CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates*

Tobias Ziermann, Stefan Wildermann and Jürgen Teich
Hardware/Software Co-Design, Department of Computer Science, University of Erlangen-Nuremberg
{tobias.ziermann, stefan.wildermann, teich}@informatik.uni-erlangen.de

Abstract

As the number of electronic components in automobiles steadily increases, the demand for higher communication bandwidth also rises dramatically. Instead of installing new wiring harnesses and new bus structures, it would be useful, if already available structures could be used, but driven at higher data rates. In this paper, we a) propose an extension of the well-known Controller Area Network (CAN) called CAN+ with which the target rate of 1Mbit/s can be increased up to 16 times. Moreover, b) existing CAN hardware and devices not dedicated to these boosted data rates can still be used without interferences on communication. The major idea is a change of the protocol. In particular, we exploit the fact that data could be sent in time slots, where CAN-conform nodes don't listen. Finally, c) an implementation of this type of overclocking scheme on an FPGA is provided to prove the feasibility and the impressive throughput gains.

1. Introduction

Year by year, the number of electronic components inside cars increases. New *electronic control units* (ECUs) are not only dedicated to entertainment, but also for increasing safety. Advanced driver assistance systems, e.g., monitor the environment through various sensors, detect critical situations and take proper actions or at least warn the driver. More and more mechanical connections are replaced by electronic ones to save energy and increase comfort, e.g., steer-by-wire.

All these electronic devices need a way of exchanging information on a fast, reliable and robust way. Today, the most widespread communication system used is CAN [1]. Using a priority-based event-triggered communication mechanism, it is known to provide very short response times, as long as the working load is relatively low and high priority messages don't block low priority messages for a long time. The increasing number of electronic components boosts the workload of the bus and therefore the response time. This can lead to a degradation, where real-time capability is not given anymore. One possible solution is to add additional wiring harnesses. These additional buses, however, increase the costs and the weight of an automobile, which both is not desired. Another problem is that it is not always possible to split the electronic components to communicate on different types of buses, because complex gateways would have to be inserted for protocol conversion and routing, thus creating additional overhead and cost.

So, what can be done to achieve higher data rates while using the existing CAN hardware and communication protocol? Interestingly, there exist already some thoughts, for example, of applying intelligent scheduling schemes of the CAN messages to support higher traffic without missing deadlines. A simple analytical approach is presented, e.g., by Tindell et al. [10], while Klehmet et al. [4] apply the method of Network Calculus to predict the transmission delays. A further improvement has been reported through using dynamic priorities for messages, see, e.g., Zuberi and Shin [11].

Our solution presented here to cope with the increasing demand for higher throughput is to increase the data rate on a CAN bus. Just increasing the bit rate of a CAN network, however, is not possible as it seems, mainly because of the electromagnetic compatibility and hardware delays, but also because the medium access (MAC) method requires to have a certain bit length. Now, all the restrictions beside the MAC could be eliminated by using, for example, faster transceivers to reduce the delay or shielded cables to cope with electromagnetic interference. The MAC method nevertheless allows an increase of the bit rate during certain time intervals. The CAN+ protocol as introduced here analyses and uses exactly these time slots in order to enhance data rate without disturbing existing CAN-conform hardware. Thus, our technology and methodology is backward-compatible. We will show experimentally, that an increase of up to 16x the data rate of a conventional CAN standard (1 Mbit/s) is possible.

*Supported in part by FuE-Programm "Informations- und Kommunikationstechnik" IUK-0706-0003 in Bavaria, Germany

978-3-9810801-5-5/DATE09 © 2009 EDAA

The paper is organized as follows. Section 2 shortly introduces the CAN standard and protocol, and explains why there are transmission speed limits. Section 3 introduces the idea to insert additional information into data packets at controlled points of time in order to increase data rates. At the end of this section, the layout of the CAN+ protocol is defined. Section 4 describes the experimental setup, which was used to verify and quantify the CAN+ protocol. In Section 5, the achievable speed increase is analyzed and the effects of the configuration of the CAN conform nodes on CAN+ are described. Section 6 describes the feasibility of the approach by showing that real-time image data transmission is feasible by using the CAN+ protocol, while Section 7 concludes our work.

