Pulsed Decimal Encoding for IoT Single-channel Dynamic Signaling

Shahzad Muzaffar and Ibrahim (Abe) M. Elfadel

Khalifa University of Science and Technology
Masdar City, 54224, Abu Dhabi, UAE
{shahzad.muzaffar,ibrahim.elfadel}@ku.ac.ae

Abstract. Pulsed-Index Communication (PIC) is a recent technique for single-channel communication that is based on the principle of transmitting the indices of only the ON bits as a series of pulse streams. In this paper, a modified version of PIC, called Pulsed Decimal Communication (PDC), is presented that uses the same underlying principle but with key improvements in data rate and reliability. Like PIC, PDC is a protocol for single-channel, high-data rate, low-power dynamic signaling that does not require any clock and data recovery. It consists of a three-step algorithm, comprising a segmentation, an encoding, and a sub-segmentation step to achieve higher data rates. The segmentation step splits the data word into smaller segments and therefore smaller decimal numbers to represent them. The encoding step reduces the number of ON bits in the data and relocates them to lower indices. The sub-segmentation step further splits the segments into smaller sub-segments. The complete process significantly reduces the total number of pulses required for transmitting binary data, thus improving the data rate by about 78% . A theoretical model of the PDC protocol is exploited to estimate its data rate and derive the optimum segmentation. Furthermore, PDC is shown to be more reliable than PIC as it eliminates the variations in the number of symbols to be transmitted. The FPGA and ASIC ($65nm$ technology) implementations of PDC show that the low-power operation and small footprint of PIC are preserved. PDC consumes around $25\mu W$ of power at a clock frequency of $25MHz$ with a gate count of approximately 2150 gates.

Keywords: Dynamic Signaling, Single-Channel, Low-Power Communication, Clock and Data Recovery, Internet of Things, Automatic Protocol Configuration, Pulsed-Decimal Communication, Pulsed-Index Communication

1 Introduction

The basic structure of an Internet of Things (IoT) application is that of a two-tier architecture in which the first tier acts as a gateway to the Internet of People (IoP) and has privileged cloud access [8] while the second tier is made of the remaining sensing and instrumented devices that are connected to the gateways

using simple, low-power physical links and protocols. An example of such two-tier architecture can be found in smart homes where a range of electronic devices pertaining to lighting, air-conditioning, security, fire protection, as well as several other common electronic household items, need to be controlled remotely by a smartphone or computer. Some of these devices can be selected as gateways while others remain on the edge to communicate with the gateways. Single-channel protocols are very promising to achieve the connectivity between the first and second tiers as they provide easy and scalable networking options. The main requirements of such protocols are low-power for long battery life, high data rates for reliable transmission during bursts of activity and small form factor for lowering silicon implementation costs. Unfortunately, the existing protocols fail to meet these requirements simultaneously. Protocols providing high data rates, such as WiFi, WLAN, TCP/IP, USB, etc. [2, 1, 4], are power-hungry and involve complex controllers to handle two-way communications. On the other hand, low-power protocols such as 1-Wire [5], UART [3], etc. have low data rates.

To fill up the gap, a new single-channel communication technique, called Pulsed-Index Communication (PIC) has recently been introduced ([6, 7, 10]). Its fundamental novelty lies in the replacement of the transmission of *data bits* with the transmission of *ON bit indices* in the packet stream. PIC achieves very low power consumption as it does not require any clock and data recovery (CDR). It also achieves high data rates due to a novel encoding scheme that guarantees error-free decoding even in the presence of clock discrepancies between transmitter and receiver. Additionally, PIC offers a flexible and scalable single-channel networking option for the connectivity of devices in a variety of IoT use cases [8]. Though PIC has shown very competitive figures of merit, it does have few disadvantages. One of them is that PIC is prone to complete packet failure, especially when there is a corruption in either the ON bit index number or in the number of ON bit indices. At the root of this packet failure is the fact that two adjacent data words can have vastly different *numbers* of ON bits or significant differences in the *locations* of the ON bits.

The objective of this paper is therefore to present a modified PIC protocol that significantly improves packet transmission reliability while maintaining its core idea of transmitting information in the form of pulse streams instead of transmitting data bits. The novel protocol, called Pulsed Decimal Communication (PDC), is based on transmitting a number of pulses which is equal in count to the decimal number represented by the data. For long data words, it is intuitive that such transmission will need a large number of pulses. To reduce the represented decimal number, and hence the pulse count, PDC uses a three-level data segmentation and encoding algorithm with the twofold goals of reducing the decimal value of the bit stream *and* controlling the number of pulses per data segment. The three levels of the algorithm are:

1. *First-level segmentation*: This is similar to PIC and is meant to break the data word into smaller data segments to split a number into smaller decimal numbers.

2. *Encoding*: This is applied to each segment and is meant to further reduce the number by *relocating* the ON bits to the least-significant part of the segment.
3. *Second-level segmentation or sub-segmentation*: This step is applied to each of the encoded segments. Each sub-segment is made of 4 bits. This step is meant to equalise the number of pulses in each first-level segment and thus improve transmission reliability.

Like PIC, PDC is dynamic in that it can accommodate several data rates in the range of 4.87 - 12.9 Mb/s with an average of 7.33 Mb/s using a 25-MHz clock. The main advantage of PDC is that it generates a fixed number of symbols per data word, which results in a simpler and more reliable transmission with respect to packet failures. Other important side advantages of PDC with respect to PIC include a simpler packet header, a more compact decoding, and an improvement of about 78% in data rate. Furthermore, ASIC synthesis of the PDC protocol, using GLOBALFOUNDROES $65nm$ process, has shown that it is well within the power and area envelopes of PIC.

This chapter is a significantly expanded version of [11] with the following important additions:

1. We have introduced a new example to better explain the PDC protocol. See Subsection 3.6]
2. We have added a detailed PDC performance analysis and derived the PDC optimum segment size. See Subsection 4.3.
3. We have added a new treatment of the PDC tolerance to clock jitter and its robustness with respect to clock rate discrepancies between transmitter and receiver. See Subsection 4.5.
4. We have clarified the power management aspects of PDC and commented on the additional power savings that can be achieved through pulse width control. See Subsection 4.7.
5. We have added a discussion on automatic PDC parameter setting at the power-on phase in a realistic master-slave network context. See Subsection 4.6.
6. We have expanded Section 5 on Experimental Results and added several benchmarking charts and tables.

