Standalone low-cost versatile FPGA-based fault generator for PROFIBUS DP

Jean-Marc Capron, Mathieu Troch, Dimitri de Schuyter, Arne Verhoeven, Jos Knockaert, Philippe Saey

Standalone low-cost versatile FPGA-based fault generator for PROFIBUS DP

Capron Jean-Marc
Junia – ISEN & CNRS UMR 8520 - IEMN
Lille, France
jean-marc.capron@junia.com

Troch Mathieu
KU Leuven – FET – ESAT-ELECTA
Energy & Automation
Gent, Belgium
mathieu.troch@kuleuven.be

De Schuyter Dimitri
KU Leuven – FET – ESAT-ELECTA
Energy & Automation
Gent, Belgium
dimitri.deschuyter@kuleuven.be

Verhoeven Arne
KU Leuven – FET – ESAT-ELECTA
Energy & Automation
Gent, Belgium
arne.verhoeven@kuleuven.be

Knockaert Jos
Ghent University, Dep. Industrial System and Product Design
Kortrijk, Belgium
jos.knockaert@ugent.be

Saey Philippe
KU Leuven – FET – ESAT-ELECTA
Energy & Automation
Gent, Belgium
philippe.saey@kuleuven.be

Abstract **This paper presents the design, the validation steps and the measurement results of a fault generator device able to disturb the operation of a PROFIBUS DP network, under user-defined conditions. It allows to test the robustness of an industrial network, to test and compare diagnostic tools, and to investigate off-line more complex faults encountered in industry. The core of the design is an FPGA, which results in a very low latency time. It generates trigger signals and imposes faults on the RS485 level up to the maximum bit rate of 12 Mbps. The design choices for fault duration, number of successive faults and skipped triggers, waiting time, etc. allow for the emulation of a wide range of network faults.**

Keywords **Industrial network – Diagnostics – PROFIBUS DP – Data communication – Network-based control – Robustness**

I. Introduction

Training and more advanced teaching of PROFIBUS DP network technology [1] [2] requires in-depth hands-on exercises, with a variety of realistic and challenging network faults. For training relevant for industry, a lot of attention should be on diagnostics and on troubleshooting network faults [3]. Also for research, for example to compare the diagnosis reported by different diagnostic tools, to reproduce more complex faults from industry cases during later off-line analysis with similar components, for stress testing of devices [4] etc., a performant fault generator device would be very useful.

Typical "classic" faults such as short circuit, open circuit, impedance mismatches, power failure of or removing the connector of a slave, removing a module from a slave, etc. are easy to simulate during lab exercises and tests. However, these faults are – relative to the network cycle time – of very long duration and cannot produce short faults impacting only 1 or 2 telegrams, let alone on the same one – originating from the same faulty network device – in consecutive PROFIBUS DP cycles.

Nevertheless, many researchers apply these long "uncontrolled" short and open circuits and impedance mismatches using simple switches [5] [6] [7]. Also the commercial Procentec fault generator [8] uses this approach. The "DP training device" [9] is more advanced, but produces only a limited amount of fault types.

Some researchers [10] [11] use software implementations of PROFIBUS masters and slaves on microcontrollers (and not industrial devices) at lower bit rates (9.6 up to 500 kbps) than the typical 1.5 Mbps or higher used in industry, for analysis and fault implementation. This method is however flexible compared to the ones mentioned in the previous paragraph and serves its specific research goals.

From a practical view point, removing connectors or modules, switching terminating resistors on/off, … all put a heavy wear on the equipment and should be avoided as much as possible for fault simulation.

Therefore, a low-cost, versatile, standalone, compact device that 1) produces trigger signals 2) imposes a fault (destroy bits, bytes and telegrams, add e.g. an impedance mismatch) has been developed. The trigger events can be used by an oscilloscope to record other signals (shield current, motor torque, etc.) or signals at multiple different network locations. The faults are either imposed by RS485 drivers (specific characters and telegrams, highly dynamic) or by reed relays (e.g. add impedance or inject current in the shield, long duration).

The core of the presented design is an FPGA, that introduces only a very short latency time to detect the trigger conditions, which can be combinations of multiple telegram properties. These trigger conditions can be configured with high flexibility using a simple Human Machine Interface (HMI), and must be editable on-the-go, with instant effect. Finally, the timing and sequence of the output activity (trigger output, fault output) is also configurable: duration, skipped triggers, delay, number of repeats.

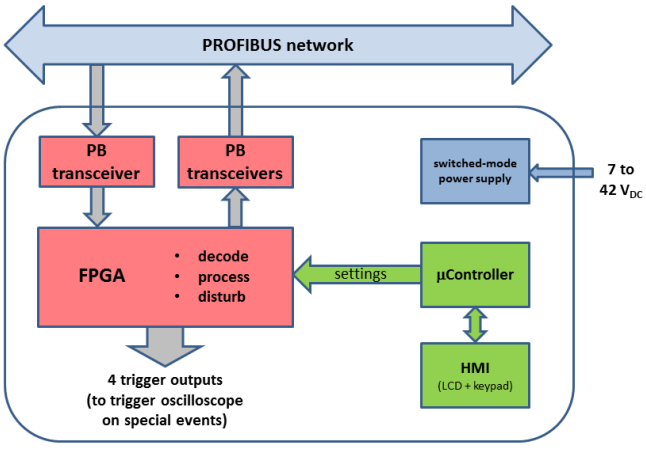

Fig. 1. Main components of the PROFIBUS DP fault generator.

This paper describes the development and some use cases of a low-cost, compact, versatile and standalone PROFIBUS DP (PB DP) fault generator based on FPGAs. It addresses all standard bit rates from 9.6 kbps up to 12 Mbps, and removes a number of limitations of an older microprocessor based implementation by the authors [12]. The FPGA in Fig. 1 is

used for the processing of the incoming information, to decode on-going messages, search for a specific condition and generate output signals (bus disturbance and external digital outputs). This enables real-time analysis and quasi instantaneous reaction time. The configuration of trigger conditions and output characteristics can be modified on-the-fly with an LCD display with keyboard switches as HMI, managed by a microcontroller.

The remainder of this paper is organized as follows: PROFIBUS DP is briefly described and design specifications are inferred in Section 2. Section 3 presents the fault generator design: use of FPGAs, serial receiver, trigger analysis, actual fault generation, duration and sequence, etc. Section 4 discusses the validation by simulation. Validation in live networks is discussed in Section 5, which covers the base-line healthy network, the fault appearance over the length of the bus line, 2 use cases, and the overall reaction time.

II. PROFIBUS DP – Design Specifications

A. *Network overview and UART character*

Detailed descriptions of PROFIBUS DP (PB DP) can be found in e.g. [2], [3] and [13]; the website of PI International can be used for a first exploration [1]. In these paragraphs, some elements that are of interest for the fault generator design are described briefly.

