Field Bus for Data Exchange and Control of Modular Power Electronic Systems with High Synchronisation Accuracy of ± 4 ns

Stefan Rietmann^{*a)} Non-member, Simon Fuchs^{*} Non-member
André Hillers^{*} Non-member, Jürgen Biela^{*} Non-member

(Manuscript received July 24, 2018, revised Nov. 9, 2018)

This paper proposes a new field bus protocol based on the IEEE 802.3 Ethernet standard. The protocol enables fast data exchange between the modules and synchronised control in the modules of modular power electronic systems. With increasing switching frequency, a highly accurate synchronisation of the different modules is a necessary requirement for a control bus. The proposed field bus protocol provides a stable and efficient scheme including a novel data frame structure and allows the realisation of a synchronisation accuracy of ± 4 ns based on the 1 GBit Ethernet standard. For validation of the new protocol, a prototype system was developed and the measurement results are reported herein. The results show that the implementation satisfies the specifications. Finally, different use-cases and the achievable data rates for the given implementation were derived.

Keywords: communication protocol, field bus, synchronisation, modular power electronic systems

1. Introduction

In power electronic systems, more and more modular converter structures are applied due to their increased availability (redundancy), simple scaling and improved output quality (e.g. less current ripple by interleaving⁽¹⁾). The most prominent example for modular converters is the modular multilevel converter (MMC). For the MMC various possibilities for distributed control systems can be found in literature. For example, distributed control methods⁽²⁾, distributed protection control⁽³⁾ or modulation methods that allow to distribute the switching signal generation on the modules⁽⁴⁾⁽⁵⁾. As for these distributed current control methods, computational power is required on each module/cell (e.g. an FPGA), also more complex communication systems for data exchange and/or communication can be implemented in the modules. Figure 1 shows an exemplary hardware implementation of such distributed control systems. On each of the modules in the shown MMC stack, there is an FPGA control board connected to an optical communication bus. As can be seen in Fig. 1, the size of the module hardware is increased only very little by the FPGA control board. With a field bus communication system, the number of data connections can be drastically reduced compared to a point-to-point implementation of a master controller to each module/cell. For a daisy chained setup, a maximum of two physical transmission lines per slave is required (as e.g. with EtherCAT). A communication system for such modular converter systems typically has the following properties:

(1) As all modules are of the same type and require the

same data to be communicated, the individual data frames received and transmitted by the modules are also of the same size.

- (2) The amount of exchanged data per module is low. It is typically in the range of 5 to 10 Bytes per communication and/or switching cycle (cf. Sect. 2).
- (3) The data exchange on the bus system is bidirectional with respect to all slaves and the master. Every bus member can read from and write to the bus.
- (4) All modules need to run on the same clock.
- (5) All modules have to refer to a common point in time, such that they can perform simultaneous or synchronised switching actions (e.g. interleaving). Especially for high switching frequencies, the accuracy of this synchronisation is required to be high (low nanosecond range).
- (6) Some communication error detection has to be

Fig. 1. Photo of an MMC stack (3 modules) using the proposed SyCCo-Bus. It can be seen, that each module is equipped with the same FPGA control hardware

a) Correspondence to: Stefan Rietmann. E-mail: rietmann@hpe.ee.ethz.ch

^{*} Laboratory for High Power Electronic Systems, ETH Zürich
Physikstrasse 3, 8092 Zürich, Switzerland

present to protect the system from e.g. implementing wrong reference values.

- (7) The hardware requirements should be rather low, such that not too much FPGA area is occupied by the communication logic itself rather than the control logic.

In ⁽⁶⁾, the CAN bus is used to control an MMC. The synchronisation accuracy was found to be $\pm 20 \mu\text{s}$. In ⁽⁷⁾⁻⁽⁹⁾ the commercial EtherCAT field bus system has been used to control an MMC topology. EtherCAT features a synchronisation option, that is claimed to result in a jitter of less than $1 \mu\text{s}$ ⁽¹⁰⁾. However, in literature the jitter of the synchronisation signals ranges between $\pm 20 \text{ ns}$ ⁽⁹⁾ and $15 \mu\text{s}$ ⁽⁷⁾. For achieving a more precise synchronisation accuracy of $\pm 5 \text{ ns}$, a basic concept of a daisy-chained field bus protocol is proposed in ⁽¹¹⁾. It is based on 100 Mbit Ethernet and dedicated physical layer ICs in combination with an FPGA. The master/slaves are interconnected with bidirectional single fibre optic cables (FOC) resulting in one interconnection per slave. The protocol tolerates different frame sizes for the individual slaves connected to the bus system and does not require to have knowledge on the number of slaves prior to the operation. This results in a rather complicated start-up and synchronisation procedure and occupies a large amount of logic elements/registers in the FPGAs. Therefore, in this paper the following contributions are made:

- A simplified frame structure is proposed, where the amount of received/transmitted data (the frame size) per slave is equal for all slaves and the number of slaves is known a priori. These assumptions lead to a much simpler, more robust and less logic consuming implementation compared to ⁽¹¹⁾. The delay introduced by the data passing through each slave is also drastically reduced, because all operations on the frame can be performed within one clock cycle and no buffering of multiple frame parts is necessary.
- The low protocol overhead allows significantly higher data rates compared to EtherCAT ⁽¹⁰⁾.
- The protocol also features a continuous synchronisation evaluation to avoid drift effects in the synchronisation accuracy of $\pm 4 \text{ ns}$. The synchronisation algorithm also needs no additional communication between individual slaves and/or the master.
- Running two or more synchronous bus systems in parallel is easily possible without changing the protocol and loss of synchronisation accuracy.
- Altera Cyclone V GX FPGAs are used enabling an all-integrated transceiver setup using Altera IP cores. They are featuring the Gigabit Ethernet standard which results in a high data throughput while allowing a still reasonable clock frequency of 125 MHz.

The implemented system including the proposed protocol is referenced as **Synchronous-Converter-Control-Bus (SyCCo-Bus)**.

