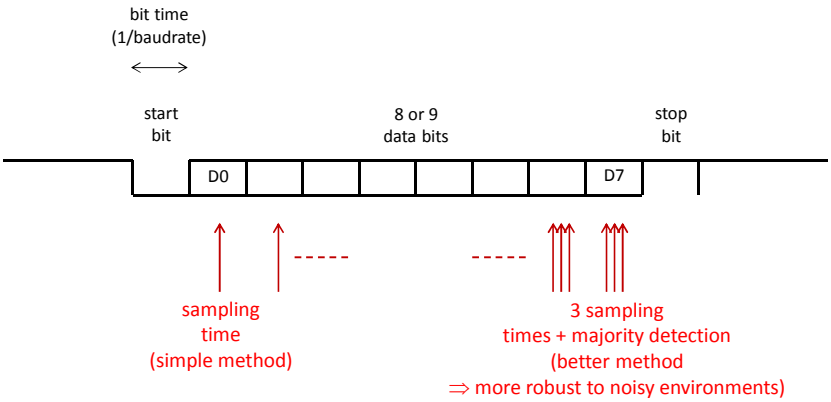
8

Picture of the board (bottom):



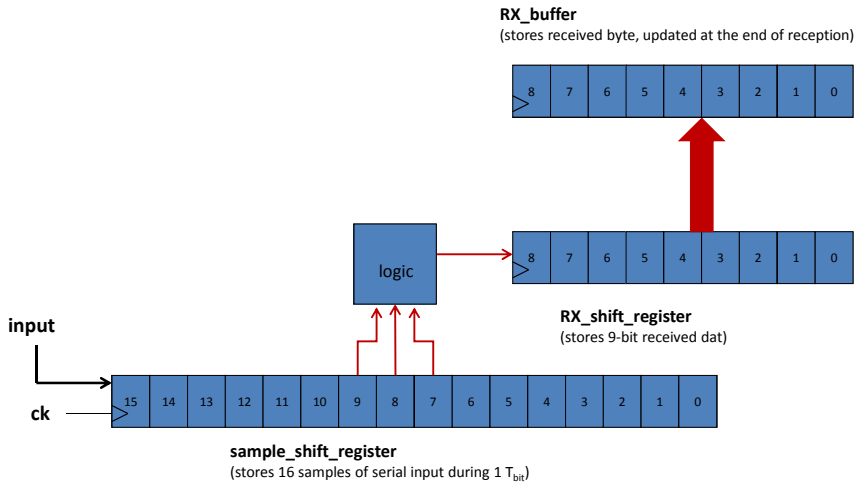
9

UART character format for PROFIBUS:



10

UART receiver structure in the FPGA



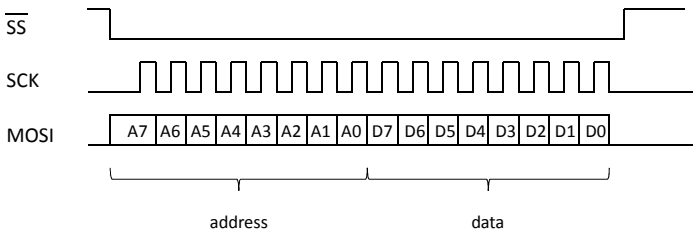
11

Interface between uC (microcontroller) and FPGA

Interface is used to load various parameters (i.e.: baudrate, trigger condition, Source or Destination Address, …)

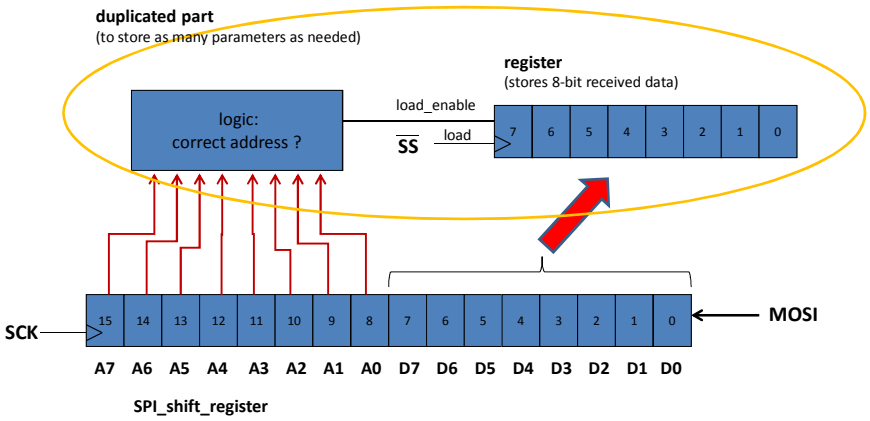
These parameters are stored in a bank of registers. (8-bit address, 8-bit data)

To load the registers, SPI interface is used (uC is master, FPGA is slave).



12

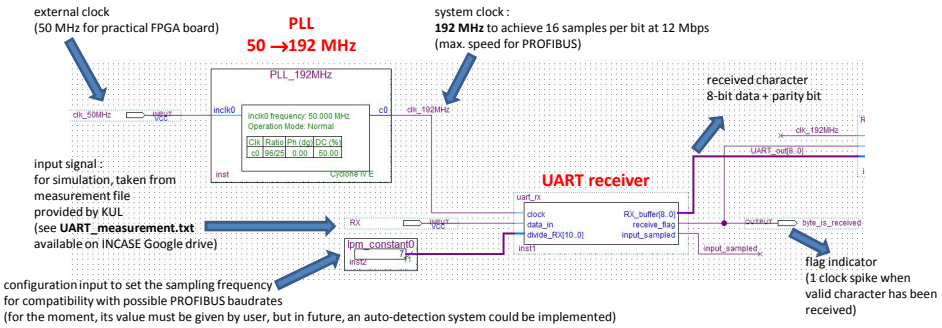
SPI peripheral has been designed and implemented in the FPGA:



