sciendo INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS

Research Article | DOI: 10.2478/ijssis-2024-0029 Issue 1 | Vol. 17 (2024)

Design and Implementation of Low Power and High Data Rate Edge Coded Signaling Architecture for IoT Devices

Jamuna M[1,*] and
VijayaPrakash A. M[2]

[1]Research Scholar, Department of ECE, Bangalore Institute of Technology, Karnataka, India

[2]Department of ECE, Bangalore Institute of Technology, Karnataka, India

*E-mail: jamunam@bit-bangalore.edu.in

Received for publication
May 19, 2024.

Abstract

Edge coded signaling (ECS) is a recently introduced communication protocol designed for single-channel signaling among constrained internet of things (IoT) nodes. This study aims to enhance the data rate of ECS, without increasing the clock frequency or power consumption. The goal is to develop an improved protocol that maintains the inherent advantages of ECS while achieving a higher data rate. The proposed solution, double data rate ECS (DDR-ECS), leverages both pulse edges of the ECS pulse stream for information encoding, drawing an analogy to DDR memory systems. The study presents a detailed design of low-power architecture for the DDR-ECS transceiver. This design was synthesized using a 65 nm ASIC process to evaluate its performance in terms of power consumption, gate count, and data rate. The synthesis results demonstrate that the DDR-ECS transceiver preserves the key features of ECS, including tolerance to clock frequency variations and compatibility with lightweight cryptographic algorithms. It effectively doubles the dynamic data rate to the range of 12–73.5 Mb/s while consuming an equivalent power of 13 µW and occupying a compact form factor of only 1,755 gates. The introduction of DDR-ECS represents a significant advancement in the ECS protocol, offering a novel approach to doubling the data rate without increasing the clock frequency or power envelope. This innovation retains the simplicity and efficiency of ECS, providing an enhanced communication solution for constrained IoT nodes.

Keywords

Edge coded signaling (ECS), Double data rate ECS, IoT, Low power

I. Introduction

The internet of things (IoT) extends the reach of the internet of people (IoP) to include inanimate objects, offering a valuable chance for innovators, engineers, and technologists to reevaluate basic assumptions about the physical infrastructure supporting IoP. IoT is undoubtedly more than just a concept these days, especially given how commonplace it is in daily life. In everyday life, the most well-known IoT framework is the “cell phone.” IoT applications are not limited to smart homes. They encompass all facets of various areas, including the business implications of public safety, healthcare, and agriculture. An alternative name for IoT is the Internet of Everything (IoE). In terms of the many iterations of IoT real-global software, IoT stands for IoT, which is a network of multiple devices that may interact with each other without

the need for direct human interaction. It also makes data transfer speedy and environmentally friendly. IoT-enabled devices enhance performance overall, safety, and operational efficiency. The IoT plays an increasingly important role in connecting the things within our environment to the Internet and makes our lives easier due to the ability to access these non-internet items from any location (outside our local network), regardless of their location. Keeping up with the growing technology is challenging as it grows more rapidly than expected [1].

A key assumption is the necessity of a synchronization circuit in the receiver of a single-channel, serial communication interface to extract data bits from the incoming bit stream. Interfaces such as USB, Ethernet, and fiber optics networks, which have significantly contributed to IoP's rapid expansion, exemplify this need. The synchronization circuit, known as clock and data recovery (CDR), is essential. It extracts the clock signal from incoming data bits and uses this inferred signal to re-time or re-sample the data bits at the optimal moment within the bit period. This process depends on transitions observed in the incoming data bits, and the selection of sampling times is managed by feedback circuit architectures, with the phase-locked loop (PLL) being the most common. Figure 1 shows a typical CDR circuit architecture using PLL.

The encoding of incoming bits is a critical aspect that affects CDR performance. A popular format called non-return-to-zero (NRZ) encoding designates HIGH for bit 1 and LOW for bit 0 over the bit duration. The CDR uses transition edges between HIGH and LOW bits to detect these edges in order to generate a clock signal that matches the bit rate. However, transition information is missing when the incoming data consists of prolonged sequences of HIGH or LOW bits, which makes it difficult for the PLL to lock onto edges. Using encodings where the 1 bit returns-to-zero (RZ) in the last 50% of the bit period is one way to counteract this information loss. This causes a transition to occur each time a string of 1 bits is received. Manchester encoding fixes the information loss problem in the CDR phase detector by guaranteeing transitions for both 1 and 0 bits over the bit period. The disadvantage of RZ encodings, however, is that they need a higher channel capacity than in the NRZ scenario, or they require data bit transfer at lower rates in order to meet the channel bandwidth requirements.