The paper is organized as follows. Section 2 describes the various requirements necessary for a bus system in modular converter systems. In Sect. 3 the applied field bus concept, the operating principles and the novel frame structure are introduced. The individual module time synchronisation scheme and the expected synchronisation accuracy are

explained. Section 4 refers to the actual hardware implementation including an FPGA based prototype system. The VHDL implementation of the protocol is described in Sect. 5. In Sect. 6 the concept is tested and particular timing measurement results are shown. Different use-cases and the achievable data rates are derived and analysed in Sect. 7. The paper concludes in Sect. 8.

2. Data Exchange Requirements in Modular Converter Systems

To explain the requirements for the data exchange in modular converter systems, a medium voltage MMC setup with $N = 90$ modules and a switching frequency of $f_{\text{sw}} = 10 \text{ kHz}$ serves as an example in this section ⁽¹²⁾. Figure 2a) shows a possible bus interconnection for the MMC. All modules are connected in a daisy chain manner. The MMC is assumed to be operated using a PWM scheme (here level shifted PWM ⁽⁴⁾⁽⁵⁾). Note, that the switching signals are processed on the modules itself, whereas the duty cycle determination is performed on a central control unit and each module receives a duty cycle for each PWM cycle. All modules (slaves) have to submit their capacitor voltages (12 bit) to the central control unit (master) once every switching period (necessary for PWM modulation ⁽⁴⁾⁽⁵⁾). For protection reasons it can be reasonable to transmit also current measurements (12 bit) on the modules ⁽³⁾. Furthermore, some information on the semiconductor/module-state (8 bit) as well as temperature information (8 bit) could be transmitted from the slaves to the master. All this data sums up to a worst case scenario of 5 Byte per slave. The central control unit has to distribute data to the modules, the duty cycle for the upcoming switching period (10–12 bit) as well as the maximum and minimum values for the current and module voltage ($4 \times 12 \text{ bit}$) ⁽³⁾. This sums up to 8 Byte of data per slave. For the presented protocol, both directions have the same frame structure, such that one needs to use 8 Byte data/module for both directions plus a CRC byte for a data integrity check.

The frame has to be communicated from the master to all modules and back again, such that $2 \cdot N \cdot (8 + 1)$ Byte of data have to be processed/transmitted within the switching period T . However, this is not enough information to define the required maximum delay introduced by the communication system. The larger the delay or round trip time (RTT), the less time remains for computations on the central control unit (cf. Fig. 3). Thus, the amount that the RTT is shorter than the switching cycle T , can be used to perform controller computations. If one, for example, requires $40 \mu\text{s}$ for the control computations, the RTT has to be shorter than $100 \mu\text{s} - 40 \mu\text{s} = 60 \mu\text{s}$. Considering the physical link speed (bandwidth of the communication channel) this means a minimum bandwidth of $2 \cdot 90 \cdot (8 + 1) \text{ Byte} / 60 \mu\text{s} = 216 \text{ Mbit/s}$. Note, that this value exceeds limits of the 100 Mbit Ethernet standard and therefore also the implementation in ⁽¹¹⁾ and EtherCAT ⁽¹⁰⁾. Note, that the protocol will add some overhead that further increases minimal bandwidth.

The switching signals are calculated on the individual modules' FPGA based on the received duty cycle as shown in Fig. 2b). Each module has its own carrier signal, where all carriers are required to be synchronous. A synchronisation error $\varepsilon_{\text{sync},k}$ of a module can result in an error of the module

Fig. 2. Application of the proposed field bus on an MMC, where a) shows a possible interconnection of the MMC modules in a daisy chain manner and b) shows the distributed PWM carriers, the duty cycles and the output voltage of the MMC modules. The PWM carriers are synchronized with the SyCCo-Bus and the duty cycle value are received over the SyCCo-Bus. In the upper part of b) it is shown that a synchronisation error of the PWM carriers results in an module output voltage error of the MMC modules

Fig. 3. Typical control timing for modular converters with a central control unit: The central controller receives measurement data from the modules and converts the controller computations based on this data. After that it sends the resulting commands to the modules which perform the switching actions accordingly. Note, that the communication delay, the round-trip time RTT , of the communication system shortens the time available for the control computations to $T - T_d$, where $T = 1/f_{sw}$. The computations and communication actions relevant for the current period κ in which the switching actions are performed are drawn black

output voltage V_k (upper part of Fig. 2b) because the module switches too early or too late⁽⁴⁾⁽⁵⁾. Also when considering phase-shifted PWM schemes⁽²⁾, where all modules have a well defined phase shift in their individual carriers, a synchronisation of all modules is necessary to precisely implement this phase shift. Therefore, the accuracy of the synchronisation should be higher than the possible implementation accuracy of the PWM generated on the modules. For a 10 bit PWM, this results in a minimum synchronisation accuracy of

$$1/f_{sw}/(2^{10} - 1) = 97.8 \text{ ns.}$$

Due to the typically rather low switching frequency in MMC applications, the requirements concerning the synchronisation accuracy and the communication delay are less demanding. Nevertheless, for systems with higher switching frequencies⁽¹³⁾, the synchronisation accuracy must be much higher. For example, a switching frequency of 100 kHz with the same duty cycle precision (10 bit) would result in a required accuracy of 9.5 ns.

Fig. 4. Frame schematic: The frame is divided into two distinct sections, the header and the data section. The header section mainly contains the preamble and the SFD bytes. Furthermore, the data section consists out of multiple slave specific sections of which each contains a slave data and a CRC section. Note that the number of possible slaves is currently limited to 255 by the current frame structure since the SCB contains 8 bit. This limitation can be solved by increasing the number of SCB bytes

3. Operating Principles

3.1 Field Bus Concept The proposed field bus protocol presumes a daisy-chained network topology. Considering the data flow direction each module is linked to its successor over the forward transmission path and hence the predecessors are attached on the backward transmission path. Compared to a star-like topology, this field bus concept does only allow broadcast message transfers, meaning that the complete set of information is always sent to all participants.