PB DP is most often implemented using RS485 for the Physical Layer (PHY). Fig. 2 shows the typical layout: it features half duplex communication over two lines designated A and B, is actively terminated on both ends of the network (segment). There are up to a typical maximum of 8 segments with each maximum 32 stations in a network. Fig. 3 shows the non-return-to-zero UART (Universal Asynchronous Receiver Transmitter) character. It has one start and stop bit, 8 data bits and a (even) parity bit. Bit rates vary from 9600 bps up to 12 Mbps; typically 1.5 Mbps or higher is used in industry.

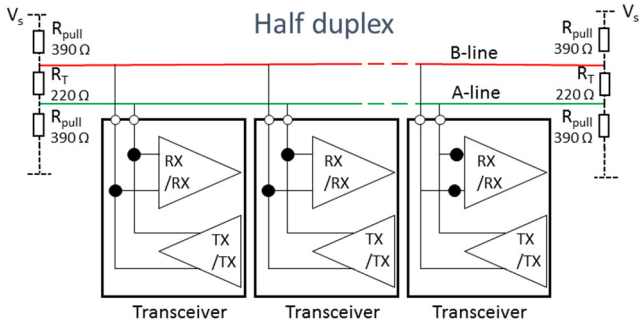

Fig. 2. Layout of the PROFIBUS DP RS485 Physical Layer.

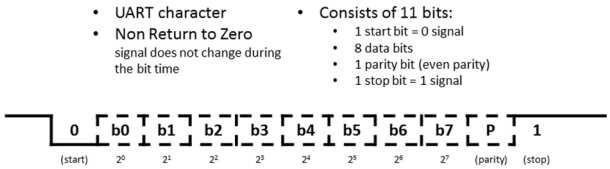

Fig. 3. UART character of PB DP: non-return-to-zero, 8 databits, 1 start and 1 stop bit, even parity.

B. *Telegram formats – Networks used for testing*

Fig. 4 shows as example the standard data exchange telegram SD2 (top). The meaning of the telegram parts and the different PB telegram types – SD types – are explained in the middle and bottom table. The tables are based on [13], and form the basis of all state machines explained in Section III.

SD2	LE	LEr	SD2	DA	SA	FC	DU	FCS	ED
68_H	·	·	68_H	·	·	·	·····	·	16_H

Byte	Explanation
SD	Start delimiter (value depends on telegram format)
LE	Length (min. 4, max. 249), LE = DA + SA + FC + DU + DSAP + SSAP
LEr	Length (redundant), for data protection
DA	Destination address (defines for which station the telegram is meant)
SA	Source address, defines from which station the telegram comes
FC	Frame control (telegram type, station type, priority & telegram ack)
DSAP	Destination service access point
SSAP	Source service access point
DU	Data unit
FCS	Data protection frame check sequence checksum
ED	End delimiter (0x16)

SD1	Without data: FDL status requests	Check if the device is still online, or to ask data from device with no outputs
SD2	Variable data length for data exchange	Most used message type, used for data exchange
SD3	Fixed data length (8 bytes)	Not (much) used in PROFIBUS
SD4	Token message	Short message so influence on cycle time is minimal
SC	Acknowledge message	Confirmation that message is correctly received

Fig. 4. Standard data exchange telegram SD2 (top), meaning of the telegram bytes (middle) and different PB DP telegram types.

For measurements, two basic networks are used in this study: a single master (DPM1, DP Master Class 1) network with 3 slaves (Fig. 5, top), and a network including 3 DPM1 combined with 3 slaves, with token passing between the masters (Fig. 5, bottom). The station addresses are indicated in the bottom right corner.

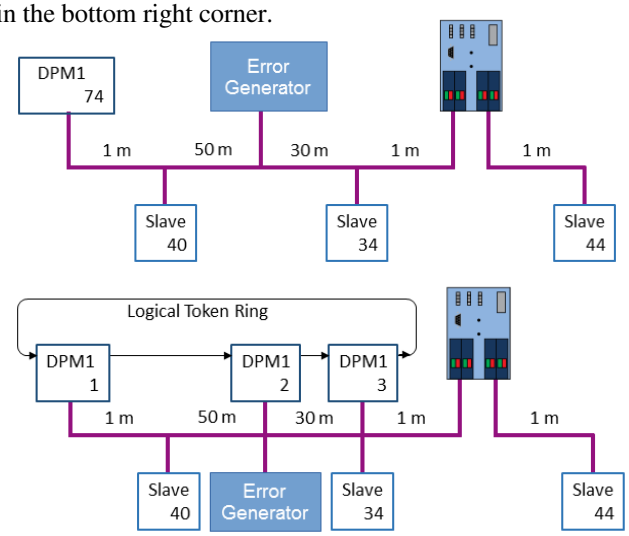

Fig. 5. Networks used for verification of the fault generator.

Fig. 6 shows the healthy single master network with measuring equipment as in Fig. 15. The bottom figure provides a detailed view at the beginning of the middle telegram, for 4 locations on the cable. Given the cable length and the extra connectors – acting as small extra impedances – the bit signals are slightly degraded, but fine and at good voltage levels. Channel 4 – after the diagnostic repeater – shows "restored" signals with extra delay caused by the diagnostic repeater. One can observe the run time of the signal over the cable.

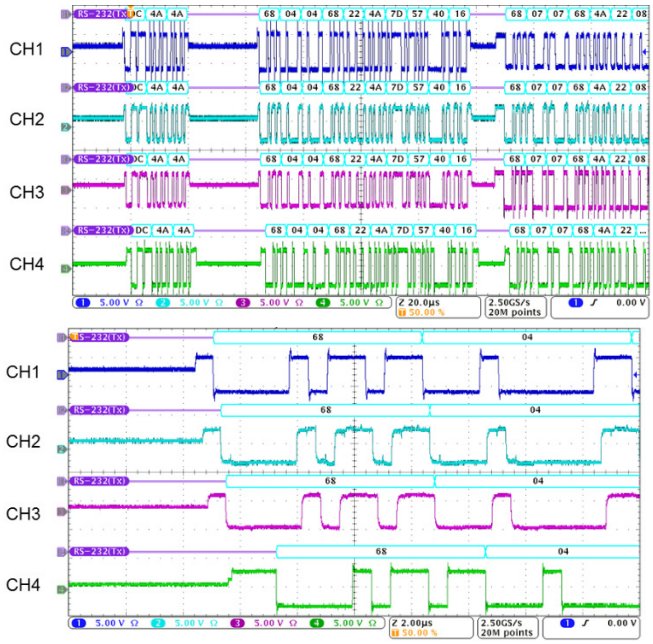

Fig. 6. Healthy network signals at 1.5 Mbps.