Using block encoding in place of bit encoding is an alternate strategy for balancing bit transitions in the data stream. The byte-oriented 8b/10b encoding [2] is one commonly used block approach that is used in many different communication technologies, including Gigabit Ethernet, USB 3.0, SATA, and Infiniband, among others. The guarantee that there can be no more than two bits of difference between 1 and 0 in a 10-bit code, and that no more than five bits can have the same value in a row, is the main reason why 8b/10b is important for CDR. Considering the proven relationship between CDR performance and data encoding, a basic but ambitious question is raised: Are there data encodings that could completely replace the need for CDR while preserving reliable and appropriate data rates? The possible reduction of receiver design in sensor or actuator devices makes this investigation relevant to the infrastructure of the IoT. This simplified construction would use less energy and take up less room. However, it is crucial that data security and authentication—two further essential requirements for restricted IoT nodes—be maintained in spite of such a straightforward, low-power, and compact architecture.

The use of lightweight cryptography is one potential solution to this problem. Lightweight cryptography still requires a large overhead in both space and power, even with recent advances. Moreover, its lightweight design leaves it open to relatively simple harmful assaults. Therefore, while pursuing

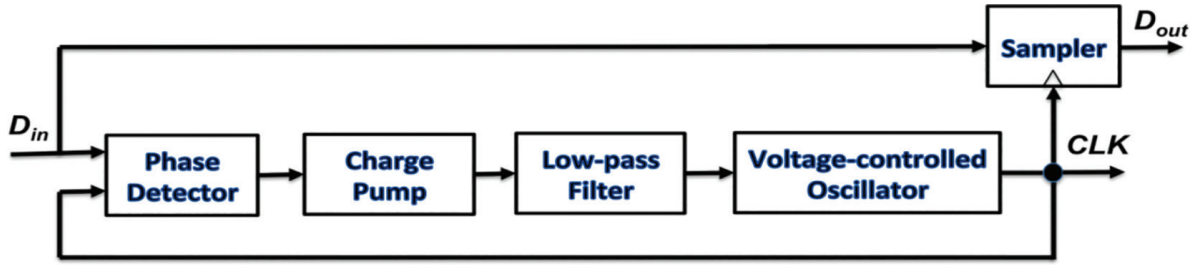

Figure 1: Architecture of a CDR circuit using PLL. CDR, clock and data recovery; PLL, phase-locked loop.

simplified architectures for IoT devices, maintaining robust data authentication and security remains a crucial consideration. The answer to this existential challenge has emerged recently in the form of a novel class of block-oriented signaling systems intended for serial, single-channel interactions. These techniques are built on the fundamental principle of encoding data bits as pulse trains, with the receiver decoding the count of pulses that are also conveyed. As the title of this article suggests, this protocol family is appropriately titled edge coded signaling (ECS) since the receiver uses the rising edges of these pulses to decode transmitted bits.

Two noteworthy members of this emerging family are pulsed index communication (PIC) [3] and pulsed decimal communication (PDC) [4]. Notably, both PIC and PDC eliminate the need for CDR, differing in fundamental aspects related to transmission security and packet reliability. PIC has demonstrated compatibility with symmetric stream ciphers, offering multilayered protection for data in transmitted packets [5]. On the contrary, PDC lacks such compatibility and is not conducive to multilayered encryption for secure data transmission. However, PDC achieves superior data rates compared with PIC and has been experimentally validated to experience fewer packet transmission failures [4]. The data protection strategy facilitated by PIC involves a close collaboration between the communication protocol and the lightweight encryption algorithm. The communication protocol supplies additional protocol parameters to the encryption algorithm, enhancing the protection of the data. In contrast to current single-wire, CDR-less protocols like 1-wire [6–8], PIC offers a dual advantage of higher data rates (in Mbps as opposed to Kbps) and enhanced security (with an increase in attack complexity by up to a factor of 220). Importantly, these advantages are achieved while staying within the power and footprint constraints of IoT devices.

This article aims to present a comprehensive overview of the current state-of-the-art in edge-coded, CDR-less protocols. This is achieved by developing, implementing, and testing a novel addition to the edge-coded family of block data encodings. This new member, referred to as ECS, seeks to amalgamate and enhance the key advantages of PIC and PDC while mitigating their respective pitfalls and limitations. Like its predecessors [3] and PDC [4], ECS adopts the fundamental concept of encoding bits as pulse streams, utilizing inter-symbol spacings to separate data words.