To initialize a transmission process the master module is issuing the data frame. During the transmission on the forward path each slave module has the chance to read data from the bus and write information back to the bus. A slave being at the end of the forward path chain detects its state as last slave, exchanges information with the bus frame and returns the frame on the backward path to its predecessor.

3.2 Frame Structure The frame structure as shown in Fig. 4 and especially the header partition is mainly based on the Gigabit Ethernet frame defined in IEEE 802.3⁽¹⁴⁾. A complete frame consist of:

- Preamble, 7 octet
- Start-of-Frame Delimiter (SFD), 1 octet
- Slave Counter Byte (SCB), 1 octet
- Data, m byte per slave for n slave modules
- CRC, 1 byte per slave for n slaves modules

In contrast to the IEEE 802.3 standard only the key elements such as the preamble and the start frame delimiter (SFD) octets have been adopted. The destination and address bytes have been omitted since the master is always addressing all slaves. As a replacement for the addressing scheme a frame internal counter byte, the Slave Counter Byte (SCB), is introduced. The complete system is not limited to any number of slaves. Nevertheless, it is worth to note that a slave count exceeding 255 slaves would require a larger SCB section. Due to the delay introduced by each slave, one has to consider the increasing round trip time (RTT) and therefore a decreasing data rate per slave for an increasing number of slaves. The static payload region contains a distinct data region and an additional CRC byte per slave.

3.3 Module Data Exchange The data exchange between the bus and the slaves is a time critical process since its duration directly contributes to the overall system latency. This latency is denoted in Fig. 6 as $T_{Forward,i}$ and $T_{Backward,i}$, respectively. Achieving a low latency data exchange will result in a high bus throughput. The static structure of the frame allows an efficient and simple data exchange with low latency on the slaves. A finite state machine (FSM) representation of the internal data exchange logic is given in Fig. 5. The complete frame is looped through each slave module. The internal state machine is triggered by the arrival of the preamble

Fig. 5. The Data Exchange Scheme describes the processing of the novel static data frame. The inner illustration shows the number of bytes processed through every step in the complete processing of the data frame

and followed by the SFD detection. Note, that a slave module does not count the number of preamble bytes rather than it checks for the specific SFD pattern. Next, the SCB byte is read and incremented on the fly. From the previously read SCB value the slaves can estimate their position in the system. Furthermore, the number of bytes per slave are defined in advance and hence the position of the slave specific data in the frame can be calculated after the appearance of the SFD. The module has to remain in the *WaitForSlot* state until the first byte of its own data arrives. At this point, the state changes to *Exchange Data*. The incoming frame is stored in internal registers and the data prepared to be sent back to the master is replacing the read data on the bus. Since all data is looped through all modules, a simple multiplexer logic as illustrated in Fig. 9 is sufficient to change the output between the incoming data frame and the stored, slave specific data. As a final step, the CRC byte which has been calculated during the *Exchange Data* state is written onto the bus. Eventually, the slave changes back to the *Idle* state and the remaining incoming data is passed on. The complete frame is received and further transmitted by every module but only the module specific part of the frame is modified.

3.4 Synchronisation The different modules linked via the SyCCo-Bus do not share a common time base nor the same base clock. However, as mentioned in the introduction, it is essential for modular distributed power electronic systems to run synchronously. Therefore, a scheme which provides a common time base and a common base clock is necessary. This scheme has to fulfill the following three requirements:

Fig. 6. Synchronisation Schematic: The illustration describes the round trip time measurement of a single slave. Note that the forward path and the backward path are equally long in terms of communication time

- Same frequency: All modules including the master module need to run on the very same clock frequency.
- Global Reference: All modules must refer to a single common point in time which needs to be an event observable by each participant with the same accuracy.
- Same clock phase: The clock of each module needs to maintain the same phase, meaning that the rising and falling edges of the clock signal have to overlap.

In this paper, the proposed protocol aims at a synchronising accuracy of $\frac{T}{2}$ where T denotes the clock period. A small outlook on how to increase the synchronisation accuracy is given in Sect. 3.4.3.

3.4.1 Same Frequency Requirement The first requirement is to have a common clock frequency on all modules to ensure that the internal logic runtime is the same on each module. Therefore, a reference clock has to be shared with every single module. For this purpose Ethernet uses 8b/10b en- and decoding to ensure enough alternating 1 and 0 bits⁽¹⁴⁾ (cf. Return-to-Zero). This allows to perform a clock data recovery (CDR) on each module.

3.4.2 Global Reference Requirement To ensure the requirement of a global time reference, the proposed frame structure includes the SFD as a common time token. A possible application example for the global time reference would be the determination of a starting point for converter switching activities or the start of the PWM cycle. The core idea of using the SFD byte is to detect the event when the last module starts to receive data and estimated when it has finished processing them. With these information it is possible to calculate the individual waiting time until a certain event occurs for each module. This time-to-wait is further denoted as $T_{\text{valid},i}$ where the index i references to the individual modules. Since this event can not be observed directly by the modules, it needs to be estimated through calculation and on-line measurements.

The SFD byte is observed twice by each module in each communication round, once during the forward transmission and once during the backward transmission. The time measured between sending the SFD during the forward transmission and receiving it during the backward transmission is referenced as $T_{\text{loop},i}$ as depicted in Fig. 6. $T_{\text{loop},i}$ includes the slave internal latency $T_{\text{Forward},i}$ and $T_{\text{Backward},i}$, the transceiver delay T_{receive} and $T_{\text{transmit},i}$ and the physical transmission delay as indicated in Fig. 6. In contrast to⁽¹¹⁾ a simple and accurate determination of a single and global point in time, on which all modules can equally rely, is possible, as explained

in the following.

The static frame structure and the fact that all slave modules are identical enables the assumption that the internal logic delays during the forward transmission is equal to those during the backward transmission. Furthermore, to ensure no differences of the physical transmission delays between the forward and the backward path, a FOC and a bidirectional compact small form-factor pluggable (CSFP) modules are used for the forward and backward transmission path.