C. *Design specifications and choices*

In general, the fault generator should be compact and operate standalone. It is designed for all PROFIBUS DP bit rates up to 12 Mbps. Two ways to inject faults are implemented: destruction of a number of bits or characters via RS485 drivers, and switch e.g. impedances, current sources, etc. via reed relays. The reed relays are in this version – given their relatively slow timing compared to PB DP telegrams and cycle times – controlled manually via the display. The focus in this paper is on fault injection via RS485 drivers and more complex network fault situations, including multiple trigger and sequencing options.

The reaction time should be short compared to the character length, as the fault generator waits to inject a fault until it has checked that the last character that fulfils the trigger condition(s) is actually a valid character. The shortest character length at 12 Mbps is 11 x 83 ns = 917 ns; as reminder, telegrams are composed of several characters.

The trigger conditions are (the content of) each of the bytes in Fig. 4 (top and middle), and can be combined (AND and OR).

The fault duration itself can be expressed in bit times or character (11 bit) times, and ranges from 1 to 999 resp. 1 to 510. The waiting time between applied faults can be up to 17 minutes (units in ms or s (both 1-999) or minutes (1-17)). Skipped triggers can vary between 0 and 9999; the number of successive faults is between 1 and 31. Also these properties can be combined.

D. *Coverage of different fault types*

Referring to Fig. 7 [14] that provides one of the few detailed overviews of typical faults in industrial networks, the bit destruction part of the fault generator typically emulates "software & devices" (9 %), "other" (20 %) and "EMC" (17 %). The "connectors" part – if intermittent – can also be covered.

"Excessive cable length" and "cable" problems are typically permanently present and not really relevant for the bit destruction part. Some of these could be applied with the reed relays by adding impedances: wrong cable type, too many terminating networks, high transition resistance, etc. [5].

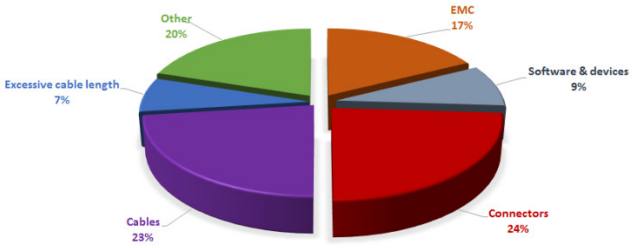

Fig. 7. Overview of typical network interventions of Indu-Sol. The figure combines the data provided in the 2019 and 2020 Vortex reports.

An overview of the typical properties of EMI (ElectroMagnetic Interference) phenomena that are at the basis of EMC problems, and the coverage by the developed fault generator, can be found in Fig. 8. With the fault generator, we emulate the effect of the EMI phenomena on the bus communication or on the temporary malfunctioning of a network device (and consequently the "silence" on the bus), by disrupting all communication of the device(s) for the appropriate duration and periodicity.

Phenomena (related standard)	Practical occurrence	Typical rise time (wide band) or frequency (narrow band)	Typical duration	Fault generator: bit destruction or via reed relays	Fault generator: timing
Electrostatic discharge (ESD) (EN61000-4-2)	Operator touching metallic part, isolated transmissions	0.8 ns	60 ns	Bit destruction	1 bit at 12 Mbps = 83 ns. Probably not relevant at bitrates lower than 1.5 Mbps (that already has a bit width of 667 ns) if the field impacts on the bus communication only and during the ESD.
Fast transient (EN61000-4-4)	Switching inductive loads, relay contact bounce	5 ns	50 ns pulse, occuring in 15 ms bursts	Bit destruction	At 1.5 Mbps up to 3.75 ms. At 12 Mbps up to 0.47 ms. To combine with skipped faults & successive faults, and/or with waiting time for longer disturbances.
Surge (EN61000-4-5)	Switching and lightning transient	1.2 µs	50 µs	Bit destruction	50 µs @ 12 Mbps is 602 bit times or 55 character times
Continuous wave (EN61000-4-3 & 6)	Radio wave, transmitters	150 kHz – 80 (230) MHz 80 MHz- 6 GHz	Continuous	Intermittent (but relative-ly long duration) effects: bit destruction. More permanent: reed relay.	/
Power magnetic field (EN61000-4-8)	Current carrying conductors and transformers	50 Hz	Continuous	Intermittent (but relative-ly long duration) effects: bit destruction. More permanent: reed relay.	/

Fig. 8. Overview of EMI phenomena, indicating typical duration and coverage by the fault generator.

Some industrial examples and their link to EMI phenomena are briefly presented in this paragraph. The combination of PWM (Pulse Width Modulation) used in the output stage of a motor inverter and reflections on the motor cable gives a damped oscillation typically in the range of 500 kHz up to 10 MHz, and relates to the "continuous wave" phenomenon. The PWM waveform itself has a rise time of some tens of ns and can be related to the burst test. Arc welding generates strong magnetic fields in the same lower MHz frequency range; spot welding typically generates low frequency magnetic fields (Hz up to kHz) that are of relatively short duration. Touching metallic parts (a button, a connector) can generate electrostatic discharges; a belt transmission is an example of an "isolated transmission" causing ESD.

In conclusion, the design choices of this fault generator make it well suited for the 48 % "interesting" (short, non-deterministic, complex, single event, etc.) faults, and can be used for a number of other ones, making it a versatile design. Also refer to [3] and [15] for a number of industrial fault use cases. (Remark: both RS485 (PROFIBUS) and Ethernet (PROFINET) based networks are in the Indu-Sol statistics. We assume a similar relative spreading, so percentages are order of magnitude.)

III. FAULT GENERATOR DESIGN

A. *Using FPGAs*

An FPGA (Field Programmable Gate Array) combines on the same die flip-flops, combinatorial logic, and configurable routing. The FPGA hardware approach allows for parallel processing, contrary to a microprocessor's sequential operation. In this application, it enables the fault generator to decode the incoming PROFIBUS bits, and analyse in real-time characters and telegrams for several concurrent criteria, with a reaction delay in the ns range.

In this design the FPGA is used for:

- PROFIBUS serial signal reception
- real-time message analysis, with multiple concurrent trigger criteria
- flexible fault generation, with configurable duration, repetition rate and fault skipping.

Most real-time message decoding and checking for (combinations of) the user-selectable trigger conditions is done in state machines (SMs), which by nature require a lot of combinatorial functions. As the latter are prone to glitch and delay introduction, attention was paid during FPGA design to keep the signal path between two clock cycles as short as possible, to increase the throughput and keep the operation frequency as high as possible. This requires intensive use of registers, but allows correct operation at maximum bit rate [16].

B. *Serial receiver*

To enable optimal operation in noisy environments, the FPGA acquires 16 samples per bit (Fig. 3 shows the individual bits in a PB DP UART character), and 3 samples in the middle of the bit time are kept. The majority of the high/low levels of these 3 is retained, which results in a low sensitivity for noise, glitches, etc. This sample rate defines the main clock frequency in the FPGA: as the maximum bit rate is 12 Mbps, a value of 192 MHz (16 × 12 MHz) is chosen, resulting in a (approximately) 5 ns main clock period.

