I2C Hardware Master Serial Interface for Asynchronous ADCs

Wojciech Andrysiewicz, Dariusz Kościelnik, Marek Miśkowicz
AGH University of Science and Technology
{andrysie,koscieln,miskow}@agh.edu.pl

***Abstract*—The implementation of the I²C-compatible serial interface for asynchronous ADCs that output data irregularly in time is reported in the paper. The device contains I²C hardware master-transmitter functionality and is capable of operating fully autonomously in I²C bus communication system. In particular, the asynchronous ADC with proposed I²C interface is able to initiate data transfer on the bus when new data are available on the converter output.**

I. Introduction

During the last decade, a new class of analog-to-digital converters (ADCs) that produce the digital words on their outputs irregularly in time have been developed and reported in the technical literature [1]–[7]. These converters termed as asynchronous ADCs (AADCs) usually adopt the level-crossing sampling [1], [2], or time encoding [3]–[6] for input signal discretization. The digital words on the AADC output represent in general information on time intervals between level-crossings in the former, or between edges of pulse train in the latter. On the other hand, in the AADCs with self-timed charge redistribution reported in [7], the conversion can be triggered by sporadic external events. The AADCs are attractive for the use in applications with constrained energy resources (e.g., biomedicine).

The irregularity of producing digital data creates new challenges for design of ADC output interface especially when the ADC communicates with the other devices via synchronous buses. The Inter-Integrated Circuit (IIC, I²C) bus is the example of standard serial synchronous bi-directional two-wire bus used to interconnect low-speed devices in the communication system [8]. The I²C-compatible interface is a standard version of the ADC serial output. The conventional ADCs available on the market operate in the I²C systems as slaves under control of master devices, see for example [9]. A master device, which is usually a processor, polls the ADC periodically with the sampling frequency and enables a transfer of data available on the ADC output.

However, the conventional technique of polling the AADC by a relevant I²C master device is no longer valid since the external master does not know a priori the time instants when the data on the converter output become available. The only way to establish the effective link between the AADC and the other devices is to design the interface with the ability to initiate the transmission on the I²C bus. This means that the AADC has to act as a master in the I²C communication system. The concept of the I²C-compatible interface for AADCs was presented in [10] and [11]. In both references, the AADC acts as a hardware master which in particular means that the AADC interface contains hardware implementation of the I²C protocol for master devices (i.e., hardware for idle bus detection, an arbitration mechanism, clock stretching on the request of a slow slave device). This paper concerns the CMOS implementation of a I²C hardware master interface whose concept has been reported in [10], [11]. To reduce hardware overhead and power consumption, the presented implementation relates to reduced architecture and includes only the master-transmitter functionality of the I²C interface.

II. Interface architecture

The general architecture of the I²C interface for asynchronous analog-to-digital converters is presented in Fig. 1. The digital words outputted irregularly in time by the AADC and representing the samples of the analog input enter the I²C interface circuit via the FIFO buffer. The AADC uses a write enable signal (WE) to inform the FIFO on valid data. Writing the digital word to the FIFO is an asynchronous event which is triggered by completion of the actual analog-to-digital conversion cycle. As soon as the data are introduced to the buffer, the FIFO produces the READY signal to the control module (CM) that manages the I²C interface operation.

Before starting the transmission of the digital word on the data line (SDA) of the I²C bus, the CM checks if the bus is currently not occupied. Therefore, the serial data module (SDM) incorporates a bus sensing logic and reports when the bus is idle using the STSPin input signal from the bus to the device. If the bus is idle, the CM issues the STSPout signal to SDM, thus initializing an I²C-specific START condition on the SDA line. Next, the address of the slave device intended to receive the transmitted data is shifted out to the bus bit-by-bit followed by the command bit (R/$\overline{\text{W}}$ - read/write) using the multiplexer (MUX). After sending the slave address, the CM requests an acknowledgement (ACK) using the ACKout signal. The ACK procedure consists in that the master keeps the SDA line in the high state (ACKin signal) and expects the slave to pull the bus low. If the SDM senses the ACK on the bus, it responds with a ACKin signal. If the slave does not send the acknowledge bit, the CM produces the STOP condition using the STSPout signal. The same procedure of shifting out the bits and verifying the acknowledge bits is applied to the data octets. When all the data octets have been transmitted successfully, the CM issues the STOP condition to the SDA line and removes the current digital word from the FIFO by the use of the NEXT signal. The master devices in I²C communication system support the arbitration mechanism to resolve the collisions on the bus using the Bitwise Arbitration CSMA protocol [8].