To calculate the waiting time, each module determines the global point in time where the data has been received by all participants. For that purpose the modules count the time between the event of writing the SFD byte to the forward path and receiving the SFD from the backward path (cf. Fig. 6) based on their own internal clock. By introducing the same propagation delay on the backward transmission path as on the forward transmission path, the time measured can be divided by two to determine the point in time where the last slave module receives the frame. In contrast to⁽¹¹⁾, due to the static frame structure it is known by each slave how much time has to be taken into account until the last slave has completely received its data and CRC bytes. Hence, $T_{\text{valid},i}$ can be stated as

$$T_{\text{valid},i} = \left(\frac{C_i}{2} + 1 + n \cdot (m + 1) \right) \cdot f_{\text{slave}}^{-1} \dots \dots \dots (1)$$

where C_i is the module individual counter number, n the number of slave modules and m the bytes per slave module as defined in Fig. 4.

The measured $T_{\text{valid},i}$ values are used during the next communication cycle. This has two important consequences. First, the complete process needs a starting frame which does not transmit vital information rather than a start-up sequence. After the starting frame, the synchronisation time is reevaluated again in each single communication cycle. This allows to compensate for possible clock drifts due to environmental effects (temperature, tolerances,...). Furthermore, the measurements do not depend on each other, therefore no subsequent errors are covered as explained in the following.

For daisy-chained topologies, a possible source of errors would be the addition of phase errors from module to module. To add the errors up, a correlation between the measurements or the reference to one single measurement would be necessary. Since every modules is determining the global reference on its own based on own measurements and due to the daisy chain topology, inherited errors are not possible. Using the global reference synchronisation always leads to a maximal,

Fig. 7. Phase Error: The relative phase error between the internal clock signal of multiple individual modules in the daisy chain can lead to a maximal synchronisation error of $\pm \frac{T}{2}$

absolute synchronisation error of $\frac{T}{2}$.

3.4.3 Clock Phase Requirement A coarse synchronisation with an error of up to one clock period is possible by only using the global time reference requirement. Since the clock of the distinct modules are only synchronised in terms of the absolute frequency a phase error between two arbitrary modules is still possible as depicted in Fig. 7. The source of this phase difference can mainly be explained by the propagation delay between two modules being not a multiple of 2π but adding a certain amount $\Delta\varphi$ to the phase of the recovered clock signal of a module. Since the acquisition/observation of a common event by a module is only handled on a rising edge of the internal clock signal, an event then appears to occur not at the very same global point in time. Referring to Fig. 7, the desirable state of the individual clock signal would be the total alignment of all rising edges. An implementation including the phase shifting of each module's clock signal is under development and the expected synchronisation accuracy is in the range of 200 ps.

3.4.4 Synchronisation Accuracy Summarizing the three requirements explained above, the synchronisation can be divided into a coarse and a fine grained process. By only using the coarse grained and global reference based approach a minimal synchronisation accuracy of $\frac{T}{2}$ is achievable. The error does not add up from module to module since the measurements do not correlate. Thus, the accuracy is also independent to the number of modules. By additionally using the fine grained and phase shift based approach a much higher accuracy is achievable. Tests have shown that a synchronisation accuracy in the range of 200 ps while maintaining a reasonable duration for the calibration process is possible.

4. Hardware

In order to demonstrate the capabilities of the proposed field bus implementation, a prototype has been developed. The prototype is part of the custom-made high-speed communication and computing platform, shown in Fig. 8(a),

Fig. 8. (a) The developed custom FPGA board including the actual Altera Cyclone V GX FPGA, the transceiver interfaces (CSFP) and a clock jitter attenuator (Si5315). (b) Simplified schematic of the hardware and their interaction signals

which has been designed at HPE, ETH Zürich. The platform is based on an *Altera Cyclone V GX* FPGA which both runs the firmware of the bus system as well as all specific user-related computation and control tasks involved with the operation of the power electronic systems, for which the platform can be used. In the following, it is briefly explained how the physical layer hardware of the communication system has been designed with an emphasis on a compact realization and how the synchronisation of the individual clock domains is achieved.

4.1 Small Footprint Physical Medium Attachment

In order to keep the board space occupied by the communication hardware to a minimum, the internal multi-purpose high-speed serial transceiver PHY IP cores (cf. Fig. 8) of the Cyclone V FPGA are used in conjunction with a common *compact small form factor pluggable* (CSFP) fibre-optic transceivers. The CSFP module presents the physical dependent sublayer (PMD) and attaches directly to the respective inputs/outputs of PHY IP cores of the Cyclone V FPGA using logic-level high-speed differential-mode signaling. The driving of the optics for sending and receiving on the physical medium is handled by the CSFP module.

As opposed to the more widespread *small form factor pluggable* (SFP) transceivers (which work exactly the same way as the CSFP modules), CSFP modules incorporate two independent transceivers in the same form factor. This is achieved by sending and receiving data on different wavelengths of light on a single fibre. The two sockets for fibers of the CSFP module shown in Fig. 8(a) in fact belong to two independent data channels.

A block-diagram of the physical layer is shown in Fig. 8(b). The physical medium attachment (PMA) and the physical

coding sublayer (PCS) are provided by the PHY IP cores of the Cyclone V FPGA. This way, the space occupied by the physical layer hardware is minimized. Compared to the implementation presented in ⁽¹¹⁾, external physical-layer ICs are thus no longer necessary.

4.2 Synchronisation Hardware As explained in Sect. 3, the presented bus system uses the recovered clock information of the incoming bit-stream as a reference clock for each node, similar to the implementation proposed in ⁽¹¹⁾. Because the recovered clock is not immediately available at startup, the system boots up with a free running clock and switches hitless over to the recovered clock as soon as the recovered clock is available and stable. The base clock frequency of the transceivers is 125 MHz which is internally multiplied to the 1.25 GHz with which the data is transferred on the medium.