C. *Telegram analysis*

PROFIBUS telegrams have different formats (Fig. 4). As trigger conditions must be compatible with any SD-type, the position of each byte (e.g. Source Address SA) is message-type dependent and thus changes; refer to Fig. 9, reproduced after [13].

As example of a small state machine, Fig. 9 (top) shows the different locations (3rd or 6th position) of the SA, and the corresponding state machine. As every hardware state machine is independent from the other ones, they all work concurrently and the number of activated trigger conditions has no impact on the reaction time.

D. *Fault generation*

a) *Fault implementation*

Destroying (a series of) bits, bytes or complete telegrams is either achieved by RS485 transceivers – by far most the versatile and dynamic method – or by reed relays.

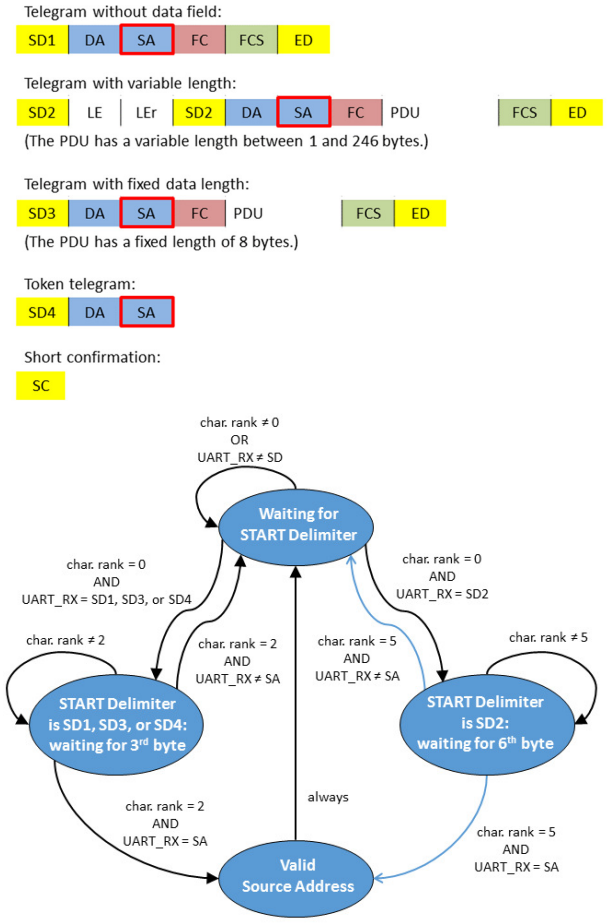

Fig. 9. (Top) Source Address position in PB DP telegram. (Bottom) State machine for Source Address trigger.

Two independent PROFIBUS RS485 transceivers (SN65HVD1176) force constant levels on the two bus lines (A and B) of the differential pair of the bus. These are individually controlled by different FPGA outputs for maximum flexibility (Fig. 10). The type of fault – any combination of A and B high or low – can be configured by the user. The transceiver at the input is an ISO1176 (Fig. 1).

Besides triggered "fast" destruction of parts of network telegrams, an "impedance mismatch", a current source to produce a shield current, etc. can be applied to the network using reed relays. The HMI allows the control of 2 normally open reed relays, with a closing time of max. 500 µs (including bouncing) and an opening time of max. 100 µs.

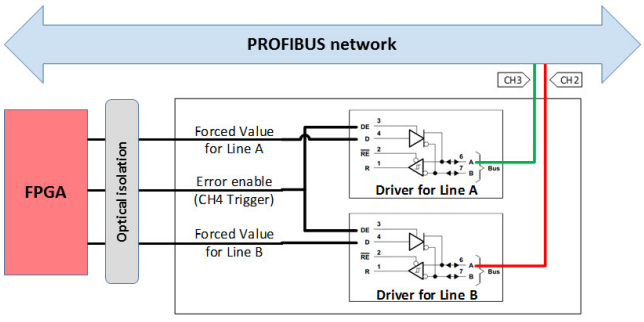

Fig. 10. Forcing the fault on the RS-485 network.

b) Duration – Sequence

The duration of a single fault can be set: its value can be expressed as a number of bit times or character times, and can vary in a very wide range. The minimum duration is one bit time at maximum bit rate, 83 ns at 12 Mbps; maximum 999 bit times can be selected. The maximum duration is chosen at 510 character times (2 maximum-length telegrams); at 9600 bps it is 0.58 s. The maximum duration at the more industrially relevant 1.5 Mbps bit rate is 3.74 ms of "character times" (no time between telegrams). Realistic length telegrams – as slaves typically do not transfer a lot of information – are far shorter than the maximum length; 510 characters is order of magnitude 15-20 telegrams (for one single fault).

The final trigger sequence is fully configurable:

- The waiting time ("delay") between successive fault triggers is expressed in ms and can have any value between 1 ms and 17 minutes.
- 0 up to 9999 trigger conditions can be skipped ("hold-off") in order to apply faults only after a certain number of triggers.
- The number of applied successive faults ("repetition") can also be set to any value between 1 and 31.

By combining these optional functionalities, almost any sequence of fault generation can be defined. Fig. 11 depicts the situation when the number of skipped faults and the number of successive faults are respectively set to 3 and 2, with a non-zero waiting time.

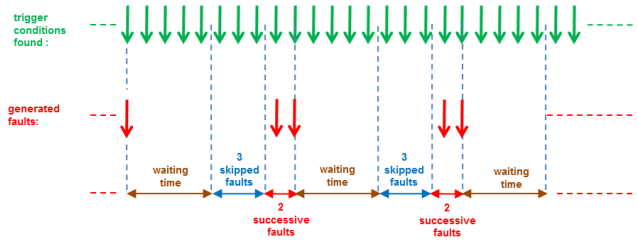

Fig. 11. Example of a sequence with waiting time ≠ 0, skipped faults = 3 and successive faults = 2.

c) The FPGA and custom PCB

The final system is an assembly of two boards, stacked together. The Terasic DE0-Nano development board (Fig. 12, top) holds the FPGA: a Cyclone IVE Intel® FPGA with 22.320 Logic Elements (LEs) and 154 General Purpose Inputs Outputs (GPIOs). The current application uses 4 % of the available LEs and 14 % of the available GPIOs.

The second one (Fig. 12, bottom) is a custom board with a 8-bit Microchip® microcontroller used for the Human Machine Interface (HMI), and interfaces with the FPGA via SPI (Serial Peripheral Interface). The HMI (4×16 LCD display and keyboard switches) allows for on-the-fly modifications that are instantly applied. Fig 1 depicts 4 trigger signal outputs from the FPGA: these can be used e.g. to trigger an oscilloscope measuring other than bus signals.