2 Review of PIC

Pulsed-Index Communication (PIC) [10] is a recent single-channel protocol that does not require any circuitry for clock and data recovery (CDR). The protocol is based on the concept of a pulsed index where instead of transmitting the bits themselves, only the indices of the ON bits in a data word of length B bits are transmitted. These indices are encoded as pulse counts. The core of the protocol is to encode these indices so as to *minimize* the number of ON bits, and *move* them to the Least Significant Bit (LSB) end of the packet. The encoding process includes a segmentation step where the data is broken into N independent segments of size l bits each (i.e., $N = B/l$). To maximize data rate,

PIC uses, on each segment, an encoding combination of bit inversion and/or segment reversion/flipping. This combination is meant to reduce the number of ON bits and decrease their index values so as to lower the number of pulses required to transmit the data bits. To facilitate decoding, flag pulses representing the type of encoding performed (i.e., inversion (1 pulse), reversal (2 pulses), both (3 pulses), or none (4 pulses)) are added to each segment. All the pieces of information including flags, number of indices, and the indices themselves are transmitted in the form of pulse streams. The pulse is characterized by its width which is the number of clock cycles during which it remains high. Within a given packet, segment pulse streams are separated by an inter-symbol delay, α, as shown in Fig. 1. The receiver counts the number of pulses for each pulse stream and applies the decoding according to the flags received. The data rate depends on the number of clock cycles required to transmit a packet. Beside the elimination of CDR, the implementation of PIC is very area-efficient, low-power and highly tolerant of clocking differences between transmitter and receiver. As empirically found in [10], PIC is dynamic and provides data rates in the range of $3.1 - 8.5\ Mb/s$ with an average of $4.1\ Mb/s$ operating at a clock rate of $25MHz$ and transmitting data words of 16-bits each. For an ASIC implementation using $65nm$ technology, PIC can reduce area by more than 80% and power by more than 70% in comparison with a CDR-based serial bit transfer protocol.

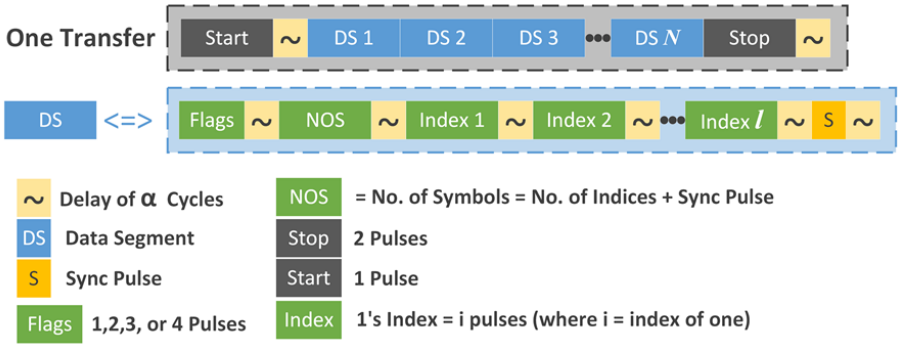


Fig. 1. PIC Packet Format

3 Pulsed Decimal Communication

Like PIC, the core concept of PDC is to break the data packet into smaller segments, and to transfer these segments as series of pulses. The number of transferred pulses of a given segment is equal in count to the decimal number represented by the segment bits. Hence the name of this technique: Pulsed Decimal Communication (PDC). High-data rates are achieved by a three-step

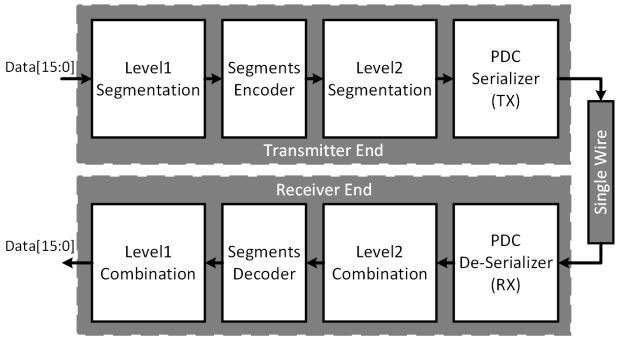


Fig. 2. Conceptual Block Diagram of PDC

encoding process of the raw bit stream so that the segments have decimal number representations that are as small as possible. The three-step encoding process comprises three operations: first-level (L1) segmentation, encoding (inversion, and/or reversal), and second-level (L2) segmentation. The conceptual block diagram of the encoding and transmission processes is shown in Fig. 2, whose blocks are explained in the following subsections.

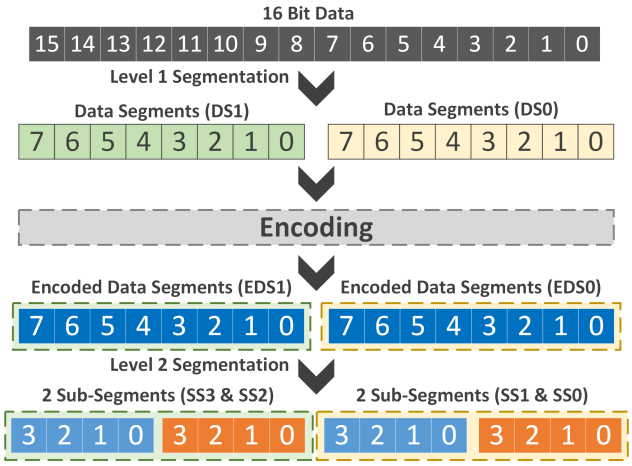


Fig. 3. PDC Three-Step Encoding Process

3.1 First Step: L1 Segmentation

The decimal number represented by the data word increases exponentially with the number of bits. The count of pulses to represent a number becomes large and

it becomes infeasible to transfer this count in pulses. Therefore, segmentation is required to break a data word into several independent segments with lower number of bits, hence representing smaller numbers. It is mentioned in [10] that a segment size of 8 bits optimizes the number of delays for PIC to increase the data rate. For PDC, 8 bits is not an optimum segment size and the segments need to be cut down further before the pulses can be transferred. On the other hand, increasing the number of segments before encoding would decrease the data rate. This is because each time a segment is encoded, flags are generated to represent the type of encoding applied. The process of encoding will be explained in the following subsection. The encoding flags are also transmitted in the form of a separate pulse stream, and an increase in the number of segments increases the number of flag streams to be transferred, which would of course reduce the data rate. Additionally, there is an increased area and power overhead associated with the encoding of each of the segments. Therefore, it is not feasible to break the data word into a large number of segments before the encoding is applied. To address this, we have introduced two levels of segmentation, pre- *and* post-encoding as shown in Fig. 3. In the first segmentation level, data is broken into segments called *Data Segments (DS)* each of which is 8 bits long. One could think of applying encoding on a 16-bit data word instead of performing the first level of segmentation, but it has been experimentally observed that encoding 8 bits is far better than encoding 16 bits in order to reduce the decimal number represented by the segment bits.