The switchover is handled by the Si5315 jitter-attenuator PLL. In the beginning, the Si5315 starts up with a free-running clock on each board (locclk). This clock is fed back to the FPGA as the reference clock (refclk) for the PMA and PCS as well as the user logic to initiate a link with the previous slave. As soon as the PMA and PCS of the PHY IP core of the Cyclone V FPGA have synchronized to the incoming bitstream of the previous module, the Si5315 is commanded (clkssel) to switchover to the recovered receive clock (recclk).

The switchover is performed hitlessly in incremental steps over consecutive clock periods to prevent a loss-of-lock. The use of an external IC like the Si5315 is necessary, because the internal general purpose PLLs of the Cyclone V FPGA are not specified for transparent clock switchover and do not meet the jitter requirements for clocking the PHY IP cores in this type of application.

5. FPGA Implementation

The proposed SyCCo-Bus protocol has been implemented on the FPGA using the hardware/components described in Sect. 4. The necessary steps to provide a running protocol implementation are described in Sect. 5.1 and Sect. 5.2.

5.1 Start-Up In steady state all slaves run on the master clock that is distributed via the bus. Nevertheless, the start-up of the slaves has to be executed based on the internal clock because the CDR does not work here yet. To suppress potential oscillations from the CDR during start-up, all slaves do not send any data, and therefore also no clock signal, to the next slave via their forward transmission path. The master continuously sends a filler frame ('Comma' ⁽¹⁴⁾). At the beginning of the start-up, the first slave synchronizes on the incoming clock recovered from the data received from the master. It then switches its Si5315 output to the recovered clock before taking its forward transmission path out of the reset state. Now, as the second slave is supplied with commas it can recover the first slave's clock which is equal to the master clock. By this mechanism, one slave after the other can synchronise on the master clock before the protocol and therefore the data transmission is started.

5.2 Protocol Implementation The complete protocol implementation is differentiated into two distinct modules, the master module and the slave module respectively. As depicted in Fig. 6 the master module only contains a single interface which is connecting it with the subsequent slave

Fig. 9. The slave module is illustrated in a schematic way to show the logical separation between the forward path (blue) and the backward path (red)

module chain. In the following, the basic structure of the slave modules is explained. Note, that the master modules structure is similar to the slave modules structure but includes some more control logic since it has to initialise the communication rounds.

The modules are differentiated into two separate paths, the forward transmission path and the backward transmission path as illustrated in Fig. 9. Since the transceiver on the FPGA module itself will detect an incoming frame due to the preamble structure the actual slave/master modules can rely on a valid signal ($RxFrameVFw_SI/RxFrameVBw_SI$) indicating that the incoming frame is valid. This signal is issued by the transceiver. Each module then contains a forward and a backward control instance which reacts on this particular valid signal. The control modules are responsible to read the current byte received from the frame. For the different input bytes, as explained in Fig. 4, the status and control signals of the controller modules output changes. It is especially worth to mention the $SFDFwDetect_S/SFDBwDetect_S$ signals which trigger the synchronisation process by starting and stopping a counter mechanism, respectively. The data exchange module is responsible for the read and write processes on the bus. It is combined with an output multiplexer which is associated with the control module to decide whether the input data frame or the new data gathered from the module has to be sent. The decision is based on the state of the current input frame since a slave module is only allowed to write to its dedicated space in the frame. The data exchange module in the backward path is optional as the data does not necessarily have to be exchanged again in the same communication round. Therefore, a simplified version of the data exchanger can be used. As a central unit to both paths the synchronisation timer triggered by the two start and stop signals, $SFDFwDetect_S$ and $SFDBwDetect_S$, issues the $SlaveDataValid_SO$ signal. The $SlaveDataValid_SO$ is representing a common time token on every module as described in Sect. 3.4. Hence, it is associated with the waiting time $T_{valid,i}$.

6. Measurement Results

The measurements have been done using different bus configurations with one master and up to eight slave modules

Fig. 10. Exemplary configuration of the SyCCo-Bus with eight slaves and one master module

Fig. 11. The measured synchronisation accuracy evaluated for a data set of $2 \cdot 10^3$ trigger points. The signals have been triggered for $V_{OH} = 2.4$ V. Note that this capture is only exemplary. The distribution of the accuracy in the proposed ± 4 ns depends on the phase shift as explained in Sect. 3.4.3

as depicted in Fig. 10. In particular, the synchronisation accuracy has been evaluated to show that the synchronisation method is working as proposed.

The measured RTT describes the time used by the master to write a complete frame on the bus, send it to all participants and read it back from the backward path. This time leads to the data rate and the data rate per slave which is compared to a theoretical value, the maximum possible data rate.

6.1 Synchronisation Accuracy The synchronisation accuracy has been measured by connecting the internal generated *SlaveDataValid_SO* signal to an output pin to observe it with an oscilloscope. Since this signal serves as the common global time token for each module over the whole bus, the falling/rising edges observed at the output pins of the different participants are expected to occur at least in the range of one clock cycle (± 4 ns). Figure 11 shows the synchronisation accuracy measured on a bus configuration with eight slaves and one master. The results show that the current implementation of the bus protocol allows an accuracy of ± 4 ns between the different modules. The measurement has been repeated several times to confirm the shown results, while the configuration has been altered to eliminate potential dependencies between the modules. Also, a long term test with 14 hours run time and more than $4.3 \cdot 10^6$ measurement points has been conducted. The test showed, that the signals, and therefore the internal clock signals, are drifting in the range of less

Table 1. RTT for different system configurations

Bytes/Slave	2 Slaves	4 Slaves	8 Slaves
8	0.984 μ s	1.904 μ s	3.172 μ s
16	1.112 μ s	2.152 μ s	4.224 μ s
32	1.368 μ s	2.664 μ s	5.248 μ s
64	1.880 μ s	3.688 μ s	7.296 μ s
128	2.896 μ s	5.728 μ s	11.384 μ s

than 100 ps. It is important to mention that the synchronisation accuracy inside the stated range depends on multiple factors as mentioned in Sect. 3.4. This leads to a phase difference between the different synchronisation signals which has shown to be non-deterministic. Note, that this phase differences eventually represent the phase differences between the internal 125 MHz clocks of the different modules. On the other hand, the long term test showed that almost zero additional phase shift due to clock drifts and environmental reasons has to be expected and therefore further corrections of the phase are only necessary in larger time intervals.