978-1-4799-1920-8/15/$31.00 ©2015 IEEE

Another functionality required in the I^{2}C communication system is a response to clock stretching. This feature is used if the slave device is not ready during a transmission to respond to the master requests on writing or reading data. The I^{2}C clock line (SCL) is driven by an open-drain outputs and the master device is in charge of producing the clock signal using its open-drain output. However, even if the master releases the clock line (i.e. sets it to the low state), the slave device may decide to hold the clock line to the high state which results in slowing down the transmission rate.

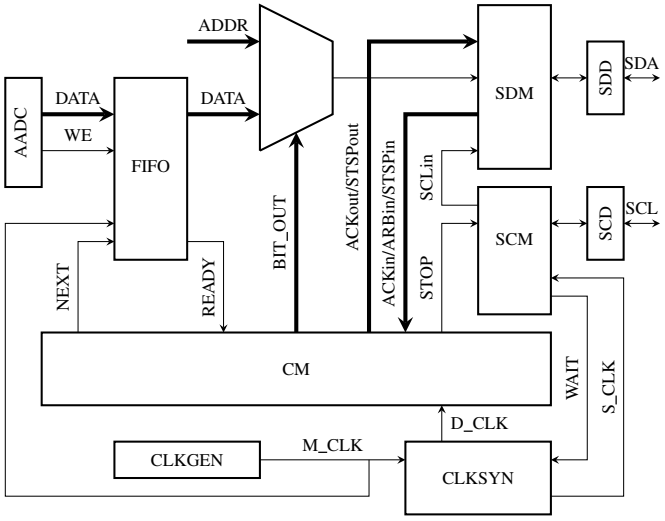

Fig. 1. Architecture of I^{2}C-compatible interface for asynchronous ADCs.

III. Implementation

A. State Machine

The control module (CM) which manages the operation of the I^{2}C interface of the AADC is designed as a state machine. To implement the I^{2}C protocol for master devices including bus arbitration, the CM is designed as an automata having 8 states (IDLE, START, ADDR, ACK_A, DATA, ACK_D, NEXT, NACK) as seen in Fig. 2.

The CM is in the IDLE state if the FIFO is empty (READY=0), and there is no transmission on the bus (BUS BUSY=0). As soon as a digital word is introduced to the FIFO (READY=1), the CM moves forward to the state START, which causes the START condition to be produced on the data line SDA. After initiating the START condition, the CM moves to the ADDR state when the address of the external slave device (aimed to be communicated with the AADC) followed by the R/$\overline{\text{W}}$ bit is shifted out on the SDA line. The device might lose the arbitration during ADDR state if the other master with lower slave address identifier transmits at the same time. If so, the device returns to the IDLE state. If there is no collision of messages on the bus, the CM is moved to the ACK_A state when the device receives the acknowledge bit related to the address octet. If the slave responds with the positive acknowledge bit, then the ACKin is set to one and then the CM enters the DATA state. Otherwise, the CM is introduced to the NACK state. In the DATA state, the data that contain the digital word produced by the AADC are shifted out bit-by-bit on the SDA line. The I^{2}C AADC device may also lose the arbitration during the DATA state. If so, the device returns to the IDLE state. Otherwise, the CM is moved to the ACK_D state when the device receives the acknowledge bit that confirms a successful reception of data octet. If the slave responds, then the ACKin is set to one, then the CM enters the NEXT state. Otherwise, the CM is introduced to the NACK state. The NEXT state is entered by the CM if the device has received the positive acknowledge bit from the slave device that confirms a successful reception of the data octet. In the NEXT state, the STOP condition is issued to the SDA line of the I^{2}C bus and the digital word is removed from the FIFO buffer (by activating the NEXT signal). If there is another digital word in the FIFO queue, it will be ready for the next transmission after visiting the IDLE state by the CM. The NACK state is entered by the CM if the device has not received the positive acknowledge bit. Then the STOP condition is produced on the SDA line but the current digital word is not removed from the FIFO making it ready for retransmission.