13

3 – Simulation results

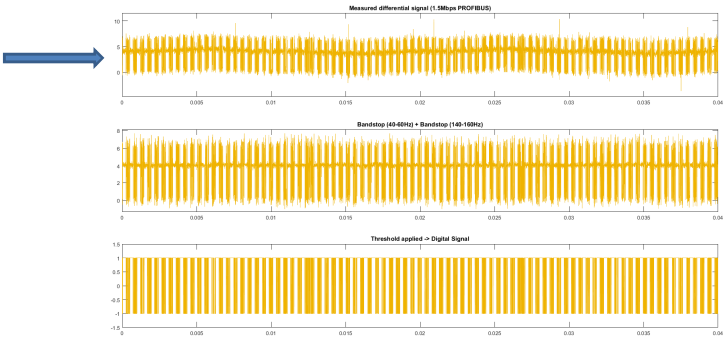
Operation has first been checked by simulation using real recorded PROFIBUS signals



14

To simulate the receiver (VHDL description file **uart_rx.vhd** available on INCASE Google Drive) being as close as possible to real conditions , input signal is extracted from a measurement file provided by KUL. This file is a set of measurement obtained with an oscilloscope monitoring a PROFIBUS signal.

input signal
for simulation

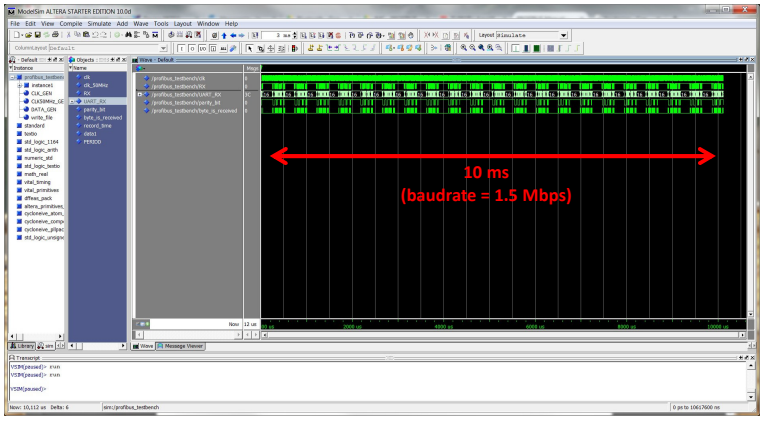


The receiver being simulated with a digital simulator (ModelSim), a testbench has been written to:
- read analog values from the measurement file,
- convert them to digital values,
- use them as simulation inputs.

The testbench (**testbench.vhd**) is avalable on INCASE Google Drive.

15

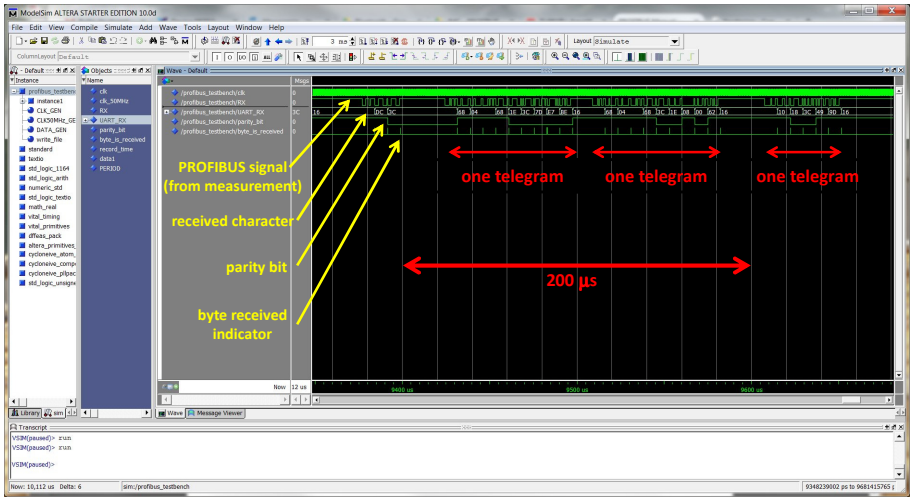
Simulation result to check that the FPGA can correctly decode characters in a PROFIBUS message



The simulation result shows that the UART receiver implemented in the FPGA can correctly decode the PROFIBUS signal.
Valid telegrams can be read (see detail next slide).

16

Simulation result (zoomed)



17

Next step: check that a specific event can be detected by the FPGA

List of useful triggers (defined according to project partners KUL + ISEN) *(3) (5) (8)* **:**

BASIC RAW TRIGGERS

Type	Remark	Detected at	Possible with Current Simulation Log File
SD (Start Delimiter / Message type)	SD1 SD2 SD3 SD4 or SC	First byte	YES
SA (Source Address)	(Not available in SC)	Depends on packet type	YES
DA (Destination Address)	(Not available in SC)	Depends on packet type	YES
Framing Error	No valid packet (wrong SD, LE, FCS, or ED)	At the end of a packet/error	NO
Parity Error	Parity of UART character incorrect	At every byte	NO
PDU# (Protocol Data Unit #)	(only in SD2, SD3)	Depends on packet type	YES
SAP (Service Access Point)	EXT (bit 7) of DA and SA should by high (only in SD2, SD3)	Depends on packet type	NO
No SAP	EXT (bit 7) of DA and SA should by low (Not available in SC)	Depends on packet type	YES
DSAP (Destination SAP)	If SAP -> Check Destination SAP (PDU1) (only in SD2, SD3)	Depends on packet type	NO
SSAP (Source SAP)	If SAP -> Check Source SAP (PDU2) (only in SD2, SD3)	Depends on packet type	NO
FC (Frame Counter)	Multiple sub triggers: Function code , Request/response, …	Depends on packet type	YES
LE (Length)	Only in SD2	Second byte (third byte)	YES

18

Telegram formats

We differentiate between the following telegram formats:
- Telegrams without data field
- Telegram with variable length of 4 to 249 bytes and therefore a payload in the range 1 to 246 bytes
- Telegram with fixed data length of 8 bytes data
- Token telegram
- For short, positive responses a short telegram has been defined.

Telegram without data field:

SD1	DA	SA	FC	FCS	ED

Telegram with variable length:

SD2	LE	LEr	SD2	DA	SA	FC	PDU	FCS	ED

The PDU has a variable length between 1 and 246 bytes.

Telegram with fixed data length:

SD3	DA	SA	FC	PDU	FCS	ED

The PDU has a fixed length of 8 bytes.