6.2 Communication Delay The communication delay has been evaluated in terms of the systems RTT. The RTT is taken as a bare measure to evaluate the delays on the various different participants. The measured RTT for different bus configurations are listed in Table 1. For a further investigation of the individual delays occurring on the bus, the contributors influencing the RTT are included in Eq. (2).

$$\text{RTT} = 2 \cdot N_{\text{Slave}} \cdot (T_{\text{Trans}} + T_{\text{Prot}}) + T_{\text{Clk}} \cdot (N_{\text{Byte}} \cdot N_{\text{Slave}} + N_{\text{Header}}) \dots \dots \dots (2)$$

T_{Prot} and T_{Trans} describe the protocol internal and physical delay from one slave module to another module, respectively. Since the size of the complete frame matters, N_{Slave} , the total number of slaves and N_{Byte} , the number of bytes per slave, have to be taken into account. Additionally, N_{Header} is the protocol overhead due to the preamble, the SFD and the SCB octets (cf. Sect. 3.2).

The VHDL implementation results in T_{Prot} being equal to only one FPGA clock cycle (8 ns), such that all values except for the T_{Trans} are known. In addition those values are adjustable. Whereas, T_{Trans} can only be measured since it includes the physical transmission time via the fibre optical cable and the processing time inside the provided PHY IP and the CSFP modules. The transmission time T_{Trans} has been found to be in the range of 186 - 189 ns, whereas the physical transmission time in the fiber occupies $\approx 0.5\%$ ($n_r = 1.444$, $length = 0.2$ m).

In Table 1, one can see that the RTT is almost but not completely doubling, while leaving the number of bytes per slave constant but increasing the number of participants in the system by a factor of two. This is, because of the previously mentioned header which acts as a constant offset to the RTT. On the other hand, leaving the number of slaves in the system constant but doubling the number of bytes per slave shows that exactly the added bytes per slave have to be processed.

6.3 Data Rate The maximum possible data rate is determined by the underlying link speed but limited by the protocol header. By defining the header and the number of interframe gap bytes (N_{IFG}) one can calculate the data rate achievable on the physical link itself. This can be done by:

Fig. 12. A Comparison between the *Fast Communication Case* and the measured *Low Memory Case* data rate: The calculated data rate (blue, green and purple lines) assumes that no gap between two distinct frames is necessary. In fact the protocol will need a certain gap between each frame to reset the every FSM, clean the memory and to make sure that data will not be overwritten. The red, orange and yellow line show the impact of the IFG in the current implementation. Aiming towards a small time gap between two subsequent frames will lead to a substantial increase in necessary memory since every incoming data transmission needs to be stored until its individual *SlaveDataValid_SO* occurs

$$\text{Data Rate} = \frac{N_{\text{Slave}} \cdot N_{\text{Byte}} \cdot f_{\text{Link}}^{-1}}{N_{\text{Header}} + N_{\text{Slave}} \cdot N_{\text{Byte}} + N_{\text{IFG}}} \dots \dots (3)$$

To calculate the maximum data rate N_{IFG} has to be equal to zero, meaning the transmission medium is always occupied. Based on the frame structure proposed in Fig. 3.2, the data rate has been calculated and is depicted in Fig. 12. The calculated value is referenced as *Fast Communication Case* since no additional interframe gaps (IFG) are inserted. This leads to the maximal achievable data rate but also to a much higher memory consumption on the slave modules since multiple data sets received over the forward path, which are not yet set valid, must be stored in the meantime. Since the data received from the previous frame can not be set valid on any module until the frame specific *SlaveDataValid_SO* signal occurs, the data needs to be latched. If the IFG should be smaller than the period of the *SlaveDataValid_SO* signal, additional memory or registers are necessary to latch more than one single data set. The actual measurements are based on a minimal memory configuration referenced as *Low Memory Case*. In this case, only as much data is buffered in the single modules as will be set valid as soon as the next *SlaveDataValid_SO* occurs. Hence, a comparison with the data rate calculated from the measured RTT with the theoretical values of the RTT for $\text{IFG} = 0$ has been done as well in Fig. 12. The comparison between the *Fast Communication Case* and the *Low Memory Case* shows, that the influence of the modules' individual time delay and the IFG is not negligible. Furthermore, one can see that the data rate increases with increasing frame size and converges to the physical link speed for the *Fast Communication Case*. The more bytes per frame are sent over the channel, the more negligible is the header and a potential IFG.

As indicated in Sect. 3.2, an increasing amount of slaves results in a longer RTT. In Fig. 12 no obvious difference between the different configurations is noticeable as the graph depicts the throughput of the complete bus.

Fig. 13. Data rate per slave: An increasing number of slaves leads to a decreasing data rate per slave, since more participants on the bus will lead to a larger frame. This larger frame eventually results in a longer RTT which means that the individual slaves need to wait longer until they can issue the *SlaveDataValid_S* resulting in a larger IFG

A further interesting and important metric is the data rate per slave, since this is eventually determining how many communication rounds are necessary to complete a message. Figure 13 shows, that an increasing amount of participants will result in a data rate reduction of each individual slave. To counteract a decreasing data rate, it is possible with the presented hardware to run two SyCCo-Bus systems in parallel, as the master still has one free CSFP port. This can significantly reduce the RTT, because the transmission delay introduced per slave (390 ns) is much larger than the additionally generated overhead (80 ns). It can thus be assumed that the parallel operation would save approximately half of the RTT.

Of course, one would like to have both bus systems to be synchronous as well. This can be achieved by two SyCCo-Bus masters that are implemented on one FPGA on the Master FPGA Board exchanging their measured RTTs (cf. C_i in Eq. (1)) such that the difference in their cycle time is known as Δc . The one having the lower counter value has to wait for $\Delta c/2$ cycles before starting to send the next frame, such that the *SlaveDataValid_SO* signal (cf. Fig. 9) is synchronous on all slaves in both bus systems.