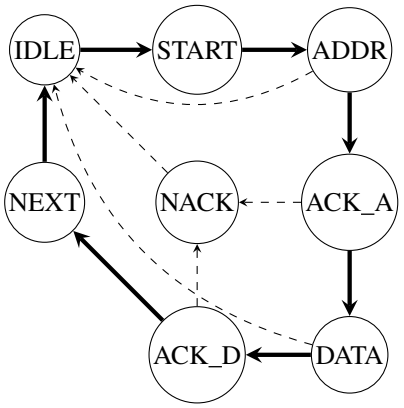

Fig. 2. Automata state diagram. Dashed lines refer to unsuccessful data transactions (*i.e.* lost arbitration or lack of acknowledgement).

B. Clock Generation

The clock generation block (CLKGEN) uses a RC circuit to provide the master clock (M_CLK). The M_CLK is running at 400 kHz and is used to clock the FIFO and synthesize other system clocks in the CLKSYN block. To support clock stretching, a receiving slave device must be able to hold the SCL line low at any time during data transmission. The master must wait for the release of the SCL line by the external slave device to continue data transmission. The clock stretching feature is implemented in the CLKSYN block using a two-bit counter clocked by the M_CLK. When CLKSYN receives the WAIT signal, it stops the clock generation. The D_CLK and S_CLK are frozen until SCL is released. However, the FIFO is then still clocked and capable of receiving new data from the AADC. It is also worth noting, that if there are no digital words waiting in the FIFO and no transmission in progress, the master clock generator (CLKGEN) is not needed and can be stopped to preserve energy if the AADC is idle. The CLKGEN must be restarted when new data arrive from the AADC to the FIFO buffer which is signalled by the WE signal produced by AADC.

C. Data Buffering

The AADC that delivers data to the FIFO buffer has no clock, which provides design conditions different from the typical FIFO applications where FIFO buffers transfer data between two clock domains. To optimize the design in terms of chip area and power consumption, the first-word-fall-through FIFO architecture (FWFT-FIFO) was adopted that is usually regarded as too expensive in terms of chip area for large buffers [12]. Nevertheless, the FWFT-FIFO architecture is suitable for the 8x8-bit FIFO buffer used in the I^{2}C interface of the AADC as it requires no multiplexing at flip-flop inputs and simple control. The FIFO architecture for the I^{2}C AADC interface is presented in Fig. 3.

The FIFO buffer is arranged as a shift register comprising of A_0,...,A_7 registers while each register A_n is 8 bits long. Each data word A_n is accompanied by a "validity" bit V_n. This bit indicates that the register stores a valid digital word. If the digital word in the register A_n is valid, while the word in the register A_{n-1} is not valid, the flip-flops in the register A_n are clocked once in order to shift them to the register A_{n-1}.

The design of a single register A_n in the FIFO buffer is presented in Fig. 4. These blocks are stacked as shown in Fig. 3 creating the FWFT-FIFO buffer architecture. The digital word id introduced to the FIFO input from the AADC by the WE signal. The WE allows the data to enter the FIFO and marks the data block corresponding to this word as valid (V_7). With each clock cycle, the data word and its validity bit V_n are propagated down the FIFO to the last empty (i.e. non-valid) position. If a digital word reaches the bottom of the FIFO (register A_0), the CM receivers the READY signal, which means that the word may be shifted to the I^{2}C bus. As soon as the digital word is successfully transmitted, the CM activates the NEXT signal, which is propagated up the FIFO allowing to shift data downwards.

Even if the FIFO is empty, a new digital word requires 8 clock cycles to propagate from the input to the output of the FIFO. The corresponding delay is referred as the fall-through time and regarded in general as a major drawback of FWFT-FIFOs. In the relevant context, this time delay is negligible as the fall-through time is $20\mu s$ whereas a single bit on the I^{2}C is transmitted during $10\mu s$.

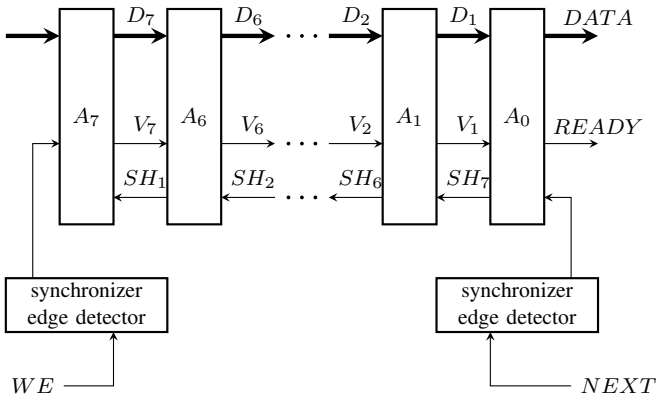

Fig. 3. FIFO buffer architecture with segments (compare Fig. 4) forming a controlled shift register.