However, ECS adds a number of characteristics that go beyond PIC and PDC, such as the following:

(i) To reduce the amount of ON bits and their index numbers, ECS uses an efficient segmentation method and a streamlined encoding scheme. This reduces the total number of packet pulses required to transmit a data word.
(ii) ECS creates the most compact packet for a given data word length, yielding the highest possible data rate. A highly compressed header outlining the encoding processes performed on the data bits is the main method used to accomplish this.
(iii) ECS offers the flexibility to design transceivers for the transmission of multiple data words or the implementation of data transmission pipelining. It also makes use of edge detection in received pulses to achieve exceptional robustness against jitters, skews, and clock inaccuracies between the transmitter and the receiver.
(iv) To improve communication security, the ECS packet has a layered architecture and dynamic components that work well along with cryptographic techniques.
(v) The architecture of ECS is flexible, enabling configuration for any topology of signaling, such as Master-Slave, Ring, Star, or Tree.

II. Literature Survey

In recent years, the IoT has become an integral part of various aspects of our lives, including mobile devices, healthcare, education, and environmental monitoring. The IoT consists of interconnected devices equipped with sensors, networks, software, and hardware, all working together to collect, analyze, and transmit data. A crucial role of IoT protocols is to facilitate communication between devices while ensuring secure communication channels. This paper [16] offers a detailed examination of the IoT architecture and reviews different IoT protocols and their respective architectures. This comparative study assists researchers in choosing protocols that meet their specific needs and goals. The rise of IoT also presents new challenges, particularly regarding spectrum constraints. The 5th Generation network, known for its high capacity and fast data transmission, provides a promising solution. This paper [17] presents a comparative analysis of various technologies based on data rates. Additionally, it provides an extensive overview and discussion of power consumption techniques for FPGA-based designs [18]. By offering critical insights into low-power methodologies, the paper aims to introduce new approaches to design optimizations, serving as a valuable resource for researchers in this field.

Table 1: Comparison of IoT communication protocols

Wired protocols					Wireless protocol				
Characteristics	1-wire protocol	PIC protocol	PDC protocol	Dynamic ECS protocol	Bluetooth	Zigbee	Z-wave	6 LoWPAN	SigFox
Standard	NA	NA	NA	NA	IEEE 802.15.1 [20]	IEEE 802.15.4 [20]	Z-Wave [20]	IEEE 802.15.4 [20]	Sigfox [21]
Frequency bands	NA	24 MHz [16]	25 MHz [17]	25 MHz [18]	2.4 GHz [22]	2.4 GHz [23]	868–908 MHz [24]	868 MHz (EU) 915 MHz (USA) 2.4 GHz (Global) [24]	868 MHz (EU) 902 MHz (USA)
Network	1-wire network [8]	Ultra-low power network [16]	Ultra-low power network [17]	Ultra-low power network [18]	WPAN [25]	WPAN [25]	WPAN [25]	WPAN [25]	LPWAN [26]
Topology	Master and Slave [8]	Master and Slave [16]	Master and Slave [17]	Master and Slave [18]	Star–Bus [27]	Star, Mesh cluster	Mesh	Star-Mesh [27]	Star
Power	Low power protocol	26.6 μW [16]	25 μW [17]	19 μW [18]	30 mA Low power [28]	30 mA Low power [28]	2.5 mA Low power [29]	(1–2 years lifetime on batteries) [29]	10–100 mW
Data rate	16 Kbps [16]	4.1 Mbps [16]	7.33 Mbps [17]	4.2–26.7 (6.4 Avg.) [18]	1 Mbps	250 Kbps	40 Kbps [30]	250 Kbps	100–600 bps
Common applications	IoT sensor applications [16]	IoT sensor applications [16]	IoT sensor applications [17]	IoT sensor applications [18]	Wireless headsets, audio applications [31]	Controlling and monitoring home industry [31]	Home monitoring and controlling [31]	Monitor and control through the internet [31]	Energy meters & street lighting

ECS, edge coded signaling; IoT, internet of things; PDC, pulsed decimal communication; PIC, pulsed index communication.