2. Controller Area Network

The 1991 published 'CAN Specification 2.0' [1] of the data link layer of the ISO/OSI-Model has become the de facto standard for fieldbuses in many domains, especially automotive.

2.1. Physical Layer

The physical setup consists of a terminated two-wire bus, where nodes may be connected in any order. Bits are represented by the Non-Return-To-Zero Method, where a logical '0' is represented by a voltage difference between the two wires and a logical '1' by no difference. The size of the voltage difference depends on the environment the bus is used. This bit representation results in a dominance of the logical '0', in CAN terminology called *dominant*, which overwrites the logical '1' (*recessive*), if both bits are pending. This could result in long zero or one sequences, where missing edges makes synchronization impossible. Therefore, bit stuffing is used, which inserts an inverted bit after five similar consecutive bits. This additional bit is filtered by the receiver.

The length of a bit is depending on the bit rate used, e.g. $1\mu s$ at 1Mbit/s. In practice, the signals are not perfect. It takes some time until the signal is stable and propagation delay also needs to be considered. Because of this, a bit transmission interval is divided into 8 to 25 time segments. After about $\frac{2}{3}$ (depending on the settings) of the bit time, the signal is sampled. Synchronization is done after each edge, by reseting the counter of the time segments. Additionally, the length of the bit time is adjusted, if the detected edge is offset by more than the predefined *Resynchronization Jump Width*, see [8] for more details.

2.2. Data Link Layer

CAN is a message-oriented protocol, where each node can read all the messages. There exist four types of messages. The data message is used for information exchange.

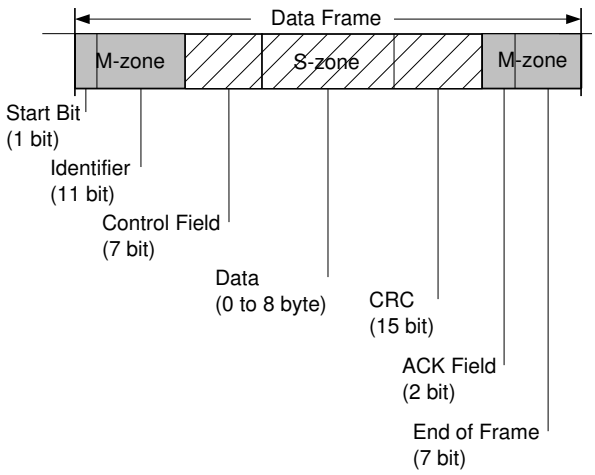

Figure 1. CAN data frame

The amount of data can vary from 0 to 8 byte. Its content is described by a unique identifier (cf. Figure 1). Besides the content description, the identifier also defines the priority by which access to the bus is granted. Bus access is decided by CSMA/BA (Carrier Sense Multiple Access with Bitwise Arbitration). The method of bitwise arbitration can be described as follows: Each node that wants to have access to the bus, starts sending its message, as soon as the bus is idle for 3 bit times. Every sent bit is also watched. When the sent bit differs from the watched, then a message with higher priority (lower identifier) also is pending and transmission is stopped. After sending the identifier, only the message with the highest priority is left and has exclusive bus access. This method ensures a non-destructive priority-based access, with up to 2032 priorities.

Beside this well designed access method, an advanced error handling scheme is probably the reason for the success of CAN: As soon as any node detects an error, it sends an error message which consists of six consecutive dominant bits. This sequence is unique and can be immediately detected. There are three states in which a node can be, depending on the number of errors counted: error active, error passive and bus off. Error active means the node is working as intended. Nodes in error passive state still monitor the bus traffic, but don't send error messages and defer to error active nodes. A bus off node is not allowed to have any influence on the bus.