6.4 Application To Example MMC System Considering again the example shown in Sect. 2, one can now check, whether the stated requirements can be met by the proposed bus system. Using Eq. (2) the RTT with the given amount of data and the number of slaves is

$$\begin{aligned} \text{RTT} &= 2 \cdot 90 \cdot (189 \text{ ns} + T_{\text{CLK}}) + T_{\text{CLK}} \cdot (9 \cdot 90 + 9) \\ &= 42 \mu\text{s}. \end{aligned}$$

Therefore, a duration of $100 \mu\text{s} - 42 \mu\text{s} = 58 \mu\text{s}$ is left for controller computations, which is sufficient for the considered example. Furthermore, the synchronisation accuracy proposed throughout this paper exceeds the requirements for the exemplary system from section 2.

7. System Suitability Evaluation

In its entirety, the SyCCo-Bus has been designed for a specific application, namely modular power electronic systems. Limitations and drawbacks can occur due to the static frame length or the need for a daisy chain topology. Note, that the performance as well as the specific characteristics strongly

depend on the chosen use-case. To use this particular system and evaluation of the most limiting properties is advised.

7.1 Use Cases The example given in Sect. 2 and evaluated in Sect. 6.4 denotes the low memory case since only the RTT is considered. The system can be configured such that it holds for other cases trading off the data rate and additional memory. The additional memory is necessary to buffer the data until the respective valid signal can be issued on all modules simultaneously. The following list names the marginal cases for which the system can be used.

- **Low Memory Case:** If direct feedback from the slaves is required before the master can transmit new commands to the slave modules, no additional memory is necessary. A possible example would be a periodic and alternating dialog between the master and the slave modules. This means that the master module needs to wait until the response of the slave modules has arrived resulting in a high number of IFGs (waiting cycles). The data rate is then limited by the system latency / RTT (cf. Eq. (2)) and mainly depends on the number of slave modules used.
- **Fast Communication Case:** If direct feedback is not required, the master module can send multiple independent frames without waiting for feedback from the slaves and vice versa. Hence, it is possible to optimally occupy the communication medium. In this case, the number of necessary intermediate memory slots for the required data rate needs to be found.

7.2 Interframe Gap (IFG) Estimation An important factor of the decision between the above mentioned use-cases is the useful occupation of the communication medium. It can be described by the number of IFGs used between two subsequent frames as explained in Eq. (3). Introducing IFGs increases the time the system is not transmitting useful data and hence decreases the overall data rate. On the other hand, working with less or even without any IFGs (continuous communication) will increase the necessary intermediate memory slots on the slave modules itself. An intermediate memory slot can only be released after the last slave module has successfully received the respective part of the frame. Figure 14 shows the relationship between the system waiting time, which corresponds to the amount of necessary interframe gaps, and the master writing time, which directly corresponds to the actual frame size. The master writing time T_{write} can be determined as

$$T_{\text{write}} = \underbrace{(N_{\text{Header}} + n \cdot (m + 1)) \cdot T_{\text{clk}}}_{\text{framelength}} \quad (4)$$

Note, that T_{write} strongly depends on the number of slave modules and the respective number of bytes per slave. $T_{\text{valid},0}$ denotes the period until the last slave has received the complete frame, seen from the perspective of the master. It can be calculated by means of Eq. (1). It is necessary to distinguish between three cases whereas the write time is smaller, larger than or equal to the time until the valid signal occurs.

- $T_{\text{write}} < \frac{T_{\text{valid},0}}{k}$: As soon as all intermediate memory slots are filled, holding data waiting to be acknowledged by the slave modules, the master module has to stop sending new data on the bus since it could not be stored (Overflow).
- $T_{\text{write}} > \frac{T_{\text{valid},0}}{k}$: The valid signal will occur before the

Fig. 14. Interframe gap evaluation: The figure shows a possible example for the evaluation of a system with n modules, m bytes per module in the frame and $k = 2$ intermediate memory blocks per module. As can be seen, the number of necessary interframe gaps/waiting time is related to the total number of memory blocks available

point in time when the master module has finished writing new data to the bus. Hence, a memory slot will be free to use for new data. This case shows optimal usage of the communication medium since only payload (neglecting meta data, headers, ...) is transmitted, but suboptimal usage of the available memory.

- $T_{\text{write}} = \frac{T_{\text{valid},0}}{k}$: The valid signal occurs right when the master module has finished writing new data to the bus. Hence, a waiting time occurs only if $T_{\text{write}} < \frac{T_{\text{valid},0}}{k}$ and can be calculated with Eq. (5), where k denotes the number of memory slots available to buffer the payload.

$$T_{\text{wait}} = T_{\text{valid},0} - (k - 1) \cdot T_{\text{write}} \quad (5)$$

To ensure periodic sending it is suggested to add the respective fraction of the complete waiting time to the end of each communication cycle. Eventually, the necessary IFGs can be calculated using Eq. (6). Note, that the T_{valid} would have to be estimated or measured during an actual configuration run since the physical delay can not be properly calculated.

$$N_{\text{IFG}} = \frac{T_{\text{wait}}}{T_{\text{clk}} \cdot k} \quad (6)$$

Knowing the amount of IFG, which need to be added to each frame, leads to a sending frequency as

$$f_{\text{send}} = T_{\text{write}} + \frac{T_{\text{wait}}}{k} \quad (7)$$

Finally, the necessary memory effort per slave module can be calculated as

$$K = m \cdot k \quad (8)$$

where K denotes the complete assigned memory in bytes.

8. Conclusion

In this paper, the bus protocol for the Synchronous-Converter-Control-Bus-System has been presented. The proposed protocol with the novel static frame structure has been

shown to work synchronously in the range of ± 4 ns. Furthermore, the implementation of the core of the protocol has been shown and the necessary hardware has been evaluated and built. Eventually, the round trip time and data rate of different bus configurations have been measured and evaluated. A comparison of the implemented prototype system with a time optimal system has been shown. Moreover, the prototype system has been found to work for a real world application as the synchronisation accuracy of ± 4 ns and the data rate/round trip time are both exceeding the necessary requirements stated for a realistic MMC system⁽¹²⁾ (cf. section 2). Since the data rate per slave starts to drop the more slaves are introduced into the system possible countermeasures have been presented based on the already existing hardware.