The central objective of IoT is to achieve seamless interconnectivity, envisioning a world where any entity can connect to anything, anywhere, using any available network [19]. Protocols are crucial in facilitating this interconnectivity within IoT ecosystems. They manage heterogeneity, adapt to dynamic changes, ensure security, and handle device management, guaranteeing smooth operation among numerous devices. This study explains the architecture of IoT systems and outlines the services offered by different protocols, each designed to meet specific communication needs and challenges.

This section provides guidelines for researchers to select an appropriate communication protocol by presenting comparative analyses among various options. The comparison takes into account criteria such as network type, standard, topology, power consumption, and data rate. For devices with limited battery power, wireless protocols like 6LoWPAN, ZigBee, Bluetooth, and Z-Wave are recommended due to their low power consumption. Among wired protocols, the Dynamic ECS protocol is notable for its low power usage.

Regarding data rate, wireless protocols such as Bluetooth, 6LoWPAN, ZigBee, Sigfox, and Z-Wave offer data rates of up to 1 Mbps. By contrast, the ECS protocol delivers an average data rate of 12 Mbps, making it a highly efficient option for signaling between constrained IoT nodes. Table 1 provides a detailed comparison of both wired and wireless communication protocols based on factors such as power consumption, data rate, frequency bands, topology, and applications. To address the challenge of improving ECS data rates, a novel protocol called double data rate ECS (DDR-ECS) has been introduced.

III. Summary of the Literature Survey

Protocols like WiFi, WLAN, TCP/IP, and USB [11–13] offer high data rates but are energy-intensive and require advanced controllers for bidirectional communication. Conversely, low-power protocols like 1-Wire [14] and UART [15] have limited data rates, making them less suitable for highly constrained IoT edge devices. Despite extensive research proposing numerous solutions to enhance data rates and minimize power consumption in IoT devices, there remains a critical need for further exploration into IoT-specific transmission and reception schemes that achieve low power consumption while maintaining high data rates. The existing literature has laid a strong foundation, highlighting various methodologies and technologies aimed at addressing these challenges. However, the unique constraints and requirements of IoT devices necessitate tailored approaches that are not only efficient but also adaptable to diverse IoT applications. In this study, we aim to bridge this gap by introducing an innovative low-power architecture specifically designed to optimize data rates for IoT devices.

IV. Problem Statement

In traditional single-channel communication systems, synchronization between the incoming bit-stream and the receiver's local clock is achieved through CDR circuitry. This CDR functionality significantly increases power consumption, adds complexity to hardware design, and enlarges the physical size of transceivers. For highly constrained IoT devices, which require minimal power consumption and compact form factors while ensuring secure communication at relatively high speeds (tens to hundreds of megabits per second), conventional methods fall short. This is due to the trade-off between power consumption and data rate. High data rate protocols like WiFi, WLAN, TCP/IP, and USB demand high power and complex controllers for bidirectional communication. By contrast, low-power protocols like 1-Wire and UART offer reduced data rates, making them unsuitable for constrained IoT edge devices.

To address the challenge of enhancing the data rate of ECS within a specified power envelope and clock frequency, this paper focuses on the "Design and Implementation of Low Power and High Data rate ECS Architecture for IoT Devices." The proposed solution introduces a novel protocol called DDR-ECS, analogous to DDR memory systems. While the concept is appealing, its hardware implementation poses significant challenges. This research aims to develop an efficient design for the DDR-ECS transceiver, maintaining ECS's inherent features while doubling the data rate at the same clock frequency and within the same power constraints. ASIC synthesis of the transceiver using a 65 nm process node shows that DDR-ECS achieves comparable power and area optimization as ECS, with only a minimal increase in gate count. Additionally, DDR-ECS effectively doubles the dynamic data rate, reaching an average minimum Mbps at a moderate MHz clock rate.

V. Existing ECS Transmission

The basic concept behind ECS is that instead of transmitting the actual data bits, the active bits in a data word are identified, and their index numbers are sent as pulse streams. This process is illustrated in

Figure 2(a) and Figure 2(b), where the bit sequence "0101" is converted into a sequence of pulses. Each pulse in the series corresponds to the ordinal position of the active bits in the binary sequence, represented as n + 1. One pulse corresponds to the initial active bit at position 0 in the first series, and three pulses correspond to the active bit at position 2 in the second series. Pulse sequences are separated from one another using an inter-symbol separator (α). It is vital to understand that α is a spacing or separation symbol measured in clock cycles rather than a time delay. There is no need for both ends to synchronize; the local transmitter clock determines the clock cycle count during transmission and the local receiver clock during reception.

