Fast Bit Compressor: A Novel Implementation of a Low Latency Modulation Using Binary Signals on Bandwidth Limited Links

Matthias Hinkfoth and Ralf Salomon

University of Rostock, 18051 Rostock, Germany
Email: matthias.hinkfoth2@uni-rostock.de

***Abstract*—Pulse-width modulation (PWM) is known for simple digital-to-analog conversion and motor current control. But up to now, PWM is not known for high speed communication. This paper shows how to implement the physical layer of a fully digital PWM-based communication system. For performance reasons the PWM senders and receivers are implemented using asynchronous logic. The proposed calibration procedure enables an adaption to a given channel bandwidth. A proof-of-concept is implemented on low-cost Cyclone II FPGAs. The experimental results regarding a single 1 m coaxial cable, are promising. An FPGA clock frequency of only 333 MHz is sufficient to achieve a data rate of 1 GBit/s, which is faster than every other communication means of this specific FPGA.**

I. INTRODUCTION

State-of-the-art field-programmable gate arrays (FPGAs) employ rather sophisticated hardware components, called serializer/deserializer (SERDES), to enable high-speed communication, High-performance FPGAs, such as Altera's Stratix 5 or Xilinx' Virtex 7 contain dedicated hardware SERDES blocks in hardware for off-chip and board-to-board communication. The vendors promise per-wire data rates of more than ten gigabits per second. The costs of these FPGAs vary between 1800 US-$ and 17800 US-$ for the Stratix 5 [1], and between 2500 US-$ and 39400 $ [2]) for the Virtex 7. It might thus be the case that the overall costs exceed the budget limits, especially in low-cost applications, such as home automation, low-power embedded systems, and other applications that do not have high-speed communication in their focus.

In order to exploit their full communication power, those SERDES transceivers also require high-bandwidth communication channels that allow for bit-serial transmissions with gigahertz frequency. Therefore, the high-frequency bit-serial transmission is not feasible in applications where the communication channel is constrained, since it transfers one single bit per time slot with all bits being of equal (time) duration. A low bandwidth simply does not allow the frequency to rise above a certain limit, and thus is limiting the maximum data rate. The data rate is therefore limited by the bandwidth of the cable and also by the clock speed of the synchronous receivers.

In order to provide a cheap and widely applicable solution to the problem of limited bandwidth, this paper resorts to a well-known technique called pulse-width modulation (PWM). In PWM, the sender considers a small number, e.g., $n = 3$ of bits as a single value, and encodes that value as the duration of one single pulse. On the other hand, the receiver decodes the transmitted value by measuring the duration of an incoming pulse. As the signal exhibits fewer transitions than a bit-serial data signal with the same data rate, it requires less bandwidth, and can be considered as a bit-compression mode. Section II discusses the relations between bandwidth, compression rate, and performance gain from a more formal point of view.

Synchronous PWM data transmission systems, which are normally used [6], are bound to their clock frequency, which constitutes an upper limit of the available processing speed. In order to overcome this limitation, Section III discusses a few asynchronously operating alternatives for the implementation of a PWM-based communication link. In particular Section III describes as simple pulse generation architecture that is based on asynchronous logic. Furthermore, Section III briefly reviews tapped-delay-lines (TDLs), which are also based on asynchronous logic, and which are wide spread in high-precision time measurement systems. With this combination, a PWM system would be able to generate and discriminate pulses in the sub-nano-second range.

The described concept was tested by means of a first prototype, which was implemented on two Terasic DE2-70 development boards. The details of this implementation as well as the experimental setup are summarized in Section IV. The results are presented in Section V. The practical validation included an unidirectional gigabit transmission across coaxial cables with lengths of 1 m and 2 m. A data rate of 1 GBit/s was achieved at a clock rate of 333 MHz.

Finally, Section VI concludes this paper with a discussion of reliability issues, possible enhancements, and options for future implementations of a full-fledged data-transmission system.