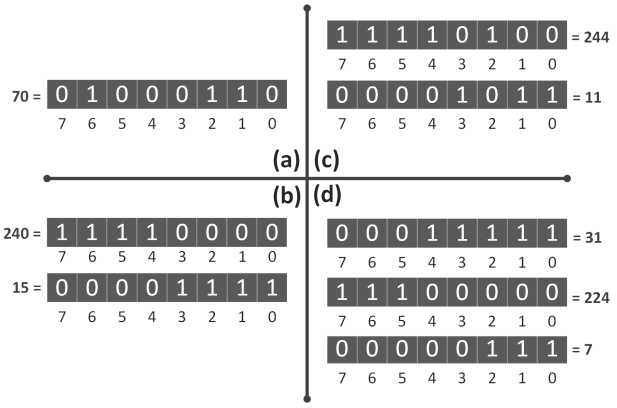


Fig. 4. PDC Encoding: (a) No Reversal or Inversion (b) Reversal (c) Inversion (d) Inversion and Reversal

3.2 Second Step: Encoding

The PDC encoding is similar to that of PIC [10]. Reducing the number of ON bits, within each segment, and mapping these bits to the locations with smallest

Pulses	Inverted	Reversed
1	NO	YES
2	YES	NO
3	YES	YES
0	NO	NO

Fig. 5. PDC Flags

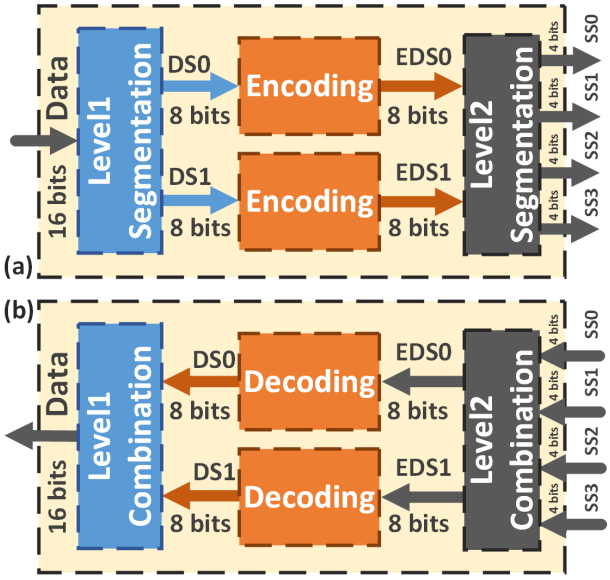


Fig. 6. The Three Steps of PDC (a) Encoding (b) Decoding

possible indices results in a smaller decimal number represented by each segment. This can be achieved using bit-wise inversion and segment-wise reversal. Bit-wise inversion is a conditional one's complement operation on the segment bits, the condition being that more than half of the length of the segment is occupied by ON bits. Reversal is a conditional right-left flipping operation, the condition being that the decimal value of the segment bits after flipping is less than its decimal value before flipping. Fig. 4 illustrates the application of inversion and reversal on a 8-bit data segment. In particular, it shows that inversion and reversal alone cannot minimize the number of pulses. On the other hand, combining inversion and reversal results in the smallest decimal number, namely 7, for that particular example. One's complement and flipping on short data words are not only very easy to implement in hardware but also very energy-efficient. Each of the four types of encoding is represented by a flag number as shown in Fig. 5.

3.3 Third Step: L2 Segmentation

As mentioned earlier, PDC needs to have a second level of segmentation post-encoding to further reduce the number of pulses per segment. This second level of segmentation, shown in Fig. 3, breaks each data segment (DS) into *Sub-segments (SS),* each of which is 4 bits long. The 4-bit length is the optimum segment size for PDC as will be explained in Section 4. The complete three-step process is shown in Fig. 6(a). EDS0 and EDS1 represent the two 8-bit encoded first-level data segments.

3.4 Serialization

The serializer collects the encoded data from the encoder and transmits it serially according to the format shown in Fig. 7. Unlike PIC, each segment is transmitted independently without any header to indicate the encoding flags. The body of the message includes the pulses of the L2 sub-segments (SS) only. The flag information is transmitted along with the start pulse prior to the transmission of SS's. As mentioned earlier, each flag is a 2-bit code representing one of the 4 encoding cases (Fig. 5) for each DS of the L1 segments. The PDC serializer combines all DS flag codes to generate one flag code that is 4 bits long, called the *CFlags*, as shown in Fig. 7. It must be remembered that in the absence of any encoding PIC used the flag code 4 instead of 0 because it needed at least one pulse to represent the absence of data pulses which is already assigned as another flag code. On the other hand, PDC replaces the flag code of 4 with 0 as the "CFlags" pulses are transmitted along with the start pulse, which helps in recognizing 0 without any additional pulse. "SSn" (SS0 to SS3) represents the number of pulses being transmitted for a particular L2 SS. Each of these parts is sent in the form of a pulse stream (one pulse is equal to 1 clock cycle), and each stream is followed by an inter-symbol delay, α, made of 4 clock cycles. Transmission is initiated with a start pulse followed by all the message pulses. It ends with two stop pulses. There is inter-symbol delay, α, of 4 clock cycles after the "Start+CFlags" and "Stop" pulses. The waveforms for PDC transmission are shown in Fig. 8.

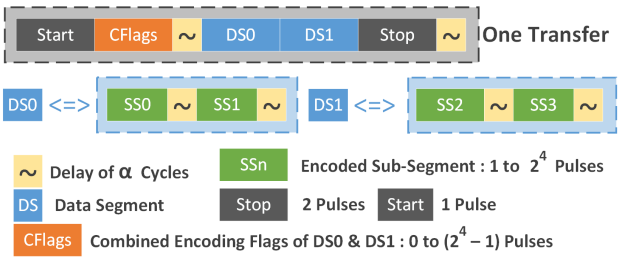


Fig. 7. PDC Transmission Format

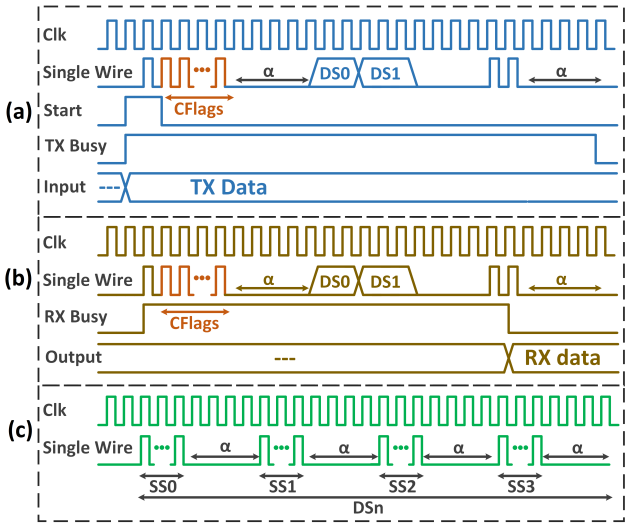


Fig. 8. (a) Transmission (b) Reception (c) L2 Segment Format

3.5 De-Serialization, Decoding and Combination

The receiver counts the pulses of each stream using rising edge detection which eliminates the need for CDR. The count of CFlags pulses is used to infer the segment encoding type which will be used in the segment decoding process. Next, the count of each of the pulse streams is considered as SS. At the reception of last sub-segment SS3, all the sub-segments are combined and decoded to generate data segments, and the decoded segments are combined to assemble the full length of transmitted data. The complete process of three-step decoding is shown in Fig. 6(b), where EDS0 and EDS1 represent the received 8-bit encoded DS. The waveforms for PDC reception are shown in Fig. 8.