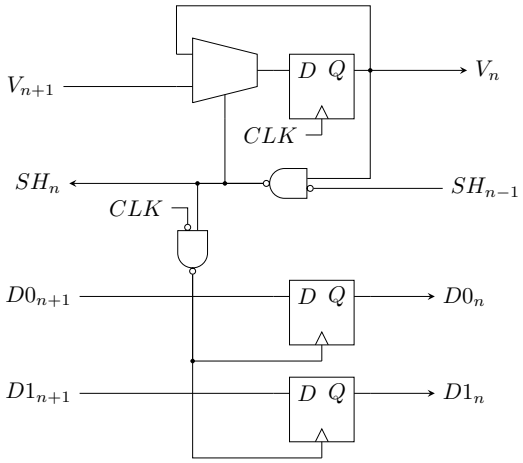

Fig. 4. Segment of FWFT-FIFO (A_n). Bits of data word (D0,D1) are accompanied with "Valid" (V_n) bit. Data bits and "Valid" bits travel down the FIFO. "Valid" bits determine the shift condition (SH) - whether the word should be shifted down or not.

IV. Verification

The I^{2}C interface of asynchronous ADC was implemented in silicon using UMC CMOS technology with the feature size $180\mu m$. The main device parameters based on simulation experiments are presented in Table I. The implementation was tested for compliance with the I^{2}C specification. In Figures 5 - 8, we present timings of the device operation in four cases described below.

CASE1: Fig. 5 shows how the master attempts to transmit a data octet 10110010 to a slave device addressed as 0011001. At first, the slave device acknowledges the successful reception of the address octet but does not acknowledge the data. The digital word is retransmitted and next acknowledged successfully but the slave device uses clock stretching. The delay between the signal WR and the start of the data transmission occurs as the digital word travels down the FIFO buffer.

CASE2: The hardware master is also capable of storing a number of digital words from the AADC and transmitting them when possible. In Fig. 6, three digital words arrive during short time interval (10110010, 10001011, 01111101) to the FIFO buffer. All the words are stored in the FIFO and consecutively transmitted over the I^{2}C bus.

CASE3: In this scenario, shown on Fig. 7, there are two active hardware masters on the bus. Master A starts transmission first (slave address: 0011001, data: 10110010), followed by master B (slave address: 0010100, data: 10001011). Because master A occupies the bus first, master B has to wait until the bus is free again. The additional plots are provided with the internal SDA signal of each master.

CASE4: This scenario is referred to verifying the bus arbitration in Fig. 8. Masters A and B transmit the same data as in CASE3, but both transmissions are triggered simultaneously. Both masters are unaware of the collision on the bus until master B wins arbitration on the fourth bit of the slave address. After detecting the collision, master A terminates the transmission, and retransmits when master B completes its

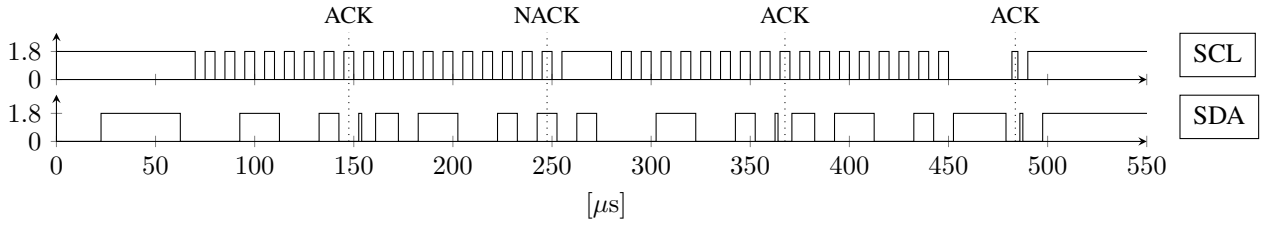

Fig. 5. Clock stretching and response to NACK. After an unacknowledged sample, the device retransmits the frame. During retransmission the slave is stretching the clock and the master waits for the ACK response.