II. BACKGROUND: PULSE-WIDTH MODULATION FOR DATA TRANSMISSION

Varying the duty cycle of a square wave is known as pulse-width modulation (PWM). The duty cycle of the signal resembles a fraction of the fundamental pulse period. PWM

978-1-5090-1314-2/16$31.00 Public Domain

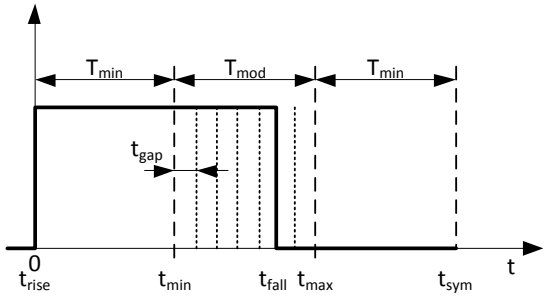

Figure 1. One symbol of a PWM-based data transmission comprises of a pulse and a dead time, which each have a minimum length of T_{min}. The symbol starts with the rising edge of the pulse denoted as t_{rise}. The middle part of the pulse is of duration T_{mod} and consists of n_{sym} slots of duration t_{gap}, which directly encodes the transmitted symbol value. The total duration of the pulse is constant, and amounts to $2T_{\text{min}} + T_{\text{mod}}$.

is widely used in motor control and light dimmers. In those applications, the pulses are smoothed, and thus provide a DC voltage corresponding to the duty cycle *dc*. When applying PWM for data transmission, the significant difference compared to analog applications lies in the fact that data transmission waives any smoothing circiutry. The data receiver is therefore able to precisely measure the time interval from the rising edge to the falling edge of every single pulse. Based on this fact, Schott [6] invented a digital PWM-based data transmission. He proposed a synchronous implementation that is still the state of the art. The transmission of pulse-width-modulated signals requires particular senders and receivers. The sender has to transform a bit value into a unique pulse width, whereas the receiver has to transform a time duration into the original bit value.

As an example, a communication line with a bandwidth of 5 MHz is given. Any single pulse cannot be shorter than 100 ns. As a result, the data rate R_{serial} of the bit-serial transmission is limited to $R_{\text{serial}} = 1\,\text{Bit}/100\,\text{ns} = 10\,\text{MBit/s}$. This is a fundamental barrier even though the sender and the receiver might clock faster, e.g. $f_{\text{clock}} = 200\,\text{MHz}$. Consequently, a transmission of the three bits $\boxed{1}\boxed{0}\boxed{1}$ takes 300 ns. In contrast, the PWM benefits from the higher clock frequency, by compressing the 3 bits into a single pulse, as shown in Figure 1.

In order to ensure to not exceed a certain bandwidth, the signal provides a stable high level for a duration T_{min} equal to the minimum pulse width in bit-serial communication. The very same duration T_{min} holds for the trailing low level at the end of the symbol.

Between the two T_{min} parts the duration T_{mod} is available for the actual pulse-width modulation. During T_{mod}, i.e., within the modulation phase, the falling edge of the pulse occurs at t_{fall}. The occurrence of t_{fall} is unique to every symbol. For example, $t_{\text{fall}} = t_{\text{min}}$ specifies the first symbol, $\mathtt{S}_0$. Then, the next symbol, $\mathtt{S}_1$ is specified by $t_{\text{fall}} = t_{\text{min}} + t_{\text{gap}}$ with t_{gap} specifying the amount of time that is required to securely differentiate different symbols. The duration of the inter-symbol gap t_{gap} depends on the hardware platform that implements the communication system's transmitter, its pulse generation module, as well as the decoding module on the receiver's side. In case of the bits $\boxed{1}\boxed{0}\boxed{1}$, $t_{\text{fall}}(5)$ is at the fifth possible time within T_{mod}, which corresponds to 25 ns with the assumed clock of 200 MHz.