Token telegram:

SD4	DA	SA

Short confirmation:

SC

SC stands for *Short Confirmation*. SD means *Start Delimiter*, where these can assume four differe
telegram. The individual control fields have the following coding for UART:

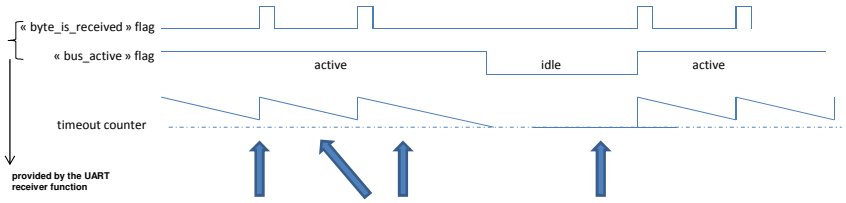
SD1	SD2	SD3	SD4	ED	SC
0x10	0x68	0xA2	0xDC	0x16	0xE5

19

- **Condition n°1: SD (Start Delimiter)**

⇒ to detect SD1, SD2, etc. as Start Delimiter and not as any other character inside the message, it is necessary to know if bus was active or not before arrival of the specific character.

Bus activity detection can be achieved with a timeout counter (or timer) which will be used to detect a « long » time between 2 received characters, which corresponds to a gap between 2 messages.
« Long » means something as 1.5, or 2, or 3 bit times.



byte_is_received = 1
⇒ timer = max value
(synchrounous with system clock)

timer ≠ 0
⇒ timer counts down

it is decremented at a
rate = 16 × bit rate,
thus 16 × 11 during one character
(there are 16 sampling clock periods during one bit,
and there are 1 start + 8 data + 1 parity + 1 stop bits for one character)

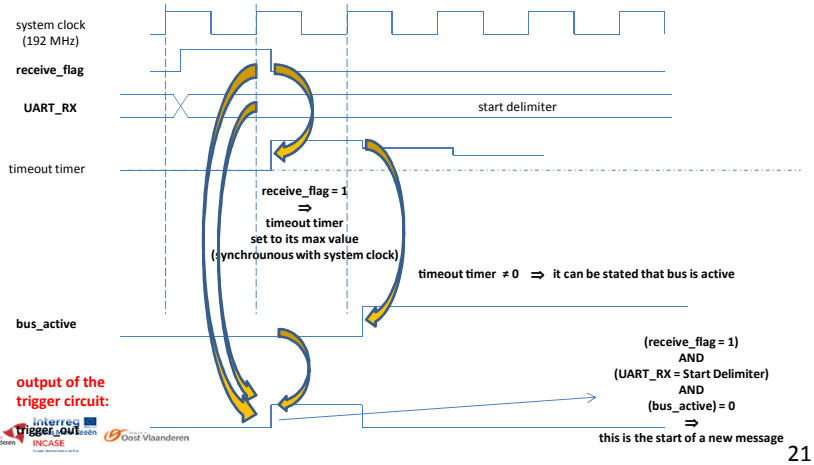
⇒ to set the timeout at 1.5 character time,
max value must be set to 1.5 × 16 × 11 = 264

timer = 0
⇒ bus state is declared as idle

20

Thanks to the « bus active » indicator explained before, trigger condition can be evaluated as follows:
- « receive_flag » signal must be high (it is one system clock spike showing that a valid character has just been received),
- « bus_active » signal must be low (because it will be set on the next system clock edge, but for the moment it is still low),
- « UART_RX » (the byte just received) must be one of the PROFIBUS Start Delimiters (SD1, SD2, SD3, SD4 or SC)

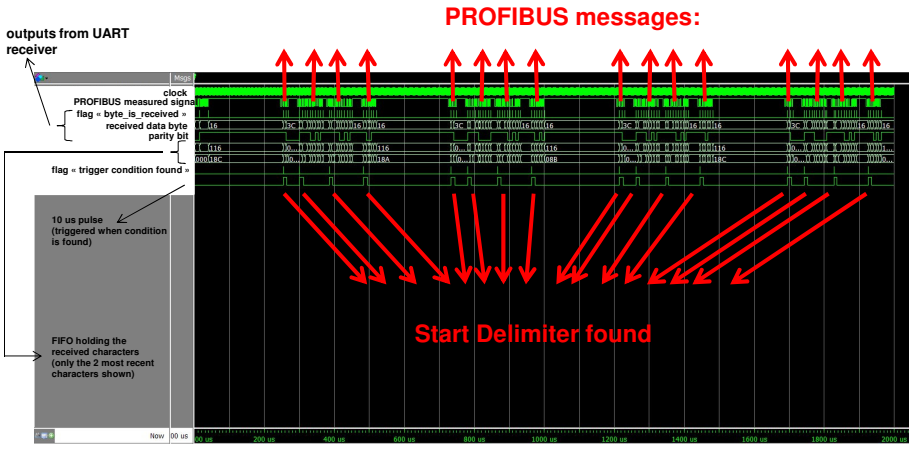
Notice: It was first planned to include in the condition an additional test, to check if the previous received character was an End Delimiter (ED = 0x16). But there is a special case: if the previous telegram is of type « Token Telegram» (its Start Delimiter is in this case SD4 = 0xDC), it doesn't end with ED. So this test has not been included.