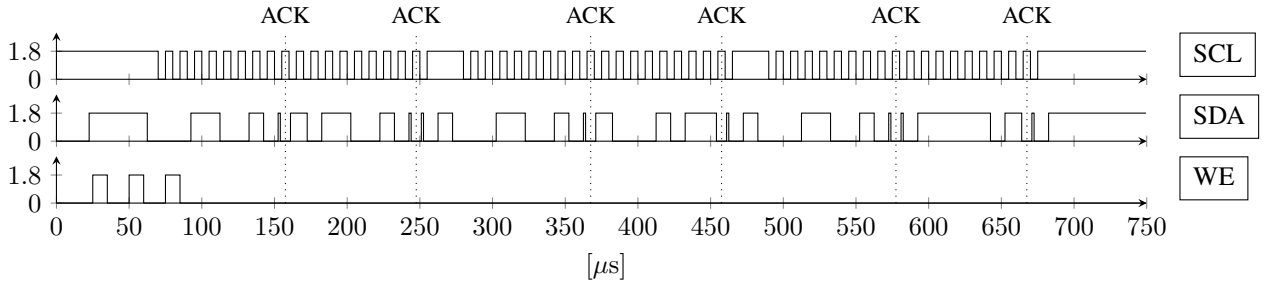

Fig. 6. Data buffering in FIFO. Three data words arrive in a short time interval. They are stored in the FIFO and sent in 3 consecutive frames.

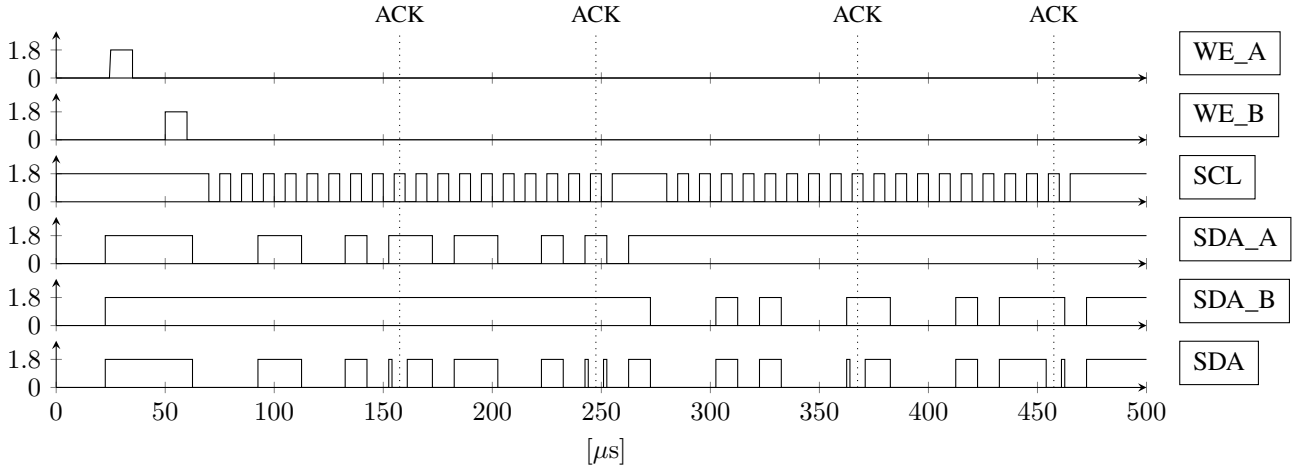

Fig. 7. Multi-master buss access. Master A receives begins transmission first. Master B is waiting for a free bus and starts its transmission when possible