2.3. Reason For Speed Limit

The 'CAN Specification 2.0' [8] itself doesn't limit the maximum bit rate, because this has to be specified in the physical layer. There are three main physical properties that influence the maximum data rate and bus length: output resistance, propagation delay and oscillator inaccuracy. Output resistance only limits the length of the bus, depending on the power used. Oscillator inaccuracy can be overcome,

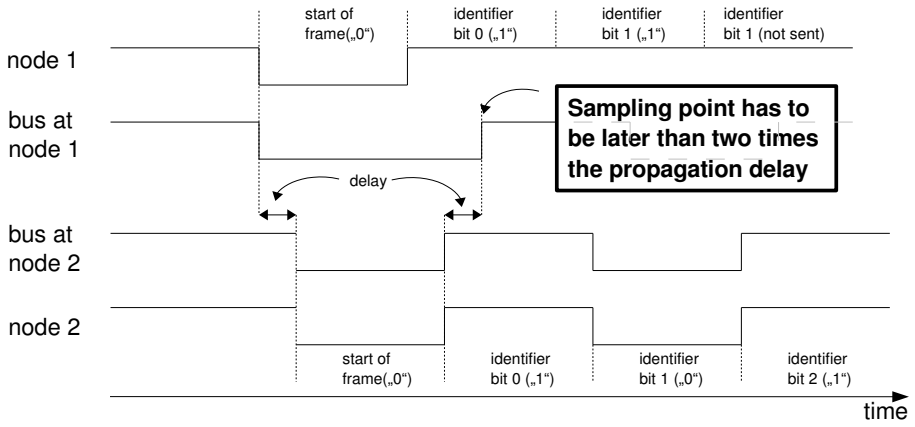

Figure 2. The impact of delay on the arbitration mechanism of CAN

even though it is more expensive, because better hardware could be employed. The propagation delay consists of two components: Signal delay on the cable and delay within the controller and transceiver.

The delay is the reason why the existing CAN protocol has bit rate limitations depending on the length of the bus. This limitation can be explained as follows: Errors occur, if during the arbitration phase a sending node cannot observe if it is overwritten until the end of the *bit time* (the time interval from beginning of transmission of a single bit to the end). That means the propagation delay is not allowed to be longer than the time from the beginning of the bit time to the sampling point. This gets clearer by looking at the example rate of 1 Mbit/s. Assuming the sampling point amounts to $\frac{2}{3}$ of the bit time (in this case 1 μs), the time from the start to the sampling is about 700 ns long. Assuming moreover a transmission delay of 5 ns/m on the cable and a controller and transceiver delay of 200 ns, the total delay within 30m is approximately $30\,\mathrm{m} * 5\,\mathrm{ns/m} + 200\,\mathrm{ns} = 350\,\mathrm{ns}$.

As can be seen in Figure 2, the worst case happens when node 2 synchronizes to the start bit of node 1. If the delay is too big, then node 1 reads the start of frame of node 2 as the first arbitration bit and would back off, even though it shouldn't at this point of arbitration. This simple example shows that the time between start of bit and sampling has to be at least two times the total delay, here 700 ns.

The delay of the controller could be minimized by using faster hardware, but the signal propagation delay on the bus is unavoidable. Therefore, the length of the bus significantly dictates the the speed limit.

3. The CAN+ Protocol

In the previous paragraph, it was shown that a general increase of the bit rate is physically not possible. As this speed limit has its roots back in the arbitration and acknowledgment phase, increasing data rates is only possible in those time intervals where it is ensured, that only one node is sending.

3.1 Previous Work

A first idea of overclocking a CAN bus in certain time intervals goes back to the idea of Cena and Valenzano [2]. They divided the CAN-message transmission interval in two areas: areas, where multiple nodes can transmit (M-zones) and areas where the bus is reserved by a single node (S-zones) as shown in Figure 1. In the M-zones, the normal data rate is applied. In the S-zones, an increase in data rate is suggested. Cena and Valenzano arrive at the conclusion that only a speedup of four times the normal speed is possible. This limitation is reached, because the higher data rate is used to decrease the transmission time instead of increasing the amount of data per message. Therefore, only the transmission time of the S-zone is reduced and an upper bound is quickly reached, because the M-zone dictates the message transmission time. On the one hand, their new protocol idea decreases the response time of high priority messages, because a higher priority message, which is ready to transmit has to wait less time until the lower priority message is finished. On the other hand, not increasing the amount of usable data transmitted per message limits the maximum data throughput. This can be seen in Fig. 3 where the usable data rate of a transmission with constant packet length is compared to a transmission with increasing packet length.