It is also important to emphasize that the pulse count that corresponds to the index number is always increased by 1. Taking this action is essential to controlling index 0 transmission. If the bit at index 0 is active, no pulse would be conveyed without this modification. The ECS receiver counts the incoming rising edges in each input pulse series, subtracts one to get the index number (n = PulseCount – 1), and places a data-bit at the determined index number. It may seem at first that transmitting pulse series requires more work than transmitting raw bits, as shown in Figure 2C. This apparent disadvantage is lessened, though, by the possibility of achieving high data rates by using an encoding process that reduces the index numbers. To do this, the bit stream is divided into smaller segments, the number of active bits in each segment is reduced, and the active bits are moved to lower index locations [9]. The index numbers are sent as part of a packet header together with the encoding information and the number of active bits in the encoded payload. Similar to how the index numbers are communicated, all of the information contained in the packet header is sent as pulse streams. Essentially, ECS encodes bits as edge counts instead of sending them directly, then sends the edge-coded bits along with structured data. The receiver can then piece together the original data word as a result [9]. Figure 3 depicts the development of an ECS packet.

Figure 4 is an illustration of ECS transmission. Let us use a concrete example to better illustrate how ECS [9] works. A case in point is the transmission

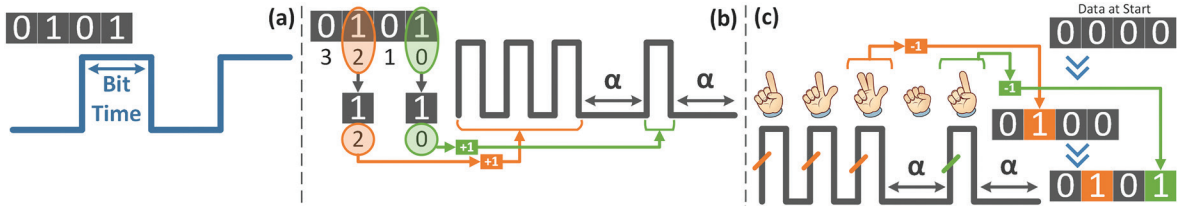

Figure 2: (A) Normal serial transfer; (B) Edge-coded transmitter; (C) Edge-coded.

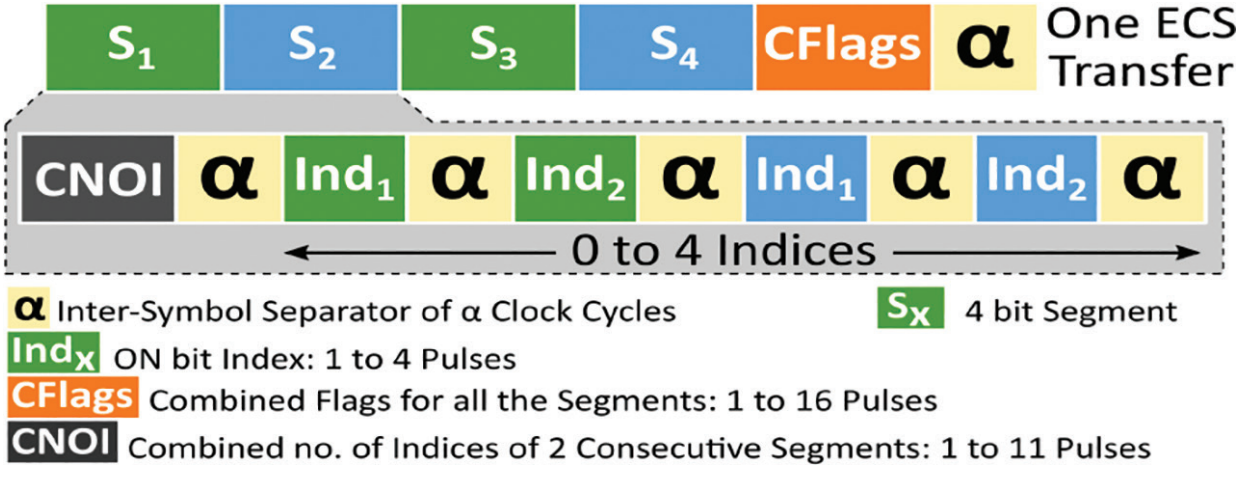

Figure 3: ECS packet formation. ECS, edge coded signaling.