An implicit property of the chosen symbol shape is a maximum bandwidth requirement of $B = 1/(2T_{\text{min}})$ in case the falling edge happens at t_{min}. As long as $t_{\text{min}} < t_{\text{fall}} < t_{\text{max}}$ holds, the bandwidth requirement is even lower. In other words, the bandwidth for transmitting the signal shown in Figure 1 is lower or equal to the bandwidth required to transmit square waves with a signal period of $2T_{\text{min}}$.

The total symbol length is $T_{\text{sym}} = 2T_{\text{min}} + T_{\text{mod}}$, and T_{mod} is used to encode the data by modifying the signal's duty cycle. Since three data bits require eight different symbols, $T_{\text{mod}} = (n_{\text{sym}} - 1) \cdot t_{\text{gap}} = (8-1) \cdot 5\,\text{ns} = 35\,\text{ns}$. Compared to 300 ns for the bit-serial transmission, $T_{\text{sym}} = 235\,\text{ns}$ implies a compression ratio of $235\,\text{ns}/300\,\text{ns} = 78\%$.

It should be clear that keeping t_{gap} as low as possible maximizes the system's throughput. In contrast, the effect of the number of symbols n_{sym} on the data rate is a little more complicated. If the designer has chosen a certain number of different symbols n_{sym} to fit into one pulse, the overall symbol length T_{sym} calculates to $T_{\text{sym}} = (n_{\text{sym}} - 1)t_{\text{gap}} + 2T_{\text{min}}$. The knowledge of the symbol length allows for calculating the achievable data throughput R. Since the data rate usually refers to the transfer of bits $b = \log_2(n_{\text{sym}})$ rather than on symbols, R can be written as $R = b/T_{\text{sym}}(b)$. According to $n_{\text{sym}} = 2^b$ and $T_{\text{sym}} = (n_{\text{sym}} - 1)t_{\text{gap}} + 2T_{\text{min}}$, R equals

$$R = b/(t_{\text{gap}}(2^b - 1) + 2T_{\text{min}}). \tag{1}$$

In order to make some sense of this equation, the numbers from the previous example are used. In the example given, T_{min} is 100 ns, and as it is a physical constraint, it is fixed in its value. The value of t_{gap} depends on the sender and receiver implementation, and is assumed to be 5 ns. As can be seen in Figure 3, the data rate depends on the number of bits per symbol b, and more important, there is an optimum for b at about 4 bit per symbol. The achievable data rate nearly 15 MBit/s.

More important, better sender and receiver implementations can reach even higher data rates, on the same bandwidth limited communication line. E.g. devices clocked with 500 MHz can reach a data rate of nearly 20 MBit/s. There is, of course, a fundamental data rate limit depending on the signal-to-noise ratio. The PWM communication is susceptible to amplitude noise due to the finite edge steepness of the pulse.

Obviously, faster devices are more expensive, consume more energy, and nevertheless have fundamental upper clock frequency limits. As shown in Figure 3, a data rate of 30 MBit/s requires end-point devices operating at 5 GHz. As a solution, the next section will show how the appropriate

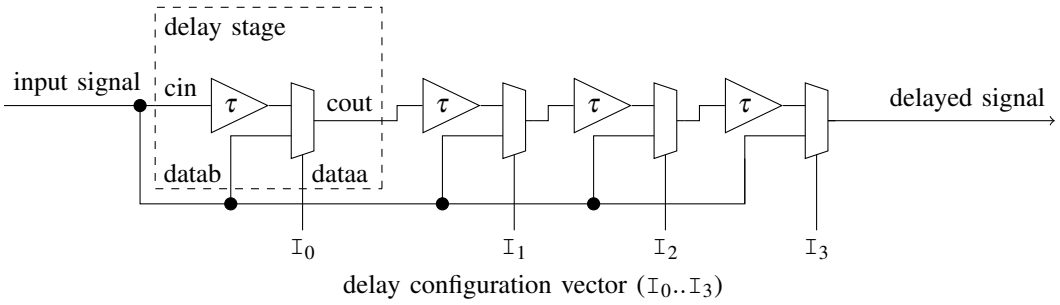

Figure 2. A configurable delay chain (CDC) implemented in the carry path of an FPGA. The selector signal `dataa` determines which incoming signal to pass to the output `cout`. The first input option is the carry-in signal `cin` of the logic element which is affected by a buffer delay τ. The second input option is an regular input `datab` which does not face any additional delays.