List of Abbreviations

CDR	Clock Data Recovery
CRC	Cyclic Redundancy Check
CSFP	Compact Small Form-Factor Pluggable
FOC	Fibre Optical Cable
IFG	Inter Frame Gap
PCS	Physical Coding Sublayer
PMA	Physical Medium Attachment
PMD	Physical Dependent Sublayer
RTT	Round Trip Time
SCB	Slave Counter Byte
SFD	Start Frame Delimiter
SFP	Small Form-Factor Pluggable

References

- (1) G. Tsolaridis and J. Biela: "Interleaved Hybrid Control Concept for Multiphase DC-DC Converters", in *IEEE Energy Conv. Congress and Exp. (ECCE)* (2017)
- (2) M. Hagiwara and H. Akagi: "Control and experiment of pulsewidth-modulated modular multilevel converters", *IEEE Trans. on Power Electr.*, Vol.24, No.7, pp.1737–1746 (2009)
- (3) A. Hillers, H. Tu, and J. Biela: "Central control and distributed protection of the DSBC and DSCC modular multilevel converters", in *IEEE Energy Conv. Congress and Exp. (ECCE)* (2016)
- (4) S. Fuchs, S. Beck, and J. Biela: "High output voltage precision PWM for modular multilevel converters", in 19th Europ. Conf. on Power Electr. and Appl. (EPE) (2017)
- (5) S. Fuchs, S. Beck, and J. Biela: "Analysis and Reduction of the Output Voltage Error of PWM for Modular Multilevel Converters", in *IEEE Trans. on Industr. Electr.*, Vol.66, No.3, pp.2291–2301 (2019)
- (6) M.A. Parker, L. Ran, and S.J. Finney: "Distributed control of a fault-tolerant modular multilevel inverter for direct-drive wind turbine grid interfacing", *IEEE Trans. on Industr. Electr.*, Vol.60, No.2, pp.509–522 (2013)
- (7) P.D. Burlacu, L. Mathe, and R. Teodorescu: "Synchronization of the distributed PWM carrier waves for modular multilevel converters", in Int. Conf. on Optimization of Electrical and Electronic Equipment (OPTIM) (2014)
- (8) C.L. Toh and L.E. Norum: "Implementation of high speed control network with fail-safe control and communication cable redundancy in modular multilevel converter", in 15th Europ. Conf. on Power Electr. and Appl. (EPE) (2013)
- (9) —: "A high speed control network synchronization jitter evaluation for embedded monitoring and control in modular multilevel converter", in *IEEE Grenoble Conf.* (2013)
- (10) "EtherCAT Technology", <https://www.ethercat.org>, Accessed: 02-26-2018.
- (11) C. Carstensen, R. Christen, H. Vollenweider, R. Stark, and J. Biela: "A Converter Control Field Bus Protocol for Power Electronic Systems with a Synchronization Accuracy of ± 5 ns", in 17th Europ. Conf. on Power Electronics and Applications (EPE, ECCE-Europe) (2015)
- (12) A. Hillers and J. Biela: "Increased efficiency and reduced realization effort of DSBC and DSCC modular multilevel converters (MMCs)", in Int. Power Electr. Conf. (IPEC) (2018)
- (13) G. Tsolaridis and J. Biela: "Flexible, highly dynamic, and precise 30 kA arbitrary current source", in *IEEE Trans. on Plasma Science* (2018)
- (14) IEEE: "IEEE standard for Ethernet", *IEEE Std 802.3-2015* (2016)

Stefan Rietmann (Non-member) has received his B.Sc. and M.Sc. degree from the Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, in 2015 and 2017, respectively. During his Master thesis he investigated a System-on-a-chip (SoC) device for custom control applications in connection with a Modular Multilevel Converter project. In November 2017 he started the Ph.D. program at the Laboratory for High Power Electronics Systems (HPE) with ETH Zürich, focusing on a wide bandwidth and highly accurate optical current measurement system.

Simon Fuchs (Non-member) received the Dipl.-Ing. degree in electrical engineering with a focus on control and power engineering from the Technical University of Kaiserslautern, Germany in 2015. He worked as an intern at Siemens Energy Sector, Erlangen, Germany, dealing with the modelling and control of liquid cooling systems in 2014. In 2014 he studied at the NTNU Trondheim, Norway, focusing on Power Electronic Systems. Since November 2015 he is a Ph.D. student at the Laboratory for High Power Electronic Systems (HPE) with the Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, working on a Modular Multilevel Converter as a medium voltage active rectifier for the generation of DC test voltages.

André Hillers (Non-member) received his B.Sc. degree and M.Sc. degree (with distinction) from the Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, in 2009 resp. 2012, where he has majored in power electronics and minored in power systems. In 2018 he received his Ph.D. degree at the Laboratory for High Power Electronic Systems (HPE) at ETH Zürich. His research focused on the optimal design and hardware realization of highly efficient, highly compact modular power converters for battery energy storage systems connected directly to the medium voltage distribution grid.

Jürgen Biela (Non-member) received the Diploma (Hons.) degree from Friedrich-Alexander Universität Erlangen-Nürnberg, Germany, in 1999, and the Ph.D. degree from the Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, in 2006, both in electrical engineering. In 2000, he joined the research department, Siemens A & D, Erlangen, Germany, and in 2002, he joined the Power Electronic Systems Laboratory, ETH Zürich, as Ph.D. student focusing on electromagnetically integrated resonant converters. From 2006 to 2010, he was a Postdoctoral Fellow with the Power Electronic Systems Laboratory. Since 2010, he is an Associate Professor at the laboratory for High Power Electronic Systems (HPE) with ETH Zürich.