3.6 PDC Example

An example of PDC data transmission is shown in Fig. 9. If we transmit the decimal number represented by data without any segmentation and encoding, we need numerous pulses (i.e. 65055). The data rate would be decreased drastically. In the L1 segmentation step, if we break data into two independent segments then we will reduce the decimal number to two smaller numbers, 31 and 254. Though the numbers of pulses are reduced significantly, still these are large enough to deteriorate data rate considerably. Segment 1 has higher number of ON bits as compared to half of the segment size and, therefore, need encoding where all the bits are complemented resulting in fewer number of ON bits. The resulting segment number is increased to 224 instead of decreasing because the ON bits are now located at higher index numbers. This needs the second step of encoding that is to flip the segment bit-wise which results in a smaller number

that is 7. The flag for segment 1 is set to $\{1, 1\} = 3$ to indicate that both steps of encoding are applied. On the other hand, Segment 2 only need inversion of bits as it results in fewer ON bits which are located at lower index numbers. The resulting number in segment 2 is 1. The flag for Segment 2 is set to $\{1, 0\} = 2$ to indicate that bits are inverted only. In L2 segmentation step, if we break data into four independent sub-segments then the number of pulses needed for each sub-segment would reduce further. Both flags are combined to generate one 4-bit CFlags. In serialization or transmission process, we have one start pulse, one CFlags with pulse count of 14, four sub-segments with pulse counts of 8, 1, 2, and 1, and two stop pulses. All of these pulse streams are then transmitted in a packet format as shown in Fig. 7, 8, and 9.

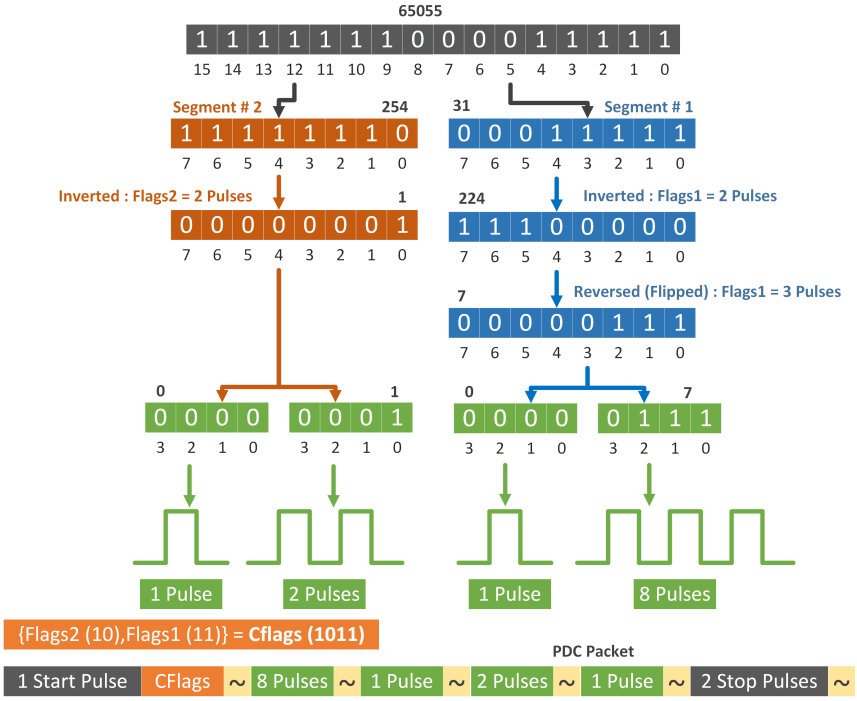


Fig. 9. PDC Example

4 PDC Protocol Analysis and Optimization

4.1 Preliminary Equations

Let b_i^n denote the i-th bit in the n-th sub-segment and f_i^d denote the i-th bit of the flags code for d-th encoded data segment. We assume the number of bits

per segment and the number of sub-segments in the data stream to be both 4. Then the total number of pulses C is given by

$$C = 3 + 6\alpha + \Phi_C + \sum_{n=0}^{3} P_n \tag{1}$$

$$P_n = 1 + \sum_{i=0}^{3} 2^i b_i^n \tag{2}$$

$$\Phi_C = 1 + \sum_{d=0}^{1} 4^d F_d \text{ with } \; F_d = f_0^d + 2f_1^d \tag{3}$$

$$R = \frac{B}{TC} \tag{4}$$

where α is the inter-symbol delay, P_n is the number of pulses required for the n-th sub-segment SS_n, F_d is the flag pulse count for the d-th encoded data segment DS_d, and Φ_C is the combined $CFlags$ flag pulse count. The expression of C in (1) results from a summation over all segments in a data stream of length $B = 16$ bits, which results in the data rate R in (4).

4.2 General Case

A data word length of 16 bits is recommended to guarantee best results and, therefore, the PDC discussion so far and the equations above have considered 16 bits of data to transmit. However, the PDC system can be generalized if data words of more than 16 bits are transmitted. To achieve high data rates, one needs to make sure that the data size is a multiple of 16. If the data words are not multiple of 16, we append 0's at the MSB end of the data. In general, the total number of $L1$ data segments would be $D = B/8$, the total number of $L2$ sub-segments would be $N = D \times 2$, and the total number of CFlags would be $M = D/2$. A data word size greater than 16 bits results in multiple CFlags. One of these CFlags is transmitted along with the start pulse, as mentioned in Section 3, and the remaining CFlags are transmitted as separate pulse streams. Each of these pulse streams is comprised of $CFlags + 1$ pulses followed by an α delay, and is transmitted prior to the transmission of the corresponding data segments. With this notation, we have the following general expression for the total pulse count

$$C = 2 + M + (M + N + 1)\alpha + \sum_{m=0}^{M-1} \Phi_{Cm} + \sum_{n=0}^{N-1} P_n \tag{5}$$

where l is the size of the $L2$ sub-segment, which is 4 in our implementation of PDC. Recall that each of the Φ_{Cm} represents the encoding flags of two adjacent data segments. For example, if $B = 32$ bits, the number of CFlags would be $M = 2$. Φ_{C0} would represent the 16 LSBs (i.e., two data segments, DS0 and DS1). Similarly, Φ_{C1} would represent the 16 MSBs (i.e., the two next data segments, DS2 and DS3). The generalized transmission packet format is shown in Fig. 10.