The reaction time of the FPGA is limited to 4 clock pulses (of 5 ns each) by using a pipelined design. The check on the trigger conditions itself starts at the end of each character, to be certain that calculations are performed on a valid character.

Fig. 12. The final assembly: FPGA board (top) and custom board (bottom).

IV. VALIDATION BY SIMULATION

The 1st phase of the system validation was done by gate-level simulation, after "Place & Route" in the FPGA, and taking into account the real propagation delays introduced by the configurable connections and the logic cells. All simulations were done with ModelSim [17].

To be even closer to real behaviour, an input signal was extracted from a live measurement in a 1.5 Mbps PB DP network. A VHDL testbench was written to read analogue values from the measurement ascii file obtained with an oscilloscope, to convert these to digital values, and finally use these as simulation inputs. It was thus possible to check that the expected functionality was reached both with "real" data but also with simulated data for rather random situations.

Fig. 13 shows a simulation result in the case of a Start Delimiter detection. The system was configured to generate a 10 µs output pulse whenever a Start Delimiter was found.

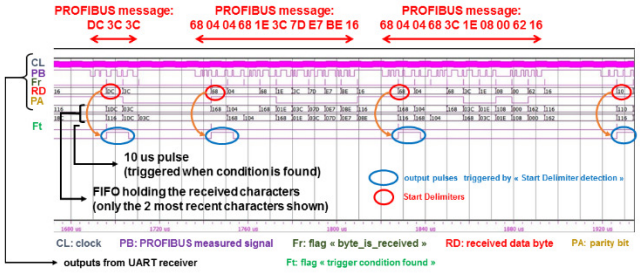

Fig. 13. Gate-level simulation result: Start Delimiter detection.

Simulation also proved useful to quickly and efficiently validate the system's ability to introduce a waiting time between successive actions, to skip some triggers, and to apply a predefined number of faults. These sequences and combinations can be checked in a flexible way in the simulation phase, but are tedious to check in a live network.

V. VALIDATION IN LIVE PROFIBUS DP NETWORKS

A. Fault injection

Fig. 14 shows both differential voltage and absolute voltages on lines A & B during a fault injection at 1.5 Mbps. Line B is forced low and line A is forced high on a cable of 100 m length.

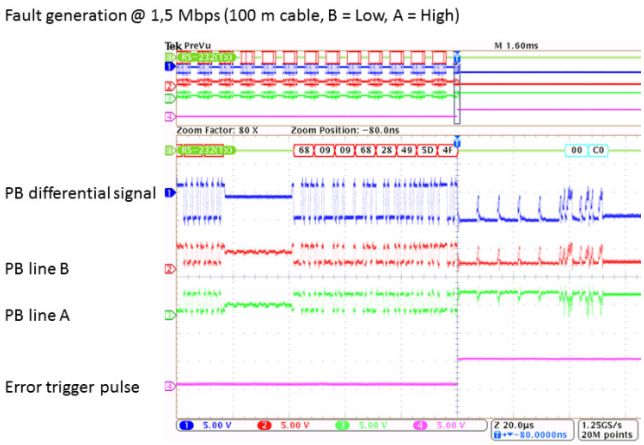

Fig. 14. Oscilloscope measurement of fault injection at 1.5 Mbps, showing both differential voltage and absolute voltages on lines A & B.

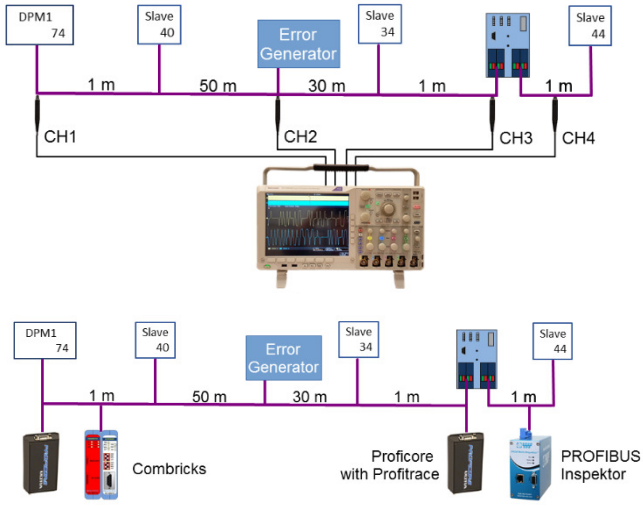

Fig. 15. Measuring equipment used for the validation in live networks.

B. *Networks and measuring equipment for the use cases*

The baseline networks were presented in Chapter II (Fig. 5). A UART decoding oscilloscope Tektronix DPO 4054B with isolating probes [18] was used for measurements on up to 4 points in the network (Fig. 15, top). Besides UART decoding by the oscilloscope, a MATLAB script was developed for further analysis. The trigger outputs of the fault generator can be used by extra oscilloscopes to determine the behavior of other network components (e.g. digital outputs of slave stations, analog torque or speed signals of drive systems, etc.).

Fig. 15 (bottom) shows a number of diagnostic tools that are connected to analyze what is actually reported on each fault: two ProfiCores with ProfiTrace software, a COMbricks (all Procentec), a diagnostic repeater (Siemens), and a PROFIBUS Inspektor (Indu-Sol). A short discussion on the reporting of these diagnostic devices – also depending on their location – is presented in the use cases. However, a detailed analysis of their diagnosis is beyond the scope of this paper.

C. *Use case 1: destroying 2 consecutive telegrams to a slave*

The following set-up is configured in the fault generator settings: 2 consecutive SD2 telegrams from DPM1 (address 74) to the slave with address 34 are destroyed every 10 s. Line B is forced high and line A low so that B-A is positive; the network bit rate is 1.5 Mbps and the fault duration is 3 characters.

Fig 16 (left) shows healthy data exchange telegrams left from the middle of the trace. The telegram in the middle of the oscilloscope image is destroyed, as well as the repeated telegram in the right part. Fig. 16 (right) shows in detail the 3 destroyed characters: these are only mildly shifted in voltage level at the master side (source of the telegram, and quite far away from the imposed fault levels), and on all the other locations a complete destruction can be observed.

In Fig. 17 a ProfiTrace recording (at the side of the master) is shown. As the slave doesn't receive a request from the DPM1 twice in a row, and consequently does not respond, the master's repeat limit is exceeded. "Repeat (lost)" is indicated the second time. The master waits for the next cycle to set parameters (indicated with "Sync"). When the slave is ready, it will (later) again get in the communication sequence.

Although a detailed analysis of what the different diagnostic tools report is not within the scope of this paper, we can briefly indicate the possibilities for analysis and comparison. All diagnostic tools detect this fault. It is seen either as a restart or a sync of slave 34. Some tools installed close to the source also detect repeats, some also detect illegals (close to the destination).