21

Simulation of Start Delimiter Detection (1/3):

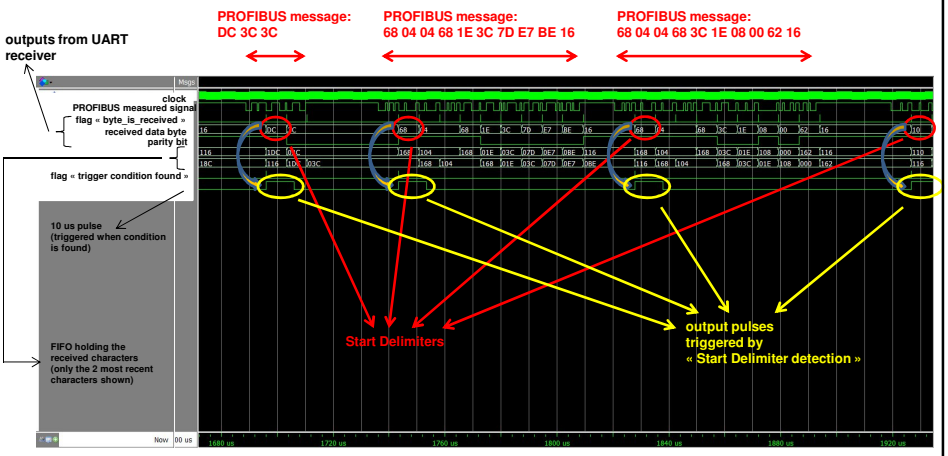
Simulation length = 2 ms: overview



22

Simulation of Start Delimiter Detection (2/3):

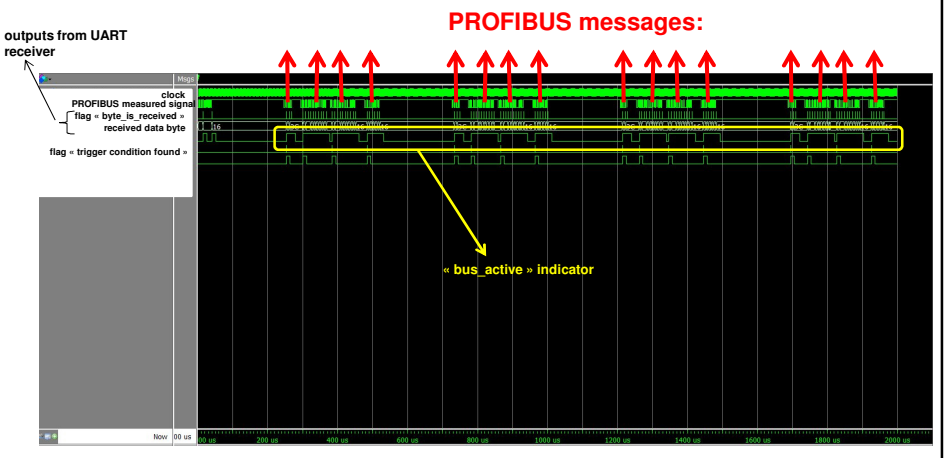
Zoomed view focusing on received messages



23

Simulation of Start Delimiter Detection (3/3):

Zoomed view showing the « bus_active » indicator which is used to detect the start of a new message

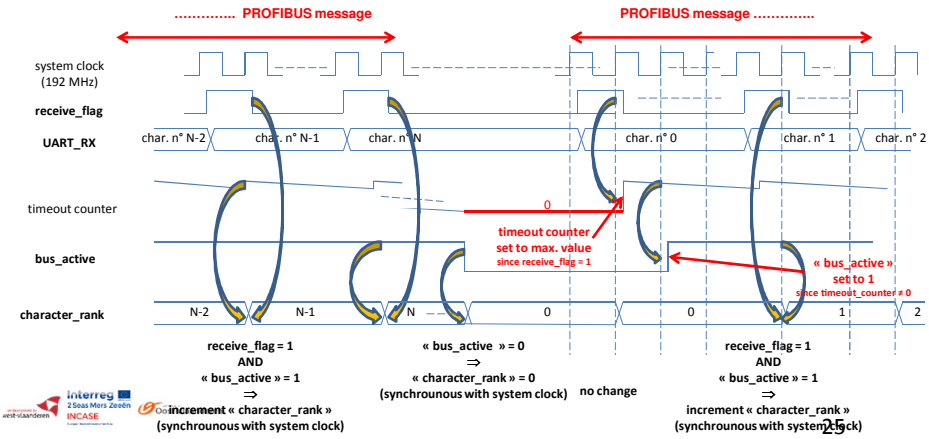


24

- **Condition n°2: SA(Source Address)**

SA is at a fixed position in the telegram (3rd byte).
⇒ in the Trigger_Condition block, a counter will be needed to store the position of the characters in a telegram.

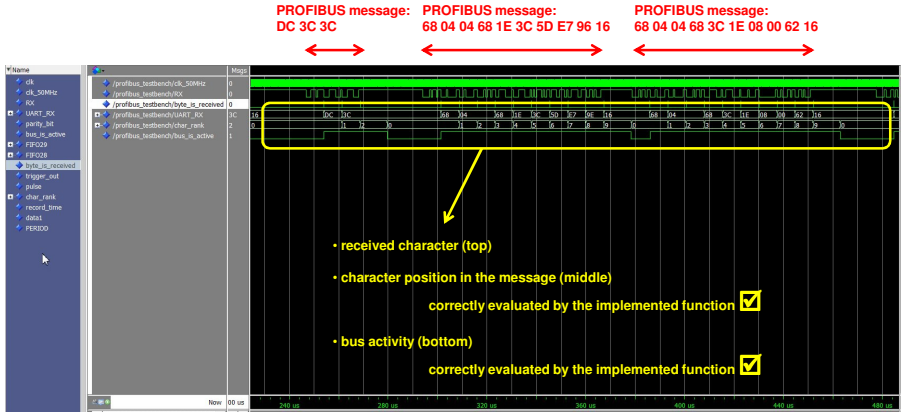
As a « bus_active » signal is available, it will be used to reset or start this counter (named « character_rank »).



25

Simulation of character position counting:

Zoomed view showing that the position of a character in a message is correctly evaluated

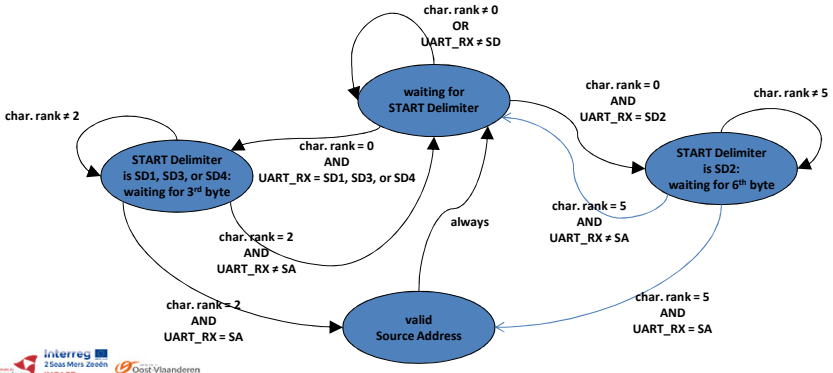


26

For the condition n°2 (source adress) to be considered TRUE, the following requirements must be fullfilled:

- When character rank is 0, the received character must be one of the valid Start Delimiters : SD1 (0x10) or SD2 (0x68) or SD3 (0xA2) or SD4 (0xDC).
- If Start Delimiter was SD1, SD3 or SD4, the 3rd byte in the message (position = 2) must be equal to the Source Adress of interest (8-bit wide).
- If Start Delimiter was SD2, the 6th byte in the message (position = 5) must be equal to the Source Adress of interest (8-bit wide).