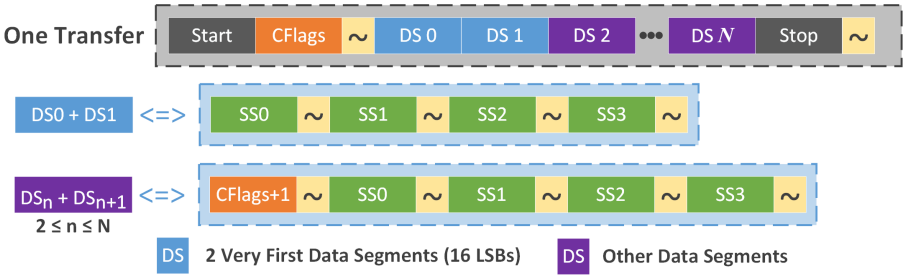


Fig. 10. Generalized PDC Transmission Format

4.3 Optimum Sub-Segment Size

The L2 sub-segment (SS) size is chosen to maximize data rate. For a short SS, α delays inserted between pulse streams to separate symbols increase linearly and reduce the data rate. Similarly, for longer SS, segments have large decimal number representations and, therefore, require a large number of pulses to transmit. The number of pulses increase exponentially with the size of SS, which in turn reduces the data rate rapidly. It is therefore intuitive that there is a segment size for which the data rate is maximum. For PDC, our experimental results have shown that the data rate is maximized when the number of bits per L2 SS is 4.

To find the optimum L2 sub-segment length, we will minimize the number of clock cycles needed to transmit the PDC packet. We know from the previous section that the number of segments is $N = B/l$. Because there is one CFlags for two consecutive segments, the number of CFlags would then be $M = N/2 = B/2l$. For a length l, the number of pulses P_n of (2) becomes

$$P_n = 1 + \sum_{i=0}^{l-1} 2^i b_i^n \tag{6}$$

Assuming that bits 0 and 1 are equally likely, the expected value of P_n is

$$E[P_n] = \frac{2^l + 1}{2} \tag{7}$$

Similarly, using

$$E[F_d] = E[f_0^d] + 2E[f_1^d] = \frac{1}{2} + 2 \times \frac{1}{2} = \frac{3}{2} \tag{8}$$

the expected value of Φ_{Cm} as expressed in (3) is given by

$$E[\Phi_{Cm}] = 1 + E[F_0] + 4E[F_1] = 1 + \frac{3}{2} + 4 \times \frac{3}{2} = \frac{17}{2} \tag{9}$$

Using $N = B/l$ and $M = B/2l$, the expected value of the total number of clock cycles C as given in (5) becomes

$$E[C] = 2 + M + (M + N + 1)\alpha + ME[\Phi_{Cm}] + NE[P_n] \tag{10}$$

$$E[C] = 2 + M(\alpha + 1) + N\alpha + \alpha + \frac{17M}{2} + N\frac{2^l + 1}{2} \tag{11}$$

$$E[C] = 2 + \frac{B}{4l}(\alpha + 1) + \frac{B}{l}\alpha + \alpha + \frac{17B}{8l} + \frac{B(2^l + 1)}{2l} \tag{12}$$

Taking the derivative with respect to l and equating it with zero, we get

$$\frac{\partial E[C]}{\partial l} = -\frac{B}{4l^2}(\alpha + 1) - \frac{B}{l^2}\alpha - \frac{17B}{8l^2} + \frac{B\ 2^l \ln(2)}{2l} - \frac{B\ 2^l}{2\ l^2} - \frac{B}{2l^2} = 0 \tag{13}$$

$$-2(\alpha + 1) - 8\alpha - 17 + 4l\ 2^l\ \ln(2) - 4\ 2^l - 4 = 0 \tag{14}$$

$$2^l(l\ \ln(2) - 1) = \frac{1}{4}(23 + 10\alpha) \tag{15}$$

which gives the following expression of α as function of l

$$\alpha = \frac{2}{5}\left[2^l(l\ \ln(2) - 1) - \frac{23}{4}\right] \tag{16}$$

This function is plotted in Fig. 11 and can be used to obtain the optimum sub-

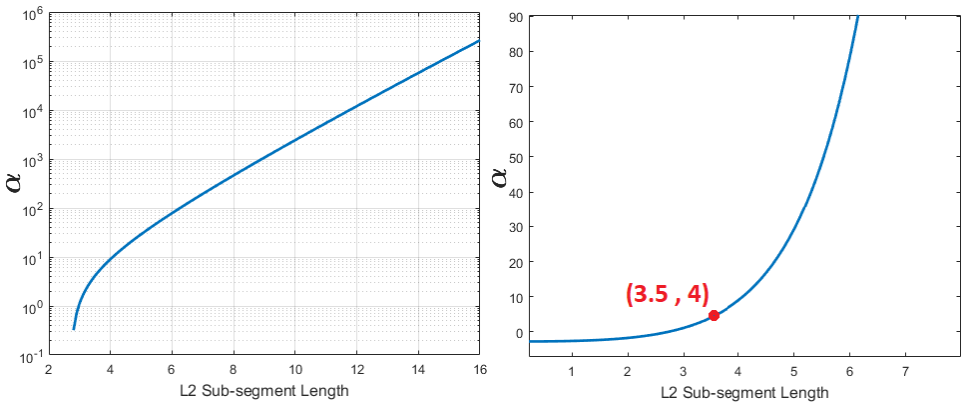


Fig. 11. α vs. L2 Sub-segment Length

segment size for any value of α. For instance, when $\alpha = 4$, we get $l_{opt} = 3.5 \approx 4$. An alternative graphical method to find l_{opt} for a given α is to plot the left- and right-hand sides of

$$l\ \ln(2) - 1 = \frac{1}{4}(23 + 10\alpha)2^{-l} \tag{17}$$

as function of l. The optimal segment size l is at the intersection of these two curves. Such plot is shown in Fig. 12 for $\alpha = 4$, which corresponds to

$$l\ \ln(2) - 1 = \frac{63}{4}2^{-l} \tag{18}$$

Again, the optimum value of the L2 sub-segment length is found to be

$$l_{opt} = 3.5 \approx 4 \tag{19}$$

If the sub-segment length is increased or decreased from this optimum value, the data rate declines rapidly. To achieve maximum data rate at a given cock frequency, one must therefore operate the PDC signaling with an L2 sub-segment length of 4 bits.