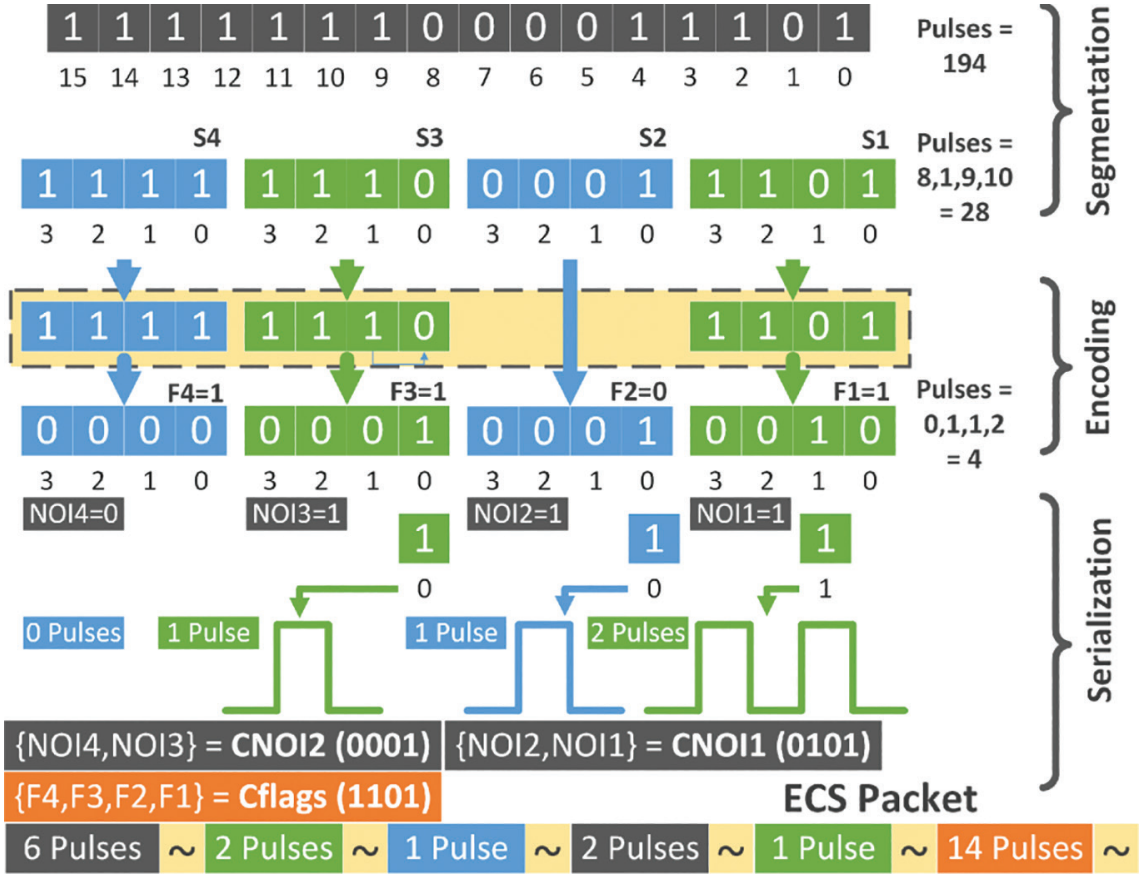

Figure 4: Example: ECS packet formation. ECS, edge coded signaling.

of the decimal number "267," which in binary representation is "0000 0001 0000 1011." N = B/l segments are produced when ECS first splits the B-bit data word into segments of size l. The four 4-bit segments in this example are S1 = 1011, S2 = 0000, S3 = 0001, and S4 = 0000. B is set to 16 and l–4. Subsequently, each segment undergoes encoding. The ECS encoding process involves flipping the segment bits if the count of active bits surpasses half of the segment size (i.e., >l/2). A Flag bit is assigned to each segment, indicating whether the segment is encoded. The four segments are concatenated to form a collective 4-bit flag, denoted as CFlags = (Flag4, Flag3, Flag2, Flag1). Additionally, a 2-bit NOI (Number of ON bits) flag is employed to specify the count of active bits in an encoded segment. The NOIs of two consecutive segments are combined to produce two collective 4-bit NOIs: CNOI1 = (NOI2, NOI1) and CNOI2 = (NOI4, NOI3).