Another problem with this approach is that it is not conform with the CAN specification according to [8]. This means that using this protocol with normal CAN nodes is not possible, because the slow nodes will read errors and overwrite the bus with the dominant error messages. Cena and Valenzano suggest to use a reserved bit in the control field to flag those messages with higher data rate. This modification still requires a redesign of the current CAN controllers. This is not practicable for a standard, that is more than 16 years on the market. Beside the already mentioned problems, it has to be noted that no practical experiments were made on achievable data rates.

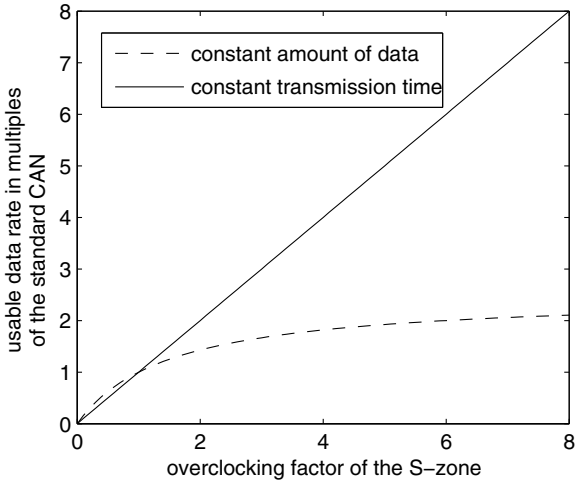

Figure 3. Increase of the usable data rate, when transmitting the bits in the S-zone faster

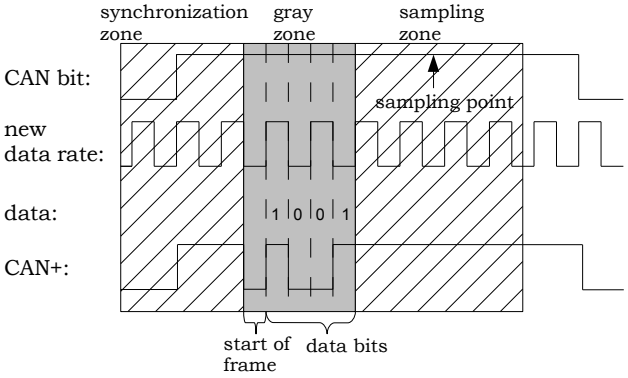

Figure 4. Transmission of a data bit for normal CAN and CAN+ protocol

3.2 Usable Gray Zone

Our CAN+ protocol overcomes the problem of incompatibility. To understand the new approach, a closer look at a single bit transmission has to be made. As can be seen in Figure 4, the CAN bit transmission interval is divided into three zones. The first one is the synchronization zone. This is where CAN-conform nodes expect an edge if a bit change happened. If edges in this zone are asserted at the wrong point of time, it will lead to an incorrect synchronization and subsequently to errors. The zone where the actual bit value is determined, is called sampling zone. In theory, sampling is done at a certain point of the bit, but experiments show a significant inaccuracy of this sampling time point. These two zones limit the gray zone. During this zone, the bus can take any value without disturbing the CAN conform communication. This unused region is used by the new protocol by inserting information at a higher data rate. The newly inserted bits will be called *overclocked bits* in the following.

3.3 Protocol within Gray Zone

Basically two approaches for transmitting the data within this small time slot are imaginable: Asynchronous or synchronous to the CAN-conform bit. An asynchronous protocol such as a Universal Asynchronous Receiver Transmitter protocol would mean that a message within the gray zone would be transmitted. The start of the message is not exactly defined, therefore a signaling bit must be used. The synchronous approach would send the information relative to the synchronization edge of the CAN protocol. The implementation is difficult, because the time from the synchronization edge to the start of the gray zone is relatively long compared to the bit length of the overclocked bits. Therefore, in this first implementation an asynchronous protocol as in Figure 4 is chosen.