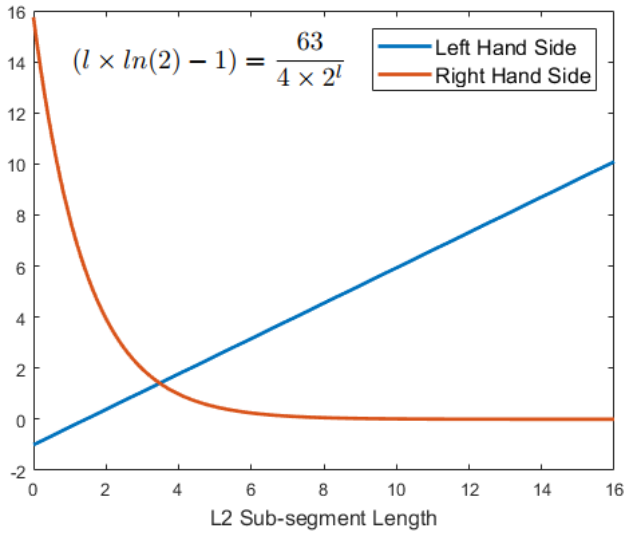


Fig. 12. Optimal Segment Length for $\alpha = 4$

4.4 Data Rate

PDC is dynamic in that the actual data rate of the protocol is dictated by the pulse C count which is very much data dependent as is clear from (1) and (5). We have analyzed the statistical distribution of PDC data rates using exhaustive sampling of 16-bit data words ($2^{16} - 1$ PRBS), each segmented into four 4-bit sub-segments. For data rate calculations, we use a 25MHz clock. The analysis has shown that the three-step encoding process (L1 Segmentation + Encoding + L2 Segmentation) is generally needed not only to maximize the (average) data rate but also to tighten the distribution of dynamic rates around the average data rate. The effect is shown in Table 1. Note that after full three-step encoding, there is a significant improvement in the average data rate and a tightening of the data rate distributions around the average. The histogram of data rate is shown in Fig. 13.

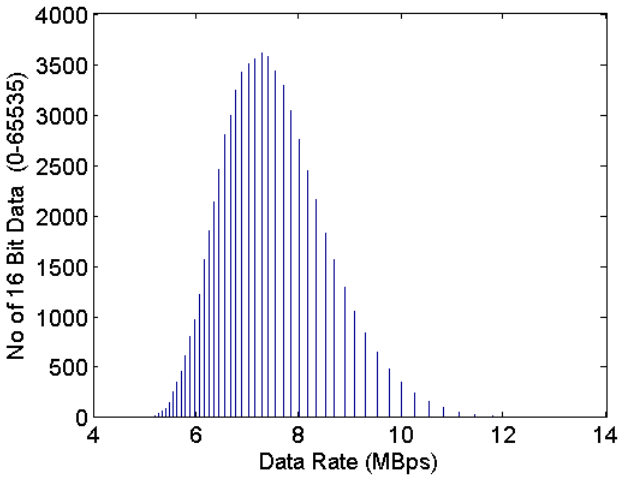


Fig. 13. Data Rate Analysis

Table 1. PDC Data Rate Comparisons

Encoding Process Steps	*Max.	*Avg.	*Min.
L1 Segmentation Only	19	1.81	0.75
L1 Segmentation + Encoding	19	3.92	0.97
L1 Seg. + Encoding + L2 Seg.	12.9	7.33	4.87

*All data rate entries are in Mb/s.

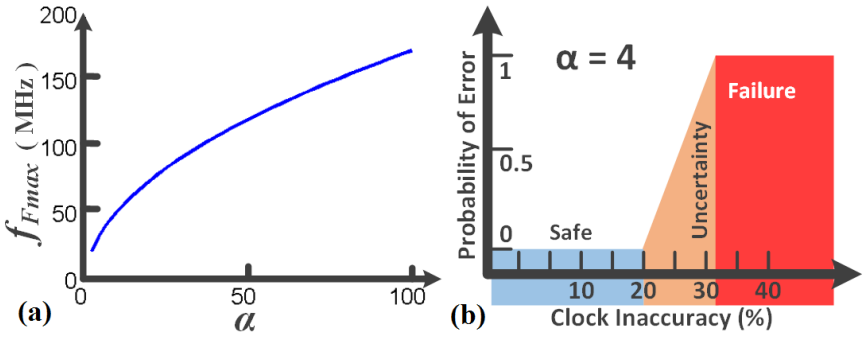


Fig. 14. (a) α vs Inaccuracy Limit (b) Regions of Operation ($f_S = 25MHz$) [7]

4.5 Tolerance with Respect to Clock Variations

Even if similar clock generators are used at the transmitter and receiver end, there would be a slight difference between their clock frequencies due to the various sources of variability. Considering the slow speed end as a reference and keeping the inter-symbol delay α the same for both ends, the rate of the fast clock should not exceed a limit above which decoding errors start to occur [7]. In Fig. 14(a), we plot the maximum limit on the clock speed of the fast clock, f_{Fmax}, for a range of α while the slow clock, f_S, is at 25MHz. Fig. 6(b) identifies different regions of operation. Beyond the safe region of operation, there is a region of uncertainty in which errors start occurring randomly. At a certain level of clock discrepancy, total failure occurs due to the failure in detecting even a single inter-symbol delay. The recommended region of operation is of course the one delimited by f_{Fmax}.

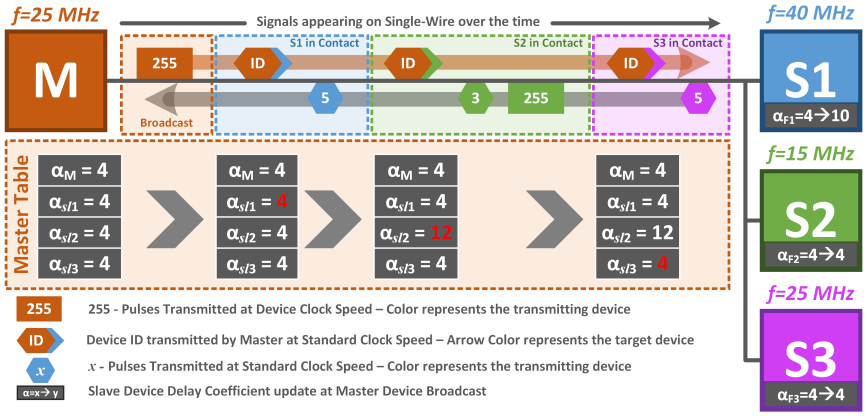


Fig. 15. An Example of Automatic Parameter Detection [9]