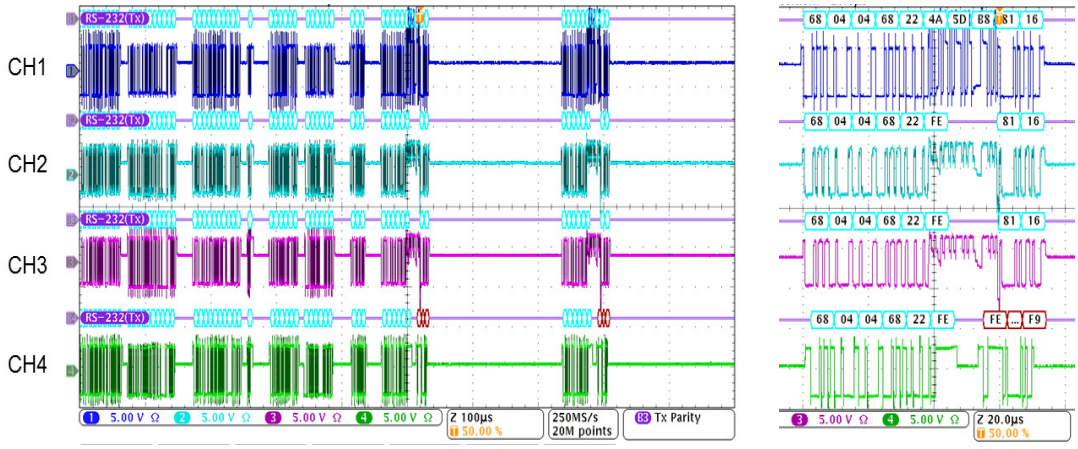

Fig. 16. Use case 1 oscilloscope recording. Left figure: the telegram to station 34 is destroyed, and the repeat (towards the right) also. Right figure: detail of the destroyed telegram.

FrameNr	Timestamp	Attention	Frame	Addr	Service	Msg type	Req/Res	SAPS	DataLen	Data
221076	21602945...		ACK			Short acknowledge	Res			
221077	21602976...		SD1	74->109	FDL Status		Req			
221078	21603226...		SD4	74->74	Token pass	Pass token				
221079	21603272...		SD2	74->34	SRD HIGH	Data Exchange	Req		1	D2
221080	21603358...		SD2	74<-34	DL	Data Exchange	Res		4	00 00 02 01
221081	21603477...		SD2	74->40	SRD HIGH	Data Exchange	Req		1	D2
221082	21603559...		SD2	74<-40	DL	Data Exchange	Res		1	00
221083	21603656...		SD2	74->44	SRD HIGH	Data Exchange	Req		1	D2
221084	21603742...		SD2	74<-44	DL	Data Exchange	Res		1	00
221085	21603840...		SD2	74->84	SRD HIGH	Data Exchange	Req		1	00
221086	21603925...		ACK			Short acknowledge	Res			
221087	21603956...		SD1	74->110	FDL Status		Req			
221088	21604206...		SD4	74->74	Token pass	Pass token				
221089	21604252...		SD2	74->34	SRD HIGH	Data Exchange	Req		1	D2
221090	21604528...	Repeat(lost)	SD2	74->34	SRD HIGH	Data Exchange	Req		1	D2
221091	21604809...		SD2	74->40	SRD HIGH	Data Exchange	Req		1	D2
221092	21604891...		SD2	74<-40	DL	Data Exchange	Res		1	00
221093	21604988...		SD2	74->44	SRD HIGH	Data Exchange	Req		1	D2
221094	21605074...		SD2	74<-44	DL	Data Exchange	Res		1	00
221095	21605172...		SD2	74->84	SRD HIGH	Data Exchange	Req		1	00
221096	21605257...		ACK			Short acknowledge	Res			
221097	21605288...		SD1	74->111	FDL Status		Req			
221098	21605538...		SD4	74->74	Token pass	Pass token				
221099	21605584...		SD2	74->40	SRD HIGH	Data Exchange	Req		1	D2
221100	21605666...		SD2	74<-40	DL	Data Exchange	Res		1	00
221101	21605764...		SD2	74->44	SRD HIGH	Data Exchange	Req		1	D2
221102	21605850...		SD2	74<-44	DL	Data Exchange	Res		1	00
221103	21605947...		SD2	74->84	SRD HIGH	Data Exchange	Req		1	00
221104	21606033...		ACK			Short acknowledge	Res			
221105	21606064...	Sync	SD2	74->34	SRD HIGH	Set Parameters	Req	62->61	54	F8 02 03 0B 80
221106	21606553...		ACK			Short acknowledge	Res			
221107	21606585...		SD1	74->112	FDL Status		Req			
221108	21606834...		SD4	74->74	Token pass	Pass token				
221109	21606881...		SD2	74->40	SRD HIGH	Data Exchange	Req		1	D2
221110	21606963...		SD2	74<-40	DL	Data Exchange	Res		1	00
221111	21607060...		SD2	74->44	SRD HIGH	Data Exchange	Req		1	D2
221112	21607146...		SD2	74<-44	DL	Data Exchange	Res		1	00
221113	21607244...		SD2	74->84	SRD HIGH	Data Exchange	Req		1	00
221114	21607330...		ACK			Short acknowledge	Res			
221115	21607361...		SD1	74->113	FDL Status		Req			
221116	21607610...		SD4	74->74	Token pass	Pass token				
221117	21607657...		SD2	74->40	SRD HIGH	Data Exchange	Req		1	D2

Fig. 17. ProfiTrace recording of Use case 1.

D. Use case 2: destroying token passes between masters

Some measurements and short analysis of the stability behavior of the token ring between DP masters have been described in e.g. [10]. These are limited to 500 Kbps (not a typical PROFIBUS DP bit rate for industrial applications); e.g. [11] even goes as low as 19200 bps in particular tests. In contrast to these severe limitations, the developed fault generator destroys token passes in a freely configurated sequence, to industrial DPM1 stations, and at a high bit rate (in a 1.5 Mbps network, a 6 Mbps one for the reaction time measurement) until the token ring is reconfigured (Fig. 5).

The following set-up is configured in the fault generator settings: 3 consecutive SD4 Token Pass telegrams to DPM1 with address 2 are destroyed every 10 s. Line B is forced high and line A low, so that B-A is positive; the network bit rate is 1.5 Mbps. The fault duration is 3 characters, in this case lasting longer than the remainder of the telegram: the trigger is reached at the end of the 2nd character in the SD4 telegram.

Fig 18 (left) shows 3 destroyed SD4 telegrams to station 2, the first one under the orange "T" mark on top. The right figure shows a detail of a destroyed SD4 telegram on 4 locations. Again, these are only mildly shifted in voltage level at the master side and still correct (source of the telegram, and quite far away from the imposed fault levels), and on all the other locations a complete destruction can be observed.