3.4 Layout of the CAN+ Protocol

The new protocol behaves, in principal, as the CAN protocol, except that it transports additional data bits in the data field. A normal CAN data message contains up to 8 bytes. The amount of the additional data depends on the *overclocking factor* f. The overclocking factor indicates how many overclocked bits are inserted during the transmission of one CAN bit. The maximum number of overclocked bits is evaluated in Section 5. Hence, an overclocking factor of $f = 0$ means that the normal CAN protocol is used. Therefore, the total maximum amount of transmitted data per message is:

$$N_{total} = 64 \text{ bits} + f \cdot 64 \text{ bits} = (1 + f) \cdot 64 \text{ bits}$$

4. Implementation

In this section, we will prove the feasibility and correctness as well as robustness of the introduced CAN+ protocol by presenting a real hardware implementation on FPGA basis. Furthermore, we will analyze, using this implementation that overclocking factors of up to $f = 16$ are feasible.

A CAN node consists of a transceiver and a controller. The transceiver is used to convert the physical signal to a logical signal and vice versa. The controller is responsible for the rest of the protocol like synchronization, bit sampling and bit processing.

4.1 CAN-Transceiver

Ready-to-use CAN-transceivers are available in many varieties, from low-power [9] to fault-tolerant ones [7]. Commercial-off-the-shelf transceivers, however, couldn't be used, because they would not support the high data rates

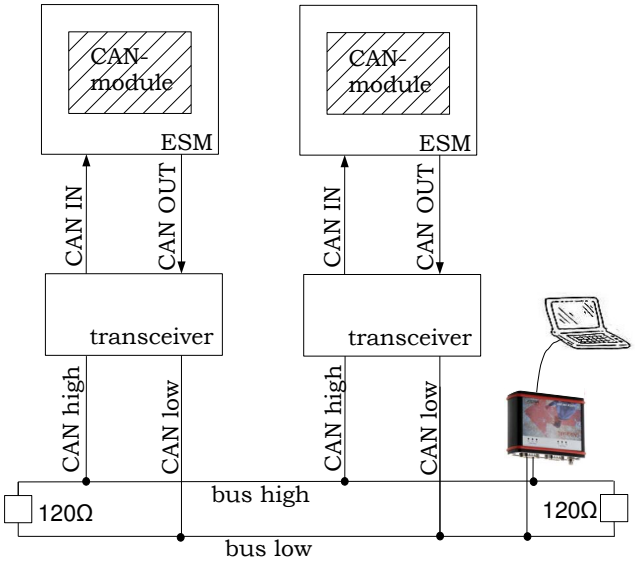

Figure 5. Schematic view of the test setup

for CAN+. Therefore, a new transceiver has been built that supports data rates up to 60 Mbit/s.

4.2 CAN-Controller

The controller is realized on an FPGA-based computer called Erlangen Slot Machine (ESM) [5]. It features a Xilinx Virtex II 6000 FPGA on which the functionality of the CAN controller has been implemented. Beside the FPGA, it has a great variety of connection possibilities, e.g., a video stream can be captured and displayed.

4.3 Test Setup

The setup consists, as can be seen in Figure 5, of two FPGA boards connected each to a transceiver. The transceivers are connected to the CAN bus. As representative of a typical Controller Area Network, the CANcaseXL [3] is connected to the bus. The CANcaseXL is used to evaluate if the CAN+ protocol is compatible to standard CAN controller. For the physical data transmission, a simple flexible flat cable (FFC) with D-Sub connectors has been used. The 1 Mbit/s CAN is always used for comparison.