4.6 Automatic Protocol Configuration

While PDC is tolerant towards clock discrepancies, its per-device parameters have to be adapted in order to support high level of clocking differences. The device parameter setting may be done manually or at the factory in a way similar to existing low-power, single wire solutions. Not only does manual or factory setting method limits the scalability of IoT network to an arbitrary number of devices, but also it prevent the devices from communicating at the highest data rate allowed by the protocol. To overcome this limitation an algorithm for automatically detecting and setting the PDC protocol parameters at the power-on phase can be used that to help remove the restriction on the IoT devices in the PIC network to communicate at a pre-specified baud rate and allows the devices

with different capabilities to communicate reliably. At power-on, based on clock-rate differences the master device configures all the slave devices connected to a single-channel network prior to the start of any device-to-device communication. Pulse count differences between the pulse trains at the transmitter and receiver are used in closed-form formulas to find a suitable inter-symbol delay coefficient so as to eliminate the need to know the exact clock rates at both ends of the link. An example flow of the configuration process is shown in Fig. 15 where the master device broadcasts a clock pulse stream and then contacts the slave devices one-by-one to receive their responses. Each of the slave devices receives the incoming clock stream and compares with its local clock speed. At master device request, if the device is at higher clock speed as compared to master, five pulses are transmitted back and the slave device updates it's α accordingly. If the device is at lower clock speed as compared to master, three pulses followed by 255 pulses are transmitted back at local clock rate. Depending on the response format, the master updates a table of α entries for the slave devices. Later, the master device picks up the corresponding α from the look-up table and starts communicating with a slave device.

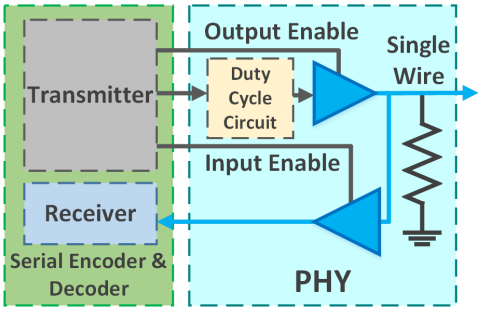


Fig. 16. Proposed PIC PHY Layer [6]

4.7 Power Management

Although PDC [10] itself is power-efficient, the PDC physical layer can be the source of significant power consumption. A PDC power management policy can be used to improve further its ultra-low power characteristics without impacting data rates. This policy shows that an additional 22% power saving at the PIC PHY layer can be achieved using the duty cycle of the pulse as a power control parameter, as shown in Fig. 16. A mathematical model helps in selecting the physical design parameters related to PDC PHY power management. For an input duty cycle of 20ns, the implementation of this policy in 45nm CMOS technology shows that the smallest duty cycle of 0.39% can be generated to achieve 22% of extra power saving.

4.8 Reliability

In this subsection, we analyze PDC in terms of its robustness and reliability in comparison with PIC. We use packet failure rates as a comparison metric. Considering the PIC transmission format for 16 bits of data, there exist several locations in the packet which, if corrupted, may be the source of complete packet failure. For instance, there are 8 to 16 locations of α delays, and each can be a source of packet failure if it gets corrupted. This corruption makes it difficult for the receiver to detect α and sample the subsequent pulse streams at the expected instants of transmission. Similarly, if the information on the number of indices (NOI, one for each packet) is not received successfully, there is no way for the receiver to know about the number of incoming index pulse streams. Additionally, the corruption of two sync pulses can also misalign the pulse streams during PIC decoding. The number of locations, PE_{PIC}, whose corruption in a PIC stream of 16 bits results in a complete packet failure during transmission has been found in the range

$$8 + 2NOI_{min} + 2 \leq PE_{PIC} \leq 8 + 2NOI_{max} + 2 \tag{20}$$

$$10 \leq PE_{PIC} \leq 18 \tag{21}$$

when $0 \leq NOI \leq 4$. Analyzing the PDC transmission stream, we know there would always be a fixed number of pulse streams that directly eliminates the threat of packet failure due to the NOI and sync pulses, as these are not included in the packet at all. In terms of α delays, there are only 6 locations ($PE_{PDC} = 6$) for all the possible 16 bit data words. This makes PDC simpler and more reliable even when compared with the best PIC case, $PE_{PIC} = 10$ in (21).

5 Experimental Results

A full experimental setup, similar to that of PIC [10], is implemented in Verilog on the Xilinx Virtex-7 FPGA platform. The prototype platform is used to verify PDC functionality and performance. Extensive simulations and real time hardware verification are performed in order to verify the results. To make a fair comparison with PIC, we have used 16-bit data words and a clock rate of 25 MHz as in the PIC testing system [10]. The same experimental flow is adopted in which the PDC transmitter sends the 16-bit data starting at 0 with an increment of 1 at each transmission. The receiver resends the same data back. The returned and original data words are compared to verify the complete round-trip chain.

To get the most realistic comparison with PIC in terms of power and area, we have synthesized the PDC system using GLOBALFOUNDRIES $65nm$ technology and found that PDC consumes around $25\mu W$ with a gate count of around 2150, while offering dynamic data rates in the range of 4.87 - 12.9 Mb/s (7.33 Mb/s average) with a 25-MHz clock. As compared to PIC, the PDC increases average data rate by 78%, makes the transmission more reliable with respect to

packet failure rates, and still remains well within the power and area budget of PIC. Table 2 shows a comparison between PIC and PDC. In Fig. 17, an overall comparison of PIC and PDC is presented. Along with the increased data rate, the gate count, power consumption and energy per bit in PDC are also reduced as compared to PIC. The reliability in terms of vulnerable locations is also reduced by 40% as compared to the best reliability case in PIC transmission.

Table 3 presents a comparison of PDC with CDR based simple serial transfer technique. The ASIC synthesis results show that PDC can reduce power consumption by more than 72% and area by more than 89% as compared to the best cases of CDR based serial transfer techniques available to-date.

Table 2. Synthesis Results

	Data Rate (Mb/s)	**Power** (μW)	**Avg. E$_b$** (pJ/bit)	**Area** ($gate\ count$)
PDC	4.87-12.9 (7.33 Avg.)	≈25	3.41	≈2150
PIC	3.1-8.5 (4.1 Avg.)	≈ 26.6	6.49	≈2356

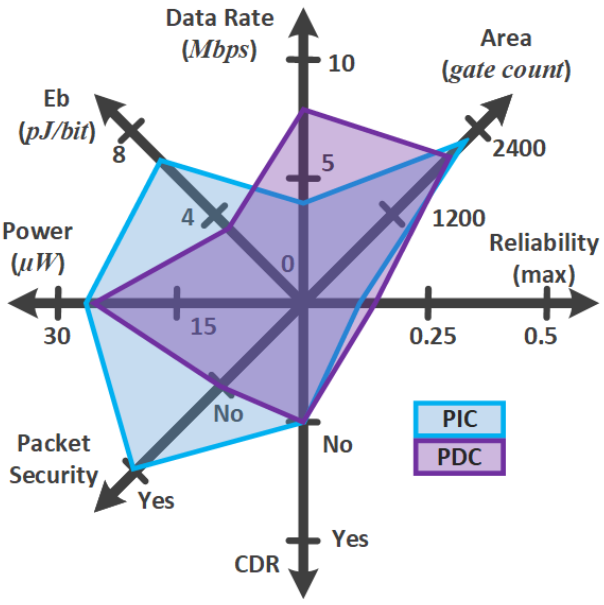


Fig. 17. PDC vs. PIC

Table 3. PDC Comparison With Simple-Serial

	Power (μW)			**Area** (*gate count*)			
	SRL[a]	**CDR**	**Total**[d] (PI)[e]	**SRL**	**CDR**[c]	**Total**[d] (PI)[e]	
PDC	25	N/A	25	2150	N/A	2150	65nm
NST[b]	32.1	70	102.1 (75.5%)	1327	15600	16927 (87.3%)	90nm [12]
		62.5	94.6 (73.6%)		60000	61327 (96.5%)	90nm [14]
		90	122.1 (79.5%)		N/A	N/A	90nm [15]
		57.5	89.6 (72.1%)		19800	21127 (89.8%)	65nm [16]
		60.6	92.7 (73%)		N/A	N/A	28nm [13]