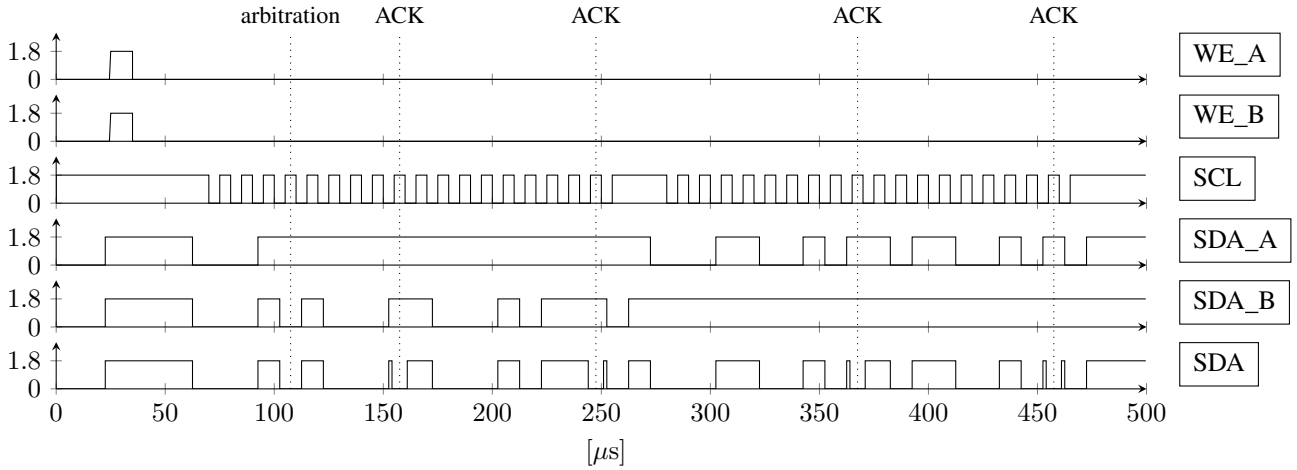

Fig. 8. Bus arbitration. Master A and Master B start transmission simultaneously. Master A loses arbitration due to higher slave address identifier, and terminates the transmission. When master B finishes, master A retransmits its frame.

Table I. SELECTED DEVICE INFORMATION

Operating voltage	$1.8V$
Average current consumption when transmitting	$5.9\mu A$
Bus clock frequency	$100kHz$
FIFO buffer width	$8\ bits$
FIFO buffer depth	$8\ words$
Estimated number of transistors (exl. flip-flops)	620
Estimated number of flops	95
Total estimated number of transistors	1760

transmission. The plots of internal SDA signals in both masters give a deeper insight into operation of both devices.

V. CONCLUSIONS

The CMOS implementation of the I^{2}C-compatible interface for asynchronous ADCs is presented in the present paper. The device contains I^{2}C hardware master-transmitter functionality and is capable of operating fully autonomously in I^{2}C bus communication system. Future work can address the design improvements related to further reduction of power consumption: FIFO clocking when idle at the price of slightly more complex shift logic, and a mechanism stopping the clock generator when device is idle.

ACKNOWLEDGEMENT

This work was supported by European Regional Development Fund, Operational Programme Innovative Economy under grant UDA-POIG.01.03.02-12-001/12-00

REFERENCES

[1] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin, "A new class of asynchronous A/D converters based on time quantization," in *Proceedings of International Symposium on Asynchronous Circuits and Systems*, May 2003, pp. 196–205.

[2] K. Kozmin, J. Johansson, and J. Delsing, "Level-Crossing ADC Performance Evaluation Toward Ultrasound Application," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 8, pp. 1708–1719, Aug 2009.

[3] D. Kościelnik and M. Miśkowicz, "Designing time-to-digital converter for analog-to-digital conversion," in *Proceedings of IEEE Workshop on Design Diagnostics of Electronic Circuits and Systems*, 2007, pp. 275–280.

[4] J. Daniels, W. Dehaene, M. Steyaert, and A. Wiesbauer, "A/D Conversion Using Asynchronous Delta-Sigma Modulation and Time-to-Digital Conversion," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 9, pp. 2404–2412, Sept 2010.

[5] L. Hernandez and E. Prefasi, "Analog-to-Digital Conversion Using Noise Shaping and Time Encoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 7, pp. 2026–2037, Aug 2008.

[6] D. Kościelnik and M. Miśkowicz, "Time-to-digital converters based on event-driven successive charge redistribution: a theoretical approach," *Measurement*, vol. 45, pp. 2511–2528, 2012.

[7] D. Kościelnik and M. Miśkowicz, "Event-driven successive charge redistribution schemes for clockless analog-to-digital conversion," in *Design, Modeling and Testing of Data Converters*. Springer, 2014, pp. 161–209.

[8] NXP Semiconductors, *I2C-bus specification and user manual.*

[9] Texas Instruments, *ADC121C021 Datasheet, 2013.*

[10] D. Kościelnik and M. Miśkowicz, "Designing asynchronous analog-to-digital converter with I2C interface," in *IEEE Conference on Emerging Technologies in Factory Automation*, Sept 2009, pp. 1–4.

[11] M. Miśkowicz and D. Kościelnik, "Interface for communication between sensing devices and I2C bus," 2014, US Patent 8,868,812.

[12] D. Norris, "Propagating FIFO storage device," 1989, US Patent 4,805,139.