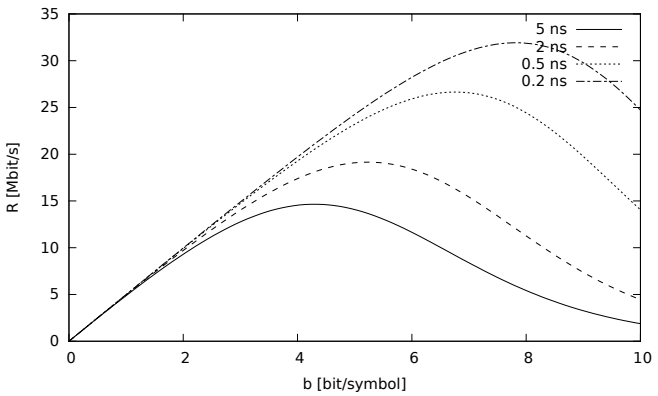

Figure 3. The graphs show the achievable data throughput as a function of the number of bits per symbol. The four graphs are parameterized with the sender's and receiver's time resolution, which are 5 ns, 2 ns, 0.5 ns, and 0.2 ns.

application of asynchronous logic can reach even better timing resolutions with only moderate hardware requirements.

III. Asynchronous Logic for Really Fast Time Measurement and its Use for PWM

The time resolution of synchronous logic is fundamentally tied to the clock period. When doing PWM data transmission, a high time resolution is crucial in order to achieve a high data rate. Therefore, asynchronous logic decouples the clock frequency limit from the timing resolution.

The sender has to transform a bit value into a unique pulse width. Costinett et al. developed an architecture suitable for general purpose FPGAs that is able to generate pulses with varying lengths [4]. A pretty similar approach is used in the pulse generation unit of the sender. Its basic block is a configurable delay chain shown in Figure 2. It comprises a sequence of buffers acting as active delay elements with multiplexers interleaved. Depending on the delay configuration vector ($\mathtt{I}_0$, $\mathtt{I}_1$, $\mathtt{I}_2$, $\mathtt{I}_3$), the delay chain activates or deactivates active logic gates, with each one imposing a delay of τ. Roughly speaking, the resulting delay is the sum of the involved multiplexers and the sum of the activated logic gates.

The internal FPGA structure might cause the following problem: On a Cyclone II FPGA, 16 look-up tables form one logic array block. Usually, global routing resources are placed between the logic array blocks. Within one logic array block, the delay between two logic elements amounts to about 130 ps. But several effects as crossing the border of a logic array block, drive strength variation in the flip-flop drivers, and non-uniform delays in the carry chain come into play. As a result the delay can increase to up to about 300 ps. These bigger delay steps are widely known as ultra wide bins [8], and they determine the worst case timing performance of classical single stage tapped delay lines. However, the communication system under consideration here can easily overcome this limitation. The transmitter simply employs a second configurable delay line in parallel to the first one, as shown in Figure 4. Now, one delay line acts as a delay for the falling edge, effectively widening the pulse, as the second delay line acts as delay for the rising edge of the input signal, and thus shortening the pulse. On the left-hand-side, the circuit is fed with a simple rectangular base clock that determines the pulse rate. The signal is fed into two distinct configurable delay line cdc_{rise} and cdc_{fall}. The PWM signal on the right-hand-side is $delay_{rise}$ `and not` $delay_{fall}$. Its pulse width t_i depends on the delay difference $t_i = delay_{fall} - delay_{rise}$ of the delayed-signal paths. In order to generate pulses with *varying lengths*, the delay Δt is configurable via the delay configuration vector $\mathtt{I}_0..\mathtt{I}_3$ and $\mathtt{I}_4..\mathtt{I}_7$ for both delay chains.