The condition n°2 can thus be found with the following state machine (the advantage of this solution is that it minimizes the FPGA ressources which are needed: it is not necessary to record the entire message) :

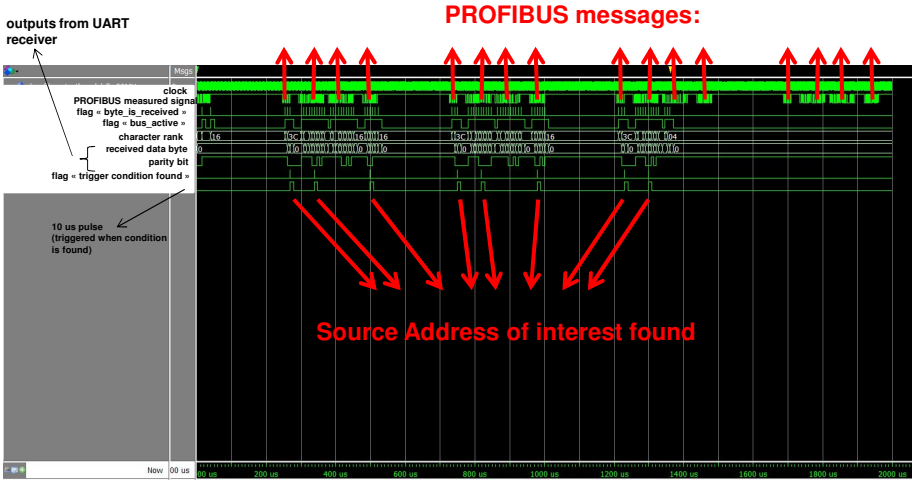


27

Simulation of Source Address Detection (1/3):

Source Address value checked for this simulation = 0x3C

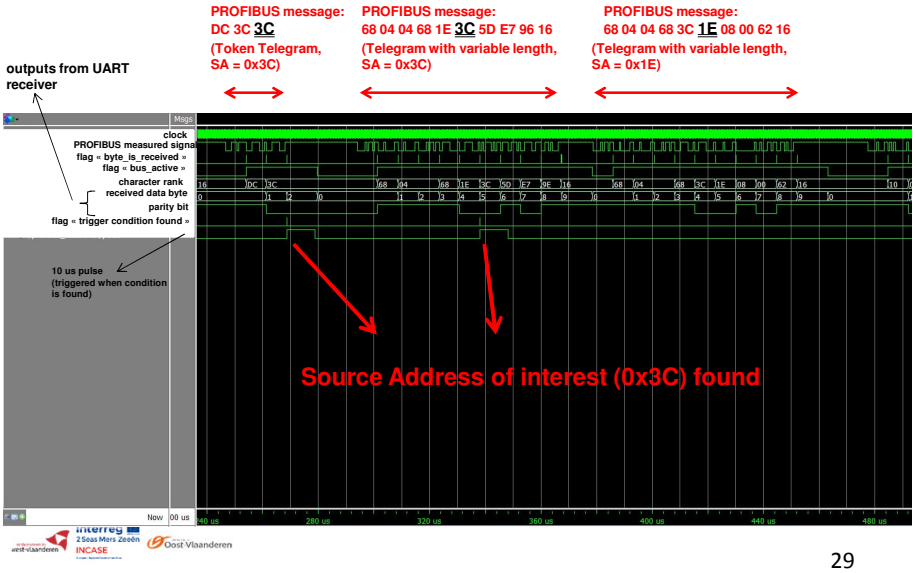
Simulation length = 2 ms: overview



28

Simulation of Source Address Detection (2/3):

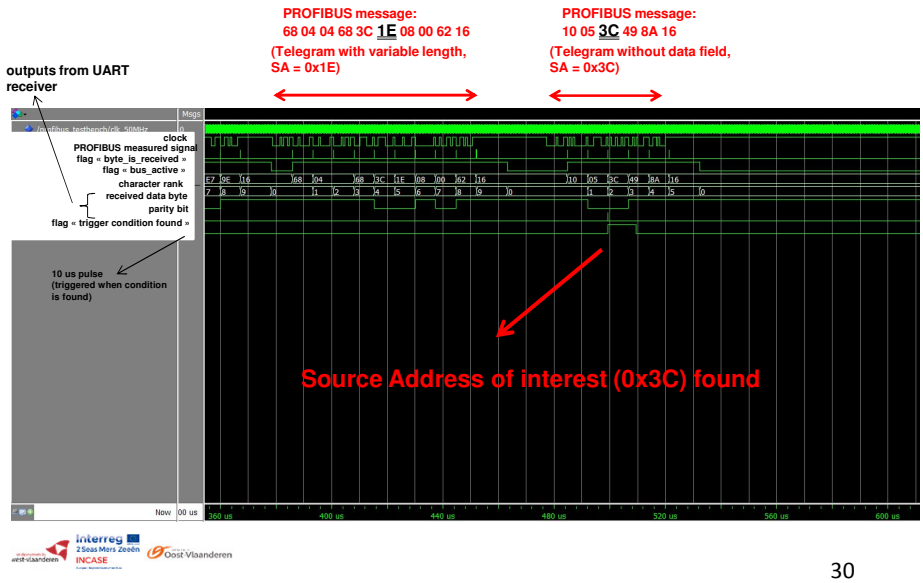
Source Address value checked for this simulation = 0x3C
zoomed view focusing on received messages