5. Experimental performance analysis

Experiments with the previous setup were made to determine the length of the gray zone. To find out in which areas of the bit the CAN conform node isn't listening, an inverted signal has been inserted on different positions of the bit transmission interval, as can be seen on Figure 6. A stream including these corrupted messages is sent on

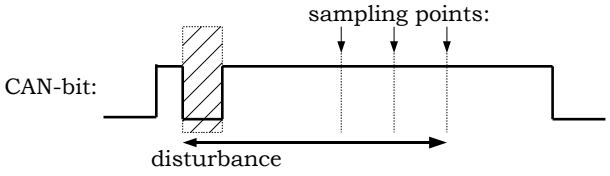

Figure 6. CAN-bit with inverted signal is sent to size the gray zone by testing the acceptance of the corrupted messages

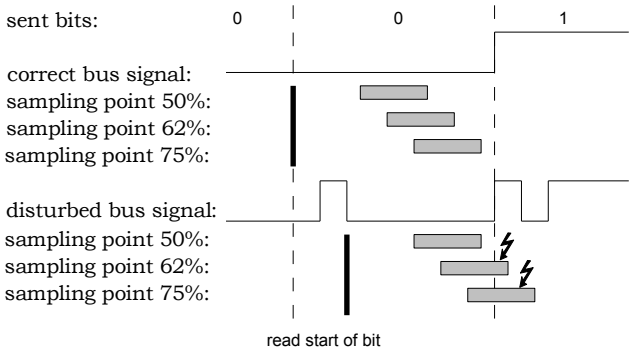

Figure 8. Effect of wrong synchronization caused by the inverted signal

the bus. The CANcaseXL, which uses the standard CAN controller Philips SJA1000 [6], reads the messages. If the CANcaseXL doesn't throw any error messages, the controller isn't noticing the inverted disturbance signal. The Philips SJA1000 has following configuration options:

- *Sampling Point Position*

  Time where the signal is taken (50%, 62% or 75%)
- *Synchronization Jump Width*

  Amount of adjustment per synchronization
- *Sampling*

  Single or triple sampling

In Figure 7 the usable gray zone for different sampling point positions can be seen. As described in 3.2, the synchronization zone and the sampling zone, which is mainly influenced by the sampling point position, limit the gray zone.

One exception is the outlier at a sampling point of 50%, where the inverted signal is in the synchronization zone. This causes the controller to synchronize to the inverted signal, if the normal CAN-signal doesn't provide an edge. The sampling zone is moved backward, as can be seen in Figure 8. When the sampling point is set at 50%, the sampling zone is still in the bit transmission interval. With a setting of 62% or 75%, respectively, the sampling zone is moved into the next bit transmission interval and the following bit is read instead. This effect is boosted by a big *Synchronization Jump Width*.

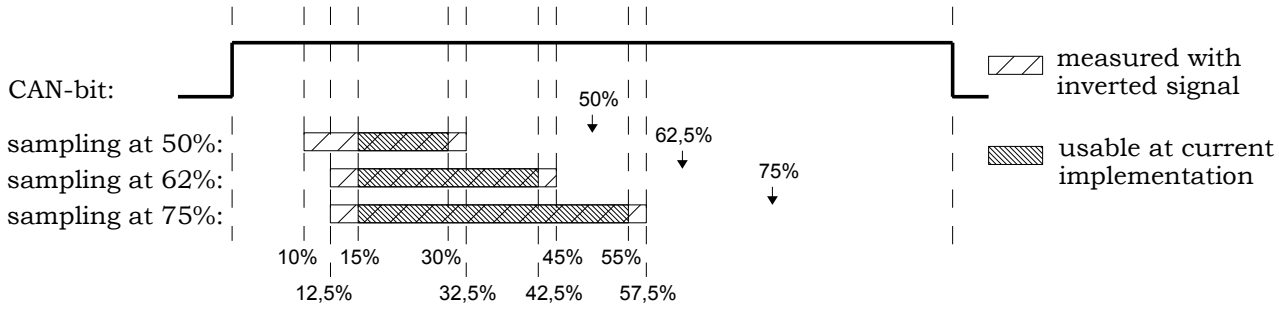

Figure 7. Analysis of the size of the grey zone with respect to the parameter Sampling Point Position

The *Sampling* setting does not have any effect on the results, because the two additional samples are taken with equal distance before and after the preset sampling point. If the sampling point lies on the inverted signal, at least one of the additional sampling points also lies on the inverted signal. Therefore, the majority reads the inverted signal.