Presumably, only pulses that suite the communication channel and that can be properly distinguished by the receiver are

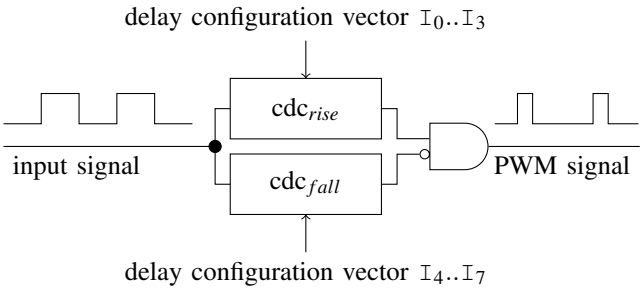

Figure 4. The realized pulse generation unit consists of two independently controllable delay lines, one for each pulse edge. As a result, the pulse width depends on the configuration vector ($\mathtt{I}_0$, $\mathtt{I}_1$, $\mathtt{I}_2$, $\mathtt{I}_3$, $\mathtt{I}_4$, $\mathtt{I}_5$, $\mathtt{I}_6$, $\mathtt{I}_7$).

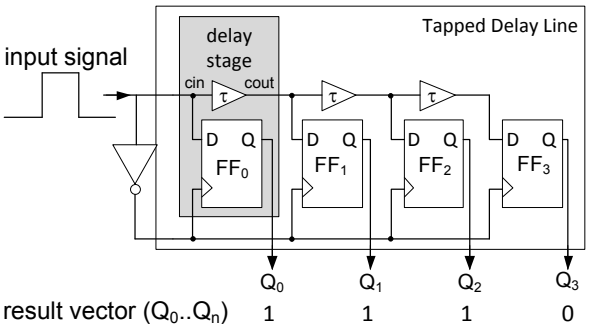

Figure 5. The duration of an incoming pulse can be determined quite accurately by tapped delay lines [5].

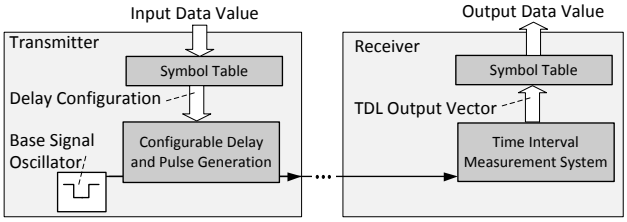

Figure 6. The transmitter generates a single output pulse with a pulse width according to a pare-defined symbol mapping. The receiver's time interval measurement system in combination with the symbol table decodes the pulse length into the output data value.

useful for the transmission. Obviously, a calibration procedure is required to select those pulses, and to also specify a meaning to them.

The task of the receiver is to determine the lengths of the incoming pulses for which tapped delay lines are a well-known option [5]. The general architecture of a tapped delay line is illustrated in Figure 5. It basically consists of a chain of structurally equivalent processing elements. Every processing element consists of a flip-flop and an additional delay element. In this paper, the incoming signal is connected to the data input of the first element, i.e., $\texttt{FF}_0$. After being delayed by the amount τ, it is fed to its successor flip-flop $\texttt{FF}_1$. From the figure, it can be seen that the set input of flip-flop $\texttt{FF}_\text{i}$ is activated at time $t_i = t_0 + i\tau$.