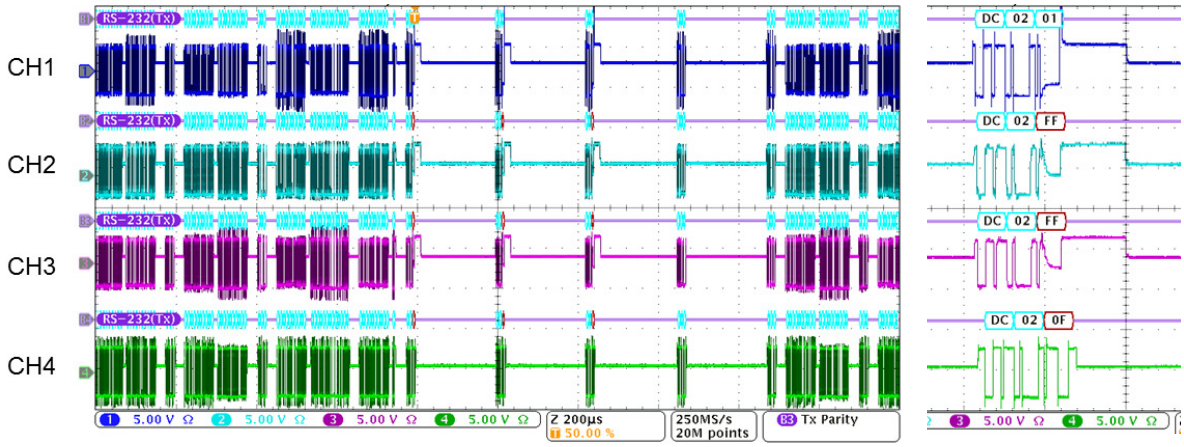

Fig. 18. Use case 2 oscilloscope recording. Left figure: 3 SD4 telegrams to station 2 are destroyed, the first one under the orange "T" dot on top. Right figure: detail of a destroyed SD4 telegram.

In Fig. 19 a ProfiTrace recording (at the side of the master) is shown. The Master on address 1 notices no communication from any master within the slot time, and repeats the token pass to Master station 2 two times. Station 2 is removed from the LAS (List of Active Stations), and the token is passed from 1 to 3. The latter doesn't expect it from 1, so it is repeated before station 3 accepts the token.

Brief analysis: all diagnostic tools detect this fault, either as a repeat (close to the source) or as an illegal (close to the destination). The COMbricks – in this case a Head Station Type 1C (HW V1.9; SW V1.303), with as Repeater a Type 1 (HW V1.4; SW V1.14) – does not indicate repeats for token passes.

FrameNr	Timestamp	Attention	Frame	Addr	Service	Msg type	Req/Res	SAPS	DataLen	Data
45220	3468948....		SD2	1<-34	DL	Data Exchange	Res		4	00 00 02 01
45221	3469068....		SD2	1->84	SRD HIGH	Data Exchange	Req		1	00
45222	3469154....		ACK			Short acknowledge	Res			
45223	3469186....		SD4	1->2	Token pass	Pass token				
45224	3469409....	Repeat(lost)	SD4	1->2	Token pass	Pass token				
45225	3469632....	Repeat(lost)	SD4	1->2	Token pass	Pass token				
45226	3469860....		SD4	1->3	Token pass	Pass token				
45227	3470084....	Repeat	SD4	1->3	Token pass	Pass token				
45228	3470131....		SD2	3->44	SRD HIGH	Data Exchange	Req		1	55
45229	3470216....		SD2	3<-44	DL	Data Exchange	Res		1	00
45230	3470314....		SD4	3->1	Token pass	Pass token				
45231	3470360....		SD2	1->34	SRD HIGH	Data Exchange	Req		1	83
45232	3470446....		SD2	1<-34	DL	Data Exchange	Res		4	00 00 02 01
45233	3470566....		SD2	1->84	SRD HIGH	Data Exchange	Req		1	00
45234	3470651....		ACK			Short acknowledge	Res			
45235	3470682....		SD4	1->3	Token pass	Pass token				
45236	3470730....		SD2	3->44	SRD HIGH	Data Exchange	Req		1	55
45237	3470814....		SD2	3<-44	DL	Data Exchange	Res		1	00
45238	3470912....		SD4	3->1	Token pass	Pass token				
45239	3470959....		SD2	1->34	SRD HIGH	Data Exchange	Req		1	83
45240	3471044....		SD2	1<-34	DL	Data Exchange	Res		4	00 00 02 01
45241	3471164....		SD2	1->84	SRD HIGH	Data Exchange	Req		1	00
45242	3471250....		ACK			Short acknowledge	Res			
45243	3471281....		SD4	1->3	Token pass	Pass token				
45244	3471328....		SD2	3->44	SRD HIGH	Data Exchange	Req		1	55
45245	3471413....		SD2	3<-44	DL	Data Exchange	Res		1	00
45246	3471511....		SD4	3->1	Token pass	Pass token				
45247	3471558....		SD2	1->34	SRD HIGH	Data Exchange	Req		1	83
45248	3471643....		SD2	1<-34	DL	Data Exchange	Res		4	00 00 02 01
45249	3471763....		SD2	1->84	SRD HIGH	Data Exchange	Req		1	00
45250	3471848....		ACK			Short acknowledge	Res			
45251	3471880....		SD4	1->3	Token pass	Pass token				
45252	3471927....		SD2	3->44	SRD HIGH	Data Exchange	Req		1	55
45253	3472012....		SD2	3<-44	DL	Data Exchange	Res		1	00
45254	3472110....		SD4	3->1	Token pass	Pass token				
45255	3472156....		SD2	1->34	SRD HIGH	Data Exchange	Req		1	83
45256	3472242....		SD2	1<-34	DL	Data Exchange	Res		4	00 00 02 01
45257	3472362....		SD2	1->84	SRD HIGH	Data Exchange	Req		1	00
45258	3472447....		ACK			Short acknowledge	Res			
45259	3472478....		SD1	1->2	FDL Status		Req			
45260	3472531....		SD1	1<-2	Act.RDY		Res			
45261	3472599....		SD4	1->2	Token pass	Pass token				
45262	3472822....	Repeat	SD4	1->2	Token pass	Pass token				
45263	3472870....		SD2	2->40	SRD HIGH	Data Exchange	Req		1	DD
45264	3472951....		SD2	2<-40	DL	Data Exchange	Res		1	00
45265	3473050....		SD4	2->3	Token pass	Pass token				

Fig. 19. ProfiTrace recording of Use case 2.