In summary, the length of the gray zone typically ranges from 22,5% to 45% of the length of a bit transmission interval. However, only a part of the gray zone can currently be used for real data transmission, as can be seen in Figure 7. For our current implementation, there needs to be at least a delay of 15% from the start of the bit to the transmission of the high data rate bits. Additionally, when sending random data too close to the sampling zone, oscillations overshoot into the sampling zone.

This leaves a usable gray zone of about 15% to 40% of the length of a bit transmission interval. The bit length at 1 Mbit/s is 1 μs. So, the time for a data transmission ranges from 150 ns to 400 ns. With the current setup, a bit length of 25 ns can be used within the gray zone. Therefore, a maximum of $\frac{400\,ns}{25\,ns} = 16$ bit can be inserted. The used protocol within the gray zone has one bit overhead. This leads to a usable transmission rate of 16 times the normal CAN.

6. Video transmission case study

In order to show the functional correctness and robustness of the CAN+ protocol implementation and to verify the achievable transmission rate, a video application was implemented that makes it possible to send uncompressed video data over the CAN bus. Within the described experimental setting, a video stream taken from a webcam is fed into one of the two FPGA boards. On the first board, this video stream is packetized into CAN messages and sent over the bus. The second board receives and reconstructs the stream. With the available VGA-interface on the used FPGA board, the video is displayed on a monitor. Remarkably, videos from a webcam with 6,25 fps with a resolution of 320x240 grayscale may be transmitted without any errors due to the presented CAN+ protocol. Notable also that other CAN nodes may communicate without any interferences together with nodes using the presented overclocking scheme.

7. Conclusions

In this paper, a new method for enhancing the CAN protocol so to achieve up to 16x higher data transmission rates as compared to the conventional CAN standard has been proposed. This might make it possible to use CAN as the leading automotive protocol for the next decade. Since the proposed CAN+ protocol is backwards compatible, a mixed communication structure is possible, so that applications with low traffic demand still can use existing CAN technology. A prototype has been built to prove the applicability and to analyze the achievable performance.

At the moment, a very basic test setup has been used, which doesn't consider electromagnetic compatibility (EMC) or any disturbance handling. In future experiments these aspects will be considered.

References

[1] Controller Area Network (CAN). http://www.semiconductors.bosch.de/en/20/can/index.asp.
[2] G. Cena and A. Valenzano. Overclocking of controller area networks. *Electronics Letters*, 35(22):1923–1925, 28 Oct 1999.
[3] O. Falkner. The ideal Tool Chain. for LIN and CAN. From design to hardware and software integration. *Elektronik Automotive, LIN Special 2004*.
[4] U. Klehmet, T. Herpel, K. Hielscher, and R. German. Real-Time Guarantees for CAN Traffic. *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 3037–3041, 2008.
[5] M. Majer, J. Teich, A. Ahmadinia, and C. Bobda. The Erlangen Slot Machine: A Dynamically Reconfigurable FPGA-based Computer. *J. VLSI Signal Process. Syst.*, 47(1):15–31, 2007.
[6] NXP. *Datasheet SJA1000 Stand-alone CAN controller*, 2000.
[7] NXP. *TJA1054 Fault-tolerant CAN transceiver*, 2004.
[8] Robert Bosch GmbH. *CAN Specification Version 2.0*, 1991.
[9] Texas Instruments. *SN65HVD1040 datasheet*, 2008.
[10] K. Tindell, A. Burns, and A. Wellings. Calculating controller area network (can) message response times. *Control Engineering Practice*, 3(8):1163–1169, 1995.
[11] K. Zuberi and K. Shin. Scheduling messages on controller area network for real-time CIMapplications. *Robotics and Automation, IEEE Transactions on*, 13(2):310–316, 1997.