By contrast, the stop signal activates the clock input of *all* flip-flops at the same time. The term "at the same time" is slightly idealized due to the existence of internal (wire) delays, which is, however, way beyond the scope of this paper. To avoid synchronization problems between the two partners, i.e., transmitter and receiver, the receiver derives its clock signal from the incoming pulse. On the occurrence of the falling edge of the transmitted pulse, all flip-flops are triggered. The width of the arrived pulse can be directly determined from the number of ones of the flip-flops.

System Architecture: Both sender and receiver have to be embedded into a complete system. Both parts have to be enriched by appropriate tables. On the sender's side, a table transforms the three-bit value into a delay configuration vector ($\texttt{I}_0..\texttt{I}_7$). The delay vector is used by the pulse generation unit to generate the proper time duration.

Likewise, the receiver has to be enriched by a table that operates in the inverse direction: it derives the symbol value from the number of logical ones that is generated by the tapped delay line. The complete system architecture is summarized in Figure 6.

The chosen architectures for modulating and demodulating pulses have the following intrinsic advantages: (1) The delay of a simple logic gate, and thus the achievable resolution, is much better than the achievable clock rate of an FPGA's clock generator. (2) The required time delays are generated by only regular internal FPGA elements, and thus ensure operability on most (if not all) current FPGA families from different vendors. (3) Since both sender and receiver resort to the very same components, their behavior largely match, which makes fine tuning and calibration relatively easy.

Calibration, though, is still required to be able to fill the symbol tables with the appropriate values.

IV. High-Resolution Pulse-Width Modulation: Implementation

A. FPGA and Tool Chain

The first prototype was realized on Altera Cyclone II low-cost, off-the-shelf FPGAs on two Terasic DE2-70 development boards. The entire system uses the on-chip PLL that provides a 333 MHz square wave. The synthesis process was tested with Altera's Quartus 12.1 build 177 and Quartus 13.0 sp1. The entire system is described in VHDL, with Altera-specific restrictions and extensions. All the required timing constraints are embedded into the VHDL source code.

B. Utilizing Carry Paths

CDC and TDL require small delays. Inside FPGAs these delays are usually realized with carry paths between LUTs. It is not straight forward to actually use the carry path in an FPGA. This paper resorts to the explicit LUT specification. The internally stored logic vectors of the Cyclone II FPGAs are set with the `cycloneii_lcell_comb` primitive from the `cycloneii_components` VHDL-package. The following VHDL description yields one single delay stage of the configurable delay chain:

```
delay_stage: cycloneii_lcell_comb
   generic map (
      -- COUT <= B when A else CIN
      lut_mask => "1101100011011000",
      sum_lutc_input => "cin")
   port map (
      cin   => delayed_signal,
      cout  => to_next_delay_stage,
      dataa => selector,
      datab => undelayed_signal );
```

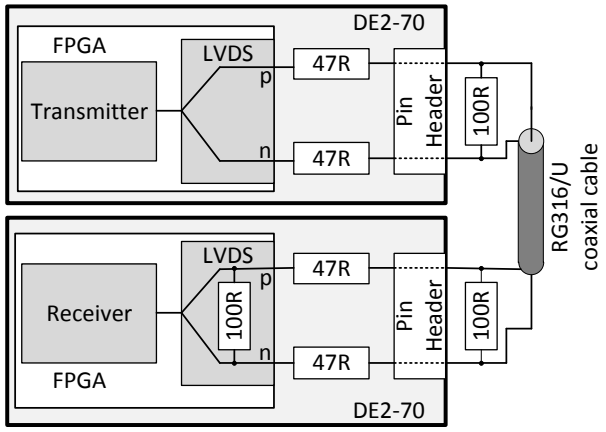

Figure 7. The electrical setup essentially consists of the two DE2-70 boards connected with a coaxial cable of 1 meter or 2 meter length.

The usage of the carry path is forced by connecting the `cin` and `cout` ports of the `lcell` to actual signals. The signal `to_next_delay_stage`, which is fed by the count is connected to the carry-in of the next LUT (set up in the same way). With this implicit carry chain, the synthesis tool will use the actual carry path of the FPGA. Therefore the lowest possible delay is achieved.