29

Simulation of Source Address Detection (3/3):

Source Address value checked for this simulation = 0x3C
zoomed view focusing on received messages



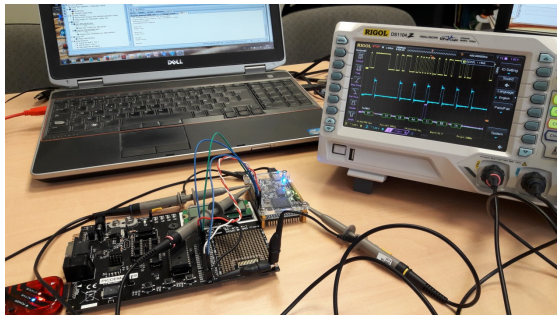
30

Practical test of Trigger Condition Detection (not on a real PB system):

Setup for the test:

PROFIBUS-like messages generated by a microcontroller (8-bit Microchip PIC18F @ 64 MHz)

Messages decoded and analyzed by an ALTERA FPGA (Cyclone IV E @ 192 MHz)



baudrate:

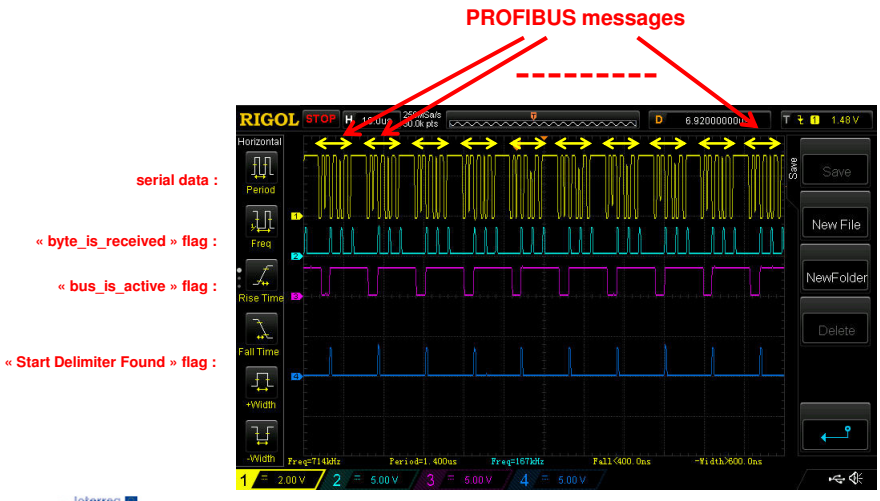
must be a submultiple of 16 MHz for the uC
must be a submiltiple of 12 MHz for the FPGA

⇒ baudrate set to 4 Mbps (highest common submultiple)

31

Practical test of Trigger Condition Detection (1/2):

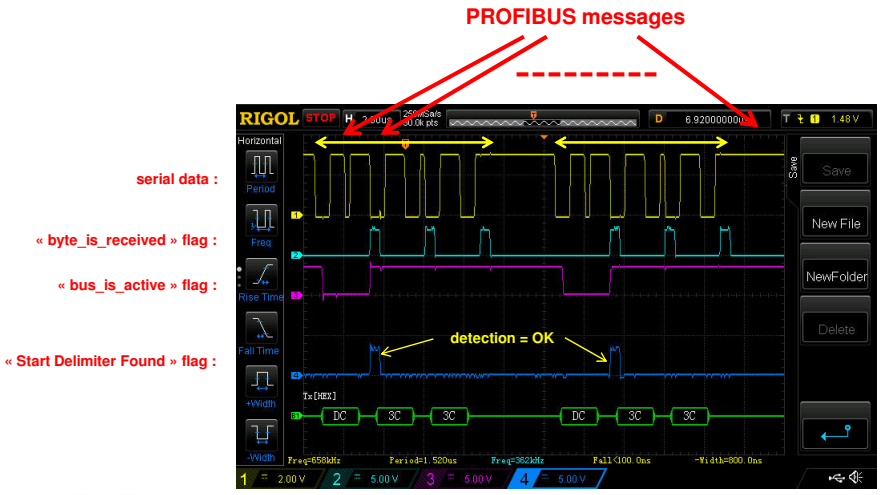
Trigger condition = Start Delimiter



32

Practical test of Trigger Condition Detection (1/2):

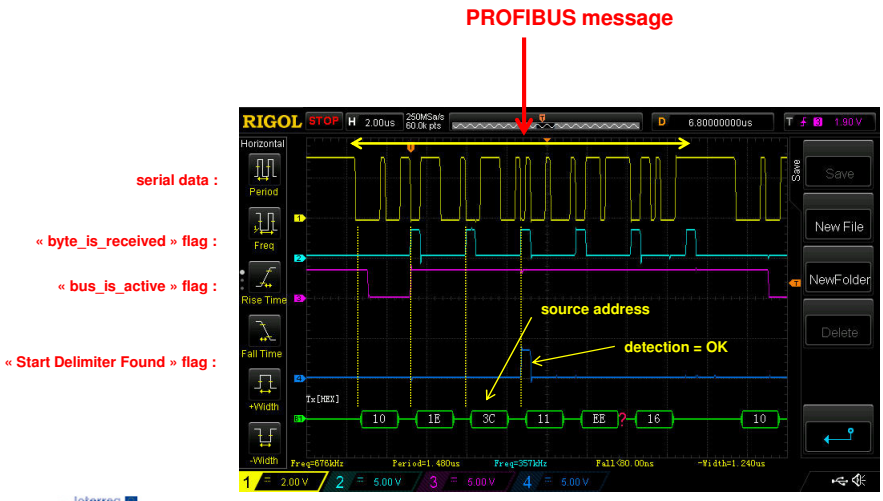
Trigger condition = Start Delimiter (zoom)



33

Practical test of Source Address Detection (1/1):

Trigger condition = Source Address = 0x3C (zoom)