In this illustration, only segment S1 meets the encoding criterion (3 > l/2). Consequently, the bits in S1 are flipped, resulting in the modified segment S1 = 0100. Simultaneously, the corresponding flag bit, Flag1, is raised to indicate the inversion. Upon completion of the encoding procedure, the collective flags are represented as CFlags = 0001. The counts of active bits would be NOI1 = 1, NOI2 = 0, NOI3 = 1, and NOI4 = 0, resulting in CNOI1 = 0001 and CNOI2 = 0001. Subsequently, the ECS transmitter proceeds to choose only the active bits within each encoded segment.

The ECS transmitter then precisely identifies the bits that are active in each encoded segment. This entails choosing index numbers 2 from S1 and 0 from S3, as shown in our case. Next, all the data elements are put together to form a packet, as seen in Figure 3. As shown in Figure 5, each packet component is then delivered as a pulse stream, with the pulse count matching the decimal number that the component represents. It is crucial to remember that ECS adds one extra pulse (such as CNOI + 1 or index + 1) to each pulse stream. In the case of indices, this extra pulse is essential for managing the encoding of index 0. If the bit at index 0 is active, no pulse would be conveyed without it. Similar to this, when the count of active bits in the

first two segments is zero, the extra pulse for CNOIs tells the receiver not to expect any index number. To signal zero content in CFlags, for example, when no segment passes the encoding inversion, an extra pulse is needed. As shown in Figure 5, each pulse stream is separated by an ideal inter-symbol delay of $\alpha = 4$ clock cycles [9]. To determine every element in the packet, the rising or falling edges of each pulse stream are totaled at the receiving end. After then, the received data is used to reassemble the broadcast data by decoding. ECS is dynamic because it can provide variable data rates based on the pulse counts of even two consecutive data words.

VI. Existing DDR-ECS Transmission

The transmission and reception process of the standard ECS, as explained in the preceding section, utilizes only one pulse edge for counting pulses, leaving the other edge unused [10]. By contrast, the DDR-ECS capitalizes on these unused edges. DDR-ECS employs both the rising and falling edges to convey an edge stream rather than transmitting a pulse stream. For instance, when transmitting the number "4," ECS requires four pulses as shown in Figure 6A, identified with their rising edges, while DDR-ECS necessitates two rising and two falling edges. The process of forming the packet remains the same as that for ECS.

The examples of DDR-ECS transmissions are illustrated in Figures 6B and 6C. It is evident that the transmission time in DDR-ECS is approximately halved, resulting in a doubling of the data rate.

VII. Proposed DDR-ECS

The proposed DDR-ECS architecture shown in Figure 7 mainly consists of transmitter and receiver architecture.

In the current ECS approach, the pulse count for each index is consistently increased by 1. This implies that each index is transmitted as a series of i + 1 consecutive pulses, where i represents the index number. Consequently, this results in an increase in the number of pulses and subsequently higher power consumption. However, the proposed method addresses this issue by employing Index coding to decrease the pulse count.

Figure 5: Standard ECS transmission (Data = 267). ECS, edge coded signaling.

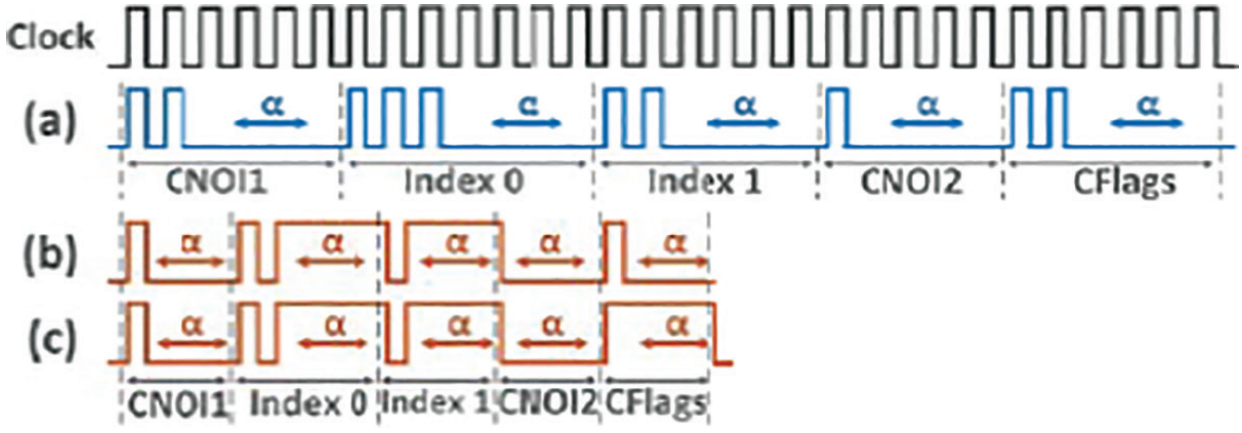

Figure 6: (A) Standard ECS transmission (Data = 267) (B) DDR-ECS transmission = (Data = 267) (C) DDR-ECS transmission (Data = 132). DDR, double data rate; ECS, edge coded signaling.