C. Pulse Generation Unit

As an approximation of Eq. 1, the system yields highest data rates with three to four bits per pulse. Thus, the transmitter implements two configurable delay chains, one for the rising edge of the signal, the other for the falling edge. Both delay chains contain 16 delay stages, each. On Altera's Cyclone II FPGA, the details are as follows: The carry path provides a resolution of 130 ps, slightly varying from stage to stage. Every stage of the delay line utilizes only one logic element.

D. Receiver's Tapped Delay Line:

The receiver's tapped delay line has to employ a certain number of stages, depending on the expected symbol length. At a clock frequency of 333 MHz, the symbol length is 3 ns. A 16-stage TDL with about 130 ps τ steps, covers a range of $16 \cdot 130\,\text{ps} = 2080ps$, which should be sufficient to cover the T_{mod} range. The tapped delay line consists of D-type flip-flops, an inverter, and carry path delays. To obtain a delay stage implementation for the receiver's tapped delay line, the VHDL code presented above has to be changed only slightly. The primitive's `lut_mask` is `1111000011110000`. and the output is connected to the logic element's flip-flop.

E. FPGA-to-FPGA Connection

In order to establish a communication, the symbols have to be transmitted over appropriate wires. The general outline is shown in Figure 7: Coming from the sender, a pair of LVDS-pins feed the signal to the on-board wires that connect to an on-board pin header. The pin header provides a connection to an SMA-adapter, and thus, to a coaxial cable. At the other end of the cable, the signals arrive at another SMA-adapter, a pin header, and an LVDS-pin before it arrives at the receiver module inside the FPGA. Sender and receiver are connected to each other with an RG316/U coaxial cable of 1 m or 2 m in length. Both ends of the wire are terminated by proper resistors.

Table I
THE TRANSMITTER'S SELECTED DELAY VALUES, THE RECEIVER'S RESULTING TDL VALUES, AND THE CORRESPONDING 3-BIT DATA COMPRESSED INTO ONE PULSE.

$delay_{falling}$	$delay_{rising}$	TDL-result	3-bit data value
3	1	5 or 6	0
4	0	8	1
5	0	9, 10 after warm-up	2
6	0	11	3
7	0	12	4
8	1	13	5
8	0	14	6
9	0	15	7

V. RESULTS

While switching through all possible pulse width values at the transmitter, the 16-step-TDL receiver observed the values 0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15. Apparently, the receiver distinguishes 11 different pulse widths. Pulses shorter than 5 TDL-steps, which is about 0.8 ns do not arrive at the receiver. From the remaining 11 pulses, eight pulses are sufficient for a compression of 3 bit into one pulse. The hardware-specific transmitter-delay combinations in table I are hand-selected, corresponding to 3-bit data from 0 to 7.

With about 130 ps spacing between symbols, and 3 bit per pulse at 333 MHz pulse rate, the prototype yields a transmission rate of $3\,\text{bit} \times 333\,\text{MHz} = 1\,\text{Gbit/s}$. Adding another 1 m piece of coaxial cable resulting in a 2 m connection between both FPGAs had no effect on the selected data values. The data transmission succeeded without recalibration.

The data path latency comprises four transmitter pipeline stages @ 333 MHz = 12 ns, about 2 ns FPGA-internal delay for both FPGAs = 4 ns, about 2 ns PCB delay for both PCBs = 4 ns, the wire propagation delay 1 m @ 2/3 c = 5 ns, and seven receiver pipeline stages @ 333 MHz = 21 ns. Which results in an overall latency of 46 ns for the 1 m-wire and 51 ns for the 2 m-wire.