E. Overall reaction time

The reaction time of the FPGA is 4 x 5 ns (Section III). The typical remaining time delays for the generation of the fault trigger are measured in Fig 20, that shows a SD4 token passing telegram in a 6 Mbps network. Channel 4 on the oscilloscope now shows the trigger signal. Channel 2 (light blue) is at the fault generator, trigger condition is fulfilled at the end of the last telegram character "01", so that there is a clear view on the destruction (line B is H, line A is L). From the end of "01" (which is correctly indicated by the decoding, at the end of 2 "H" bits of 167 ns each) to the start of the trigger pulse is about 70 ns. The input delay of the optocoupler is max. 55 ns, the FPGA reaction time is 20 ns, and the output optocoupler introduces typically 31 ns of delay time; the

components are listed in Section III D. The 70 ns in the measurement is thus within the specifications.

From "trigger condition fulfilled" to start of destruction is 80.4 ns (cursor measurement in Fig. 20, insert); at 6 Mbps it means that the character after the one that fulfils the trigger condition is destroyed (11 x 167 ns = 1.84 µs). Also at 12 Mbps, it is the 1st character after the one that fulfils the trigger condition that is destroyed (one character takes 11 x 83 = 913 ns), which means that for all bit rates the character after fulfilling the trigger conditions is destroyed.

VI. Conclusions and Future Work

A low-cost, versatile and configurable trigger and fault generator for PROFIBUS DP has been developed. It proves very useful for training purposes, analysis of the behavior of diagnostic tools and for in-depth testing of more complex industrial problems. It covers all PROFIBUS DP bit rates up to 12 Mbps, and it covers 48 % of the "interesting, highly dynamic" faults.

The reaction time of the FPGA – to calculate if the trigger conditions are fulfilled – is limited to 4 clock cycles of 5 ns. The trigger condition itself is checked at the end of the character that fulfils the trigger condition. The trigger signal is typically available 70 ns after the end of the character that fulfils the trigger condition. Total reaction time to the fault impact on the bus itself is about 80 ns.

Total cost is less than € 200 (using vendors providing quick delivery of components, so a further decrease of the price is possible), without a mounting box.

As future application, it is planned to use the fault generator in research concerning "big data" analysis of bus phenomena in real-time for early detection of bus failures. Finally, the idea of a fault generator for Ethernet based industrial networks remains to be explored.

Acknowledgment

Bram Vanseveren (UGent), Annemarie Kokosy and Bruno Stefanelli (both Junia-ISEN), Jos De Brabanter (KU Leuven) and Frederic Depuydt (now ArcelorMittal Gent, formerly at KU Leuven) for advice and support. Engineering staff of ArcelorMittal Gent for inspiration and advice. This work was funded by the Interreg Va 2 Seas project 2S01-049 INCASE.

References

[1] https://www.profibus.com/

[2] "The New Rapid Way to Profibus DP", M. Popp. PROFIBUS Nutzorganisation, 2003, Karlsruhe, Germany..

[3] "PROFIBUS: Theory & Practice, Engineering & Troubleshooting"; F. Depuydt, W. Hauspie, H. Derre, T. De Landtsheer, S. Noppe, M. Troch, D. De Schuyter, P. Saey. Internal course notes INCASE, 2019.

[4] "CAN-PROFIBUS interface for communication between MV1000 drive and PLC"); E. Hubaux, MSc thesis, KU Leuven, 2021.

[5] "The PROFIBUS Protocol Observation"; P. Drahos, I. Belai. 9th IFAC Symposium on Advances in Control Education, Nizhny Novgorod, Russia, 2012.

[6] "Artificial Neural Networks and Signal Clipping for PROFIBUS DP diagnostics", G. Sestito, P. Souza, E. Mossin, D. Brandão and A. Dias. 12th IEEE international conference on industrial informatics, INDIN 2014, pp. 242–247.

[7] "Automatic Diagnosis for Profibus Networks", E. Mossin, D. Brandão, G. Sestito, and R. Torres. Journal of Control, Automation and Electrical Systems 27.6 (2016): pp. 658-69.

[8] "Procentec fault generator"; https://procentec.com/products/automation-tools/training-tools/fault-generator/

[9] https://procentec.com/products/automation-tools/training-tools/profibus-dp-training-device/. DP slave as fault generator.

[10] "Assessment of PROFIBUS networks using a fault injection framework"; J. Carvalho, A. Carvalho, P. Portugal. IEEE ETFA, 2005.

[11] "An Open Implementation of Profibus DP"; D.K. Tran, P. Pisa, P. Smolik. Real Time Linux Workshops, Dresden, Germany, 2009.

[12] "Design of an Arduino based low-cost error generator for PROFIBUS DP"; P. Saey, W. Hauspie, H. Derre, T. De Landtsheer, A. Kokosy, J. Knockaert. IEEE Conference on Emerging Technologies and Factory Automation ETFA 2014, Sep. 16-19, Barcelona (Spain).

[13] "PROFIBUS Manual"; M. Felser, 2017. https://felser.ch/profibus-manual/index.html

[14] "Vortex report" 2019 & 2020, R. Heidl et. al., https://www.indu-sol.com/en/

[15] "PROFIBUS troubleshooting a.d.h.v permanente logging met COMbricks in ArcelorMittal Gent" ("PROFIBUS troubleshooting using permanent logging with COMbrickx in ArcelorMittal Gent"); J. Mortier, MSc thesis, KU Leuven, 2015.

[16] "Methodology for FPGA-based system design". J. Capron. Course notes INCASE, 2017.

[17] https://www.mentor.com/products/fv/modelsim/

[18] http://www.tektronix.com

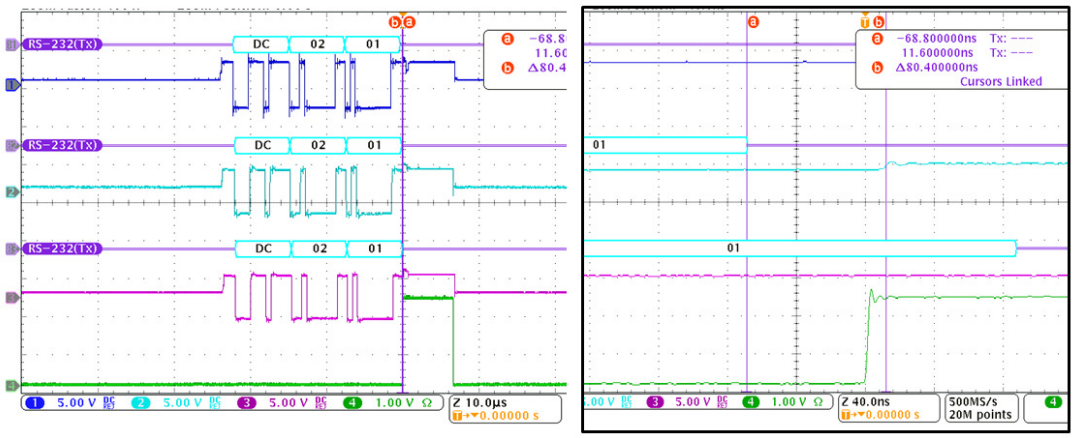

Fig. 20. Reaction time measurement at 6 Mbps.