34

4 – Practical tests results

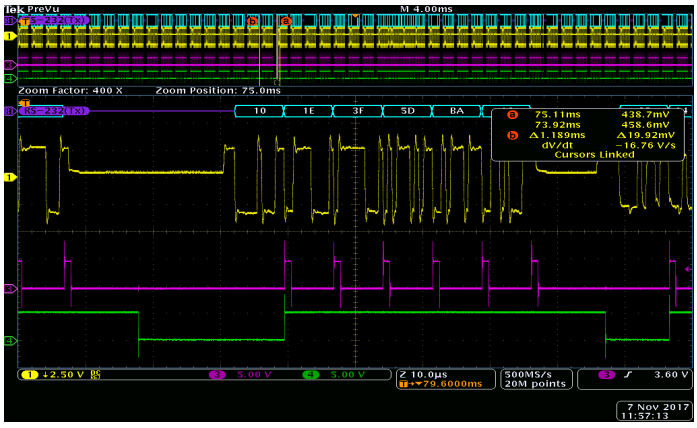
The board has been tested on a real system in Gent (KU Leuven laboratory) on **November 7th and December 7th 2017**

KUL: Philippe SAEY, Frederic DEPUYDT, Mathieu TROCH

ISEN: Jean-Marc CAPRON

35

- **Test n°1: check correct detection of characters and packets by the FPGA (bit rate = 1.5 Mbps)**



PROFIBUS message

PROFIBUS signal

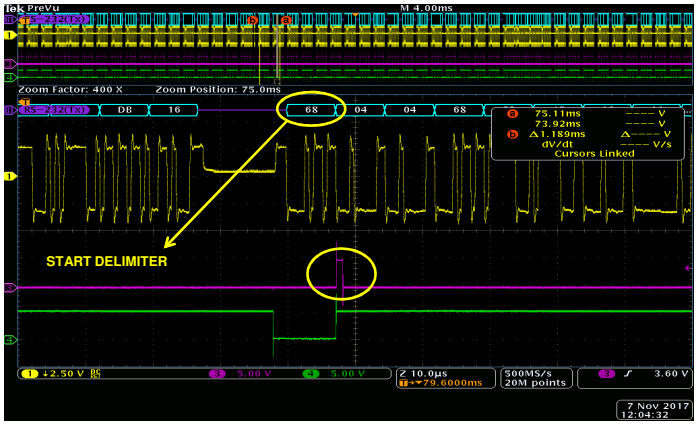
Board under test: output n°1 (one pulse when character is detected)

Board under test: output n°2 (high when message is on-going, comes back to 0 when no byte has been detected for 1.5 byte time)

Conclusion:
reading and decoding PB signals @ 1.5 Mbps is correctly achieved by the FPGA

36

- Test n°2: check correct detection of START DELIMITER event by the FPGA (bit rate = 1.5 Mbps)



PROFIBUS message

PROFIBUS signal

Board under test: output n°1 (one pulse when Start Delimiter is detected)

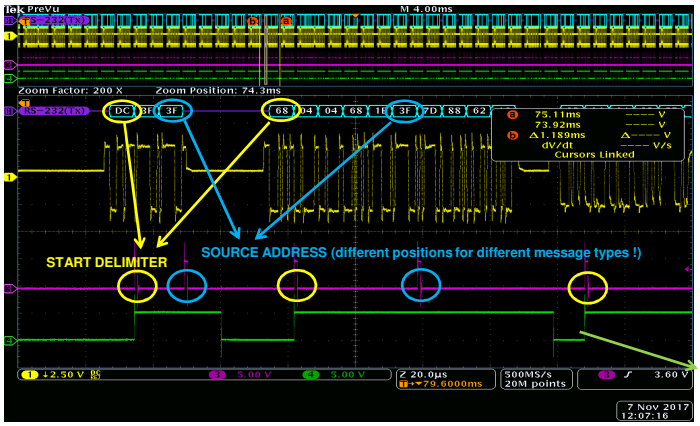
Board under test: output n°2 (high when message Is on-going)

Conclusion:

detecting the START DELIMITER @ 1.5 Mbps is correctly achieved by the FPGA

37

- Test n°3: check correct detection of a specific Source Address in a message (bit rate = 1.5 Mbps)
  - note: the Source Address position in the message depends on message type ⇒ FPGA must take this into account when searching for a specific address



PROFIBUS message

PROFIBUS signal

Board under test: output n°1 (one pulse when Start Delimiter is detected OR a given Source Address is Found – here: 3F)

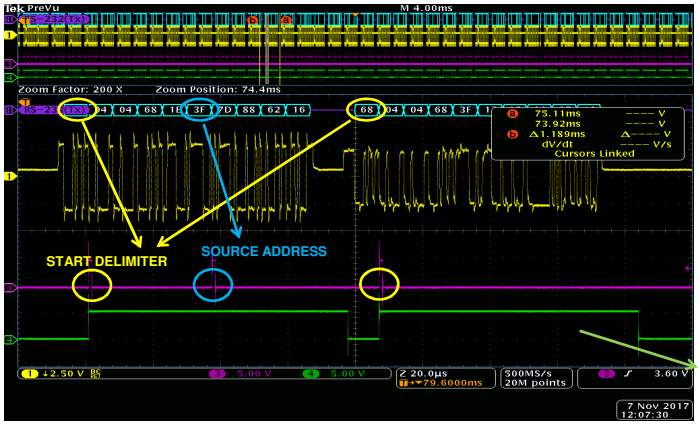
Board under test: output n°2 (high when message Is on-going)

Conclusion:

detecting a specific SOURCE ADDRESS in a message @ 1.5 Mbps is correctly achieved by the FPGA, whatever the type of message

38

- Test n°3 (continued): check correct detection of a specific Source Address in a message (bit rate = 1.5 Mbps)



PROFIBUS message

PROFIBUS signal

Board under test: output n°1 (one pulse when Start Delimiter is detected OR a given Source Address is Found – here: 3F)

Board under test: output n°2 (high when message is on-going)

Conclusion:

detecting a specific SOURCE ADDRESS in a message @ 1.5 Mbps is correctly achieved by the FPGA, whatever the type of message

39

- Test n°4 : check that error is applied on the bus when a user-defined event has been found (bit rate = 1.5 Mbps)

Event detection being correctly achieved, this information was used to trigger a error on the bus.