[a.]Serializer [b.]Normal Serial Transfer [c.]Estimated calculation [d.]SRL+CDR [e.]%Increase as compared to PDC

6 Conclusions

Pulsed Decimal Communication (PDC) is an improved version of Pulsed Index Communication (PIC). It is a novel, simple yet robust method of signaling over single-channels that fulfils the requirements for high-data rate, ultra-low power protocols for IoT sensors. The concept of transmitting the indices of ON bits is replaced with the transmission of reduced decimal numbers. The PDC segmentation and encoding process reduces the overall number of transmission pulses by reducing decimal numbers, hence improving the data rate by 78% with respect to PIC. The average data rate is 7.33 Mb/s with a 25-MHz clock consuming a power of $25uW$ with a gate count of 2150 in $65nm$ technology. Unlike PIC, there are fewer inter-symbol delays α in the PDC packet. Moreover, a fixed number of pulse streams is used for transmission, thus providing a simpler and more reliable communication technique that results in a significant decrease in packet failures, especially for low-end devices and sensors. The elimination of variations in the number of pulse streams makes the use of error detection and correction schemes easier. Like PIC, the PDC reduces silicon area and power consumption significantly by eliminating the need for a CDR. It is also robust with respect to skews, jitters, and clock variations as its decoding is based on counting the rising edges of the transmitted pulses. Additionally, all the presented PIC-related work for robustness analysis [7], auto detection of communication parameters [9], and power management [6] are valid for PDC without any change.

Acknowledgments

This work has been supported by the Semiconductor Research Corporation (SRC) under the Abu Dhabi-SRC Center of Excellence on Energy-Efficient Electronic Systems (ACE4S), Contract 2013 HJ2440, with customized funding from the Mubadala Investment Company, Abu Dhabi, UAE.

References

1. Jaewook Byun, Seong Hoon Kim, and Daeyoung Kim. Lilliput: Ontology-Based Platform for IoT Social Networks. *IEEE International Conference on Services Computing*, pages 139–146, Anchorage, AK, USA, June-July 2014.
2. Shi Dayu, Xu Huaiyu, Su Ruidan, and You Zhiqiang. A GEO-Related IoT Applications Platform Based on Google Map. *7th IEEE International Conference on e-Business Engineering (ICEBE)*, pages 380–384, Shanghai, China, November 2010.
3. C.A. dos Reis Filho, E.P. da Silva, E. de L. Azevedo, J.A.p. Seminario, and L. Dibb. Monolithic data circuit-terminating unit (DCU) for a one-wire vehicle network. *Proceedings of the 24th European Solid-State Circuits Conference (ESSCIRC '98)*, pages 228–231, Hague, Netherlands, September 1998.
4. Jenq Muh Hsu and Chin Yo Chen. A Sensor Information Gateway Based on Thing Interaction in IoT-IMS Communication Platform. *Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 835–838, Kitakyushu, Japan, August 2014.
5. MAXIM. *OneWireViewer User's Guide, Version 1.4*. AN3358, 2009.
6. Shahzad Muzaffar and Ibrahim (Abe) M. Elfadel. Power management of pulsed-index communication protocols. *33rd IEEE International Conference on Computer Design (ICCD)*, pages 375–378, New York, NY, USA, October 2015.
7. Shahzad Muzaffar and Ibrahim (Abe) M. Elfadel. Timing and robustness analysis of Pulsed-Index protocols for single-channel IoT communications. *23rd IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC 2015)*, pages 225–230, Daejeon, South Korea, October 2015.
8. Shahzad Muzaffar and Ibrahim (Abe) M. Elfadel. A versatile hardware platform for the development and characterization of IoT sensor networks. *59th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'16)*, pages 1–4, Abu Dhabi, UAE, October 2016.
9. Shahzad Muzaffar, Numan Saeed, and Ibrahim (Abe) M. Elfadel. Automatic protocol configuration in single-channel low-power dynamic signaling for IoT devices. *24th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC 2016)*, pages 1–6, Tallinn, Estonia, September 2016.
10. Shahzad Muzaffar, Ayman Shabra, Jerald Yoo, and Ibrahim (Abe) M. Elfadel. A pulsed-index technique for single-channel, low-power, dynamic signaling. *Design, Automation and Test In Europe (DATE'15)*, pages 1485–1490, Grenoble, France, March 2015.
11. Shahzad Muzaffar and Ibrahim (Abe) M. Elfadel. A pulsed decimal technique for single-channel, dynamic signaling for IoT applications. *25th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, Abu Dhabi, UAE, October 2017.
12. M. Loh and A. Emami-Neyestanak. All-digital CDR for high-density, high-speed I/O. *12th IEEE Symposium on VLSI Circuits (VLSIC'10)*, pp. 147–148, Honolulu, HI, USA, June 2010.
13. L.-K. Soh and W.-T. Wong. A 2.512.5Gbps interpolator-based clock and data recovery circuit for FPGA. *4th Asia Symposium on Quality Electronic Design (ASQED)*, pp. 373–379, Penang, Malaysia, July 2012.
14. M. Loh and A. Emami-Neyestanak. A 3x9 Gb/s Shared, All-Digital CDR for High-Speed, High-Density I/O. *IEEE Journal of Solid-State Circuits(JSSC)*, vol. 47, no. 3, pp. 641–651, March 2012.

15. Q. Du, J. Zhuang, and T. Kwasniewski. A 2.5 Gb/s, Low Power Clock and Data Recovery Circuit. *20th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 526–529, Vancouver, BC, Canada, April 2007.
16. Y. Urano, W.-J. Yun, T. Kuroda, and H. Ishikuro. A 1.26mW/Gbps 8 locking cycles versatile all-digital CDR with TDC combined DLL. *45th IEEE International Symposium on Circuits and Systems (ISCAS'13)*, pp. 1576–1579, Beijing, China, May 2013.