VI. DISCUSSION

This paper has proposed to employ pulse width modulation (PWM) as a method for data transmission. PWM data transmission systems are particularly useful on a communication channel with tight bandwidth constraints. In comparison to widely used bit-serial transmission, PWM benefits from high timing resolution of the communication endpoints. In contrast to bit-serial transmissions, the data rate of PWM is not limited by the rise time of the signal edges, since PWM encodes multiple bits within a single pulse. Unlike other contributions [7], the implemented communication setup relies solely on digital hardware. It neither uses analog-to-digital converters, nor does

it use external analog components such as low-pass filters. Therefore, the proposed PWM setup is especially suited for an FPGA implementation, avoiding the cost of precise analog measurements. Furthermore, the avoidance of analog-to-digital conversion reduces the communication channels latency.

Previous digital PWM data transmissions [6] suffered from low data rates due to their synchronous modulation circuitry. This paper has employed asynchronous tapped delay lines, known from high-precision time measurement, for demodulating incoming pulses. Thereby, this proposed approach is able to increase the attainable data rate by an order of magnitude up to 1 GBit/s. This data rate is particularly impressive in view of the output pin bandwidth of the Cyclone II FPGA. According to its device handbook [3] (volume 1 page 5-46), the Cyclone II FPGA's bit-serial data transmission through LVDS-signals is limited to 640 Mbit/s.

Another advantage of a pulse-width modulation scheme is the ability to use differential signaling. The effect of external noise sources is essentially canceled out, resulting in a surprisingly good reproducibility of the experimental results. Instead, the main error source was the temperature-dependent delay variation of the internal buffer stages.

The prototype has been implemented on an off-the-shelf, low-cost Cyclone II FPGA development board. The FPGA implementation is configurable to adapt the PWM to given bandwidth constraints of the communication channel. The benefit over a bit-serial transmission is even higher at tighter bandwidth limitations. In the experimental setup, the complete communication system runs at only 333 MHz. Which is beneficial regarding both, energy consumption and circuit demand. Nevertheless, the overall latency is only about 50 ns.

The prototype presented in this paper lacks several features of complete communication devices. In order to reduce the bit errors for the measurements, the symbols where hand-selected. Future work will be including a forward error correction layer that is able to mitigate the effects of transient errors, noise, and voltage variations of the circuits. Recalibration at runtime, a feature known from some time measurement circuits, might help against long term effects, such as the temperature dependence of the asynchronous parts. Care must be taken, to not induce additional latency by the recalibration mechanism.

A time measurement device with a higher resolution might make more symbols distinguishable, even though the data rate only rises logarithmically with the symbol count. Nevertheless there might also be other benefits as a reduced number of pipeline stages that make it worth a try. Latency can be further reduced by encoding the symbol tables in look-up-tables instead of RAM blocks.

The implemented PWM communication system shows good performance while demanding few hardware resources and low power. The design requires only basic FPGA elements, and is thus portable across different FPGA architectures from various vendors.

REFERENCES

[1] Buyaltera.com. http://www.buyaltera.com/.

[2] Digikey electronics. http://www.digikey.com/.

[3] Altera Corp., San Jose, CA. *Cyclone II Device Handbook*, 2007. Altera Document CII5V1-3.3.

[4] D. Costinett, M. Rodriguez, and D. Maksimovic. Simple digital pulse width modulator under 100 ps resolution using general-purpose fpgas. *Power Electronics, IEEE Transactions on*, 28(10):4466–4472, Oct 2013.

[5] J. Kalisz. Review of methods for time interval measurements with picosecond resolution. *Metrologia*, 41(1):17–32, 2004.

[6] B. Schott. Method and apparatus for transmitting data words asynchronously from one microprocessor to another in form of timing intervals, August 1986.

[7] S. Suh. Pulse width modulation for analog fiber-optic communications. *Lightwave Technology, Journal of*, 5(1):102–112, Jan 1987.

[8] J. Wang, S. Liu, Q. Shen, H. Li, and Q. An. A fully fledged tdc implemented in field-programmable gate arrays. *Nuclear Science, IEEE Transactions on*, 57(2):446–450, April 2010.