A error is a fixed voltage applied on PB lines (A & B), in order to « disturb » the message.

error type is user-selectable: it can be chosen through Human-Machine-Interface to clamp A & B to low or high states (any combination is feasable)

START DELIMITER

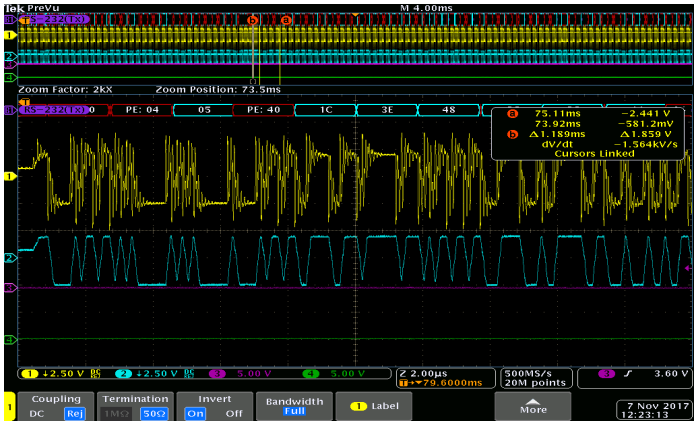
PROFIBUS signals clamped to fixed voltage

Conclusion:

applying a error on the bus @ 1.5 Mbps is correctly achieved by the board

40

- Test n°5: check correct detection of characters and packets by the FPGA (bit rate = 12 Mbps)



PROFIBUS message

PROFIBUS signal @ 12 Mbps

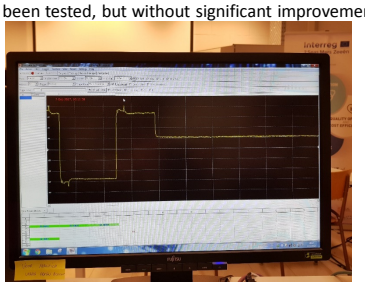
Filtered version of the signal

Conclusion:
at maximum bit rate (12 Mbps) operation is disturbed by the board connection

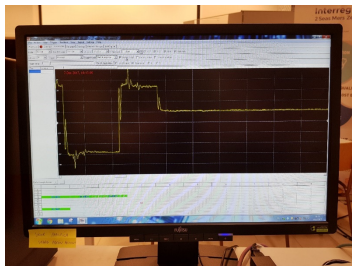
41

At maximum speed (12 Mbps), connecting the board to the bus disturbes the signal, which is not a desired feature.

Adding ferrites to the board for better filtering of oscillations and dissipation of the associated energy has been tested, but without significant improvement.



Board not connected to the bus (1.5 Mbps)



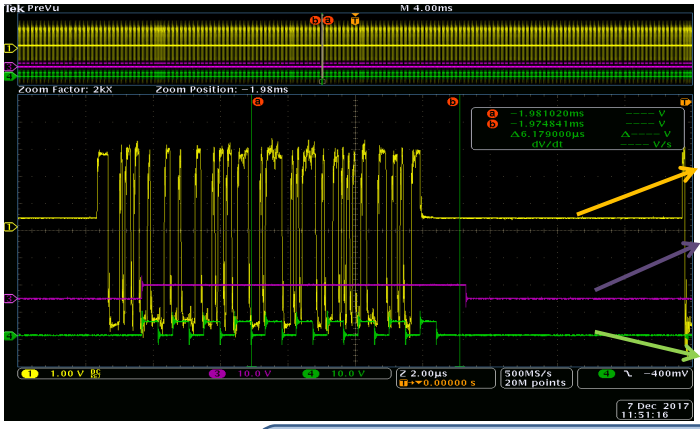
Board connected to the bus (1.5 Mbps)

However, detection of characters is still correctly achieved by the FPGA at 12 Mbps (see oscilloscope snapshot on the next slide). But correct operation of the original system could however be disturbed, in the case of longer cables, larger networks, EMI, etc. .

Conclusion:
Tests are currently on-going to find a better way to apply a error to the bus

42

However, detection of characters is still correctly achieved by the FPGA at 12 Mbps, even if correct operation of the original system could be disturbed.



Conclusion:

reading and decoding PB signals @ 12 Mbps is correctly achieved by the FPGA, even if the hardware for error generation must be redesigned for a better impedance match.

43

5 – Results summary

	DONE	IMPROVEMENT PLANNED
FPGA implementation of a PROFIBUS receiver (**VHDL** development)	✓	
FPGA implementation of PROFIBUS message decoding + event detection (**VHDL** development)	✓	
Simulation with testbench from real PB messages record (**MODELSIM**)	✓	
Software development of Human-Machine-Interface (**C-code** development)	✓	
Board design and fabrication of the error generator	✓	
Practical test @ 1.5 MBps	✓	
Practical test @ 12 MBps	✓	

The 2nd version of the board is planned to bring less disturbance to PROFIBUS DP at full speed.
Further work is under D1.2.5, and possibly under WPC for a paper.

44

6 – References

(1) “Decentralization with PROFIBUS-DP”; J. Weigmann, G. Kilian, Publicis MCD Verlag, Erlangen, Duitsland, 2000.
(2) “The New Rapid Way to PROFIBUS-DP”; M. Popp, PROFIBUS Nutzorganisation, Karlruhe, Duitsland, 2003
(3) https://procentec.com/products/profitrace/?content-1 & https://procentec.com/products/combricks/?content-1
(4) “Design of an Arduino based low-cost error generator for PROFIBUS DP”; IEEE Conference on Emerging Technologies and Factory Automation ETFA 2014, Sep. 16-19, Barcelona (Spain). Philippe Saey, Ward Hauspie, Hendrik Derre, Thomas De Landtsheer, Annemarie Kokosy, Jos Knockaert.
(5) “PROFIBUS: Theory & Practice, Engineering & Troubleshooting”; F. Depuydt, W. Hauspie, H. Derre, T. De Landtsheer, S. Noppe, M. Troch, P. Saey.
(6) PROFIBUS troubleshooting a.d.h.v permanente logging met COMbricks in ArcelorMittal Gent. Masterproef Jens Mortier, KU Leuven.
(7) Methodology for FPGA-based system design. Jean-Marc Capron, ISEN-Lille (TTM day INCASE, 17/02/2017)
(8) INCASE meeting reports, Technical Days, Internal Conference.