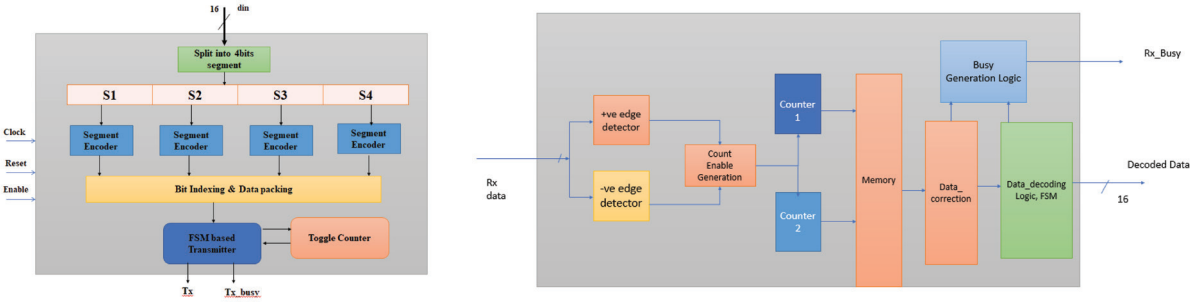

Figure 7: Proposed DDR-ECS architecture. DDR, double data rate; ECS, edge coded signaling.

The subsequent subsections will provide further details on the proposed ECS transmission and reception process.

VIII. DDR-ECS Transmission

The DDR-ECS transmission involves the following process:

a. ECS segmentation/bit splitter

The number of pulses needed for transmission increases dramatically, depending on the size of the data word, B, and the number of active bits in it. A greater pulse count is required for the expression of index numbers, especially for the most important bits. The worst-case situation, in which every bit is active, would require up to B (B + 1)/2 pulses. Nevertheless, the data rate rapidly decreases as the number of pulses increases, highlighting the necessity of reducing the pulse count. In order to overcome this difficulty, ECS divides the data word into smaller units, each with a size of l = 4 bits. This essentially caps the number of index numbers at 3. For example, as shown in Figure 8, if the objective is to send a 16-bit data with $B = 16$ and l = 4, the 16-bit data is split into four 4-bit segments. Figure 9 shows the simulation output of data segmentation for input 3EC8.

b. Encoding

With the expansion of the data word size, there arises a necessity for more inter-symbol separators to differentiate the pulse streams denoting the active bits. This increased use of separators has a notable detrimental effect on the data rate. In order to mitigate the negative effects of separators on data throughput, it is advantageous to decrease the number of active bits in each data segment. This is accomplished by applying a conditional bit-wise NOT operation to a particular segment using the ECS encoding technique. This procedure can only be performed if a segment's active bit count is greater than half of its total segment size. Table 2 shows how this procedure is carried out.

An encoder is a combinational circuit responsible for encoding data to minimize the number of ones. In the ECS encoding process, the bits within a segment are inverted when the count of active bits exceeds half the segment size (i.e., > l/2), as depicted in Figure 10. Each segment is associated with a Flag bit, used to signify whether the segment has undergone encoding or not, as illustrated in Table 1. To elaborate on the encoding strategy, let us consider $B = 16$. If the specified condition is met (i.e., the count of active bits surpasses l/2), the bits within a segment are inverted, and a 1-bit flag, Fi, is set to indicate the applied operation. The subscript i denotes the segment number. After processing each segment independently, four distinct flags, one for each segment, are generated.

Figure 11 shows the computation of Index codes and NOIs for the input 3EC8.

Algorithm for ECS Segmentation and Encoding Process

1: $S_1 = Data[3: 0]$, $S_2 = Data[7: 4]$
2: $S_3 = Data[11: 8]$, $S_4 = Data[15: 12]$
3: for i = 1 to 4 do
4: $NOI_1 = countONbits\ (S_i)$
5: $F_i = 0$
6: if $NOI_1 > l/2$ then
7: $S_i = \sim S_i$
8: $F_i = 1$
9: $NOI_1 = countONbits\ (S_i)$
10: end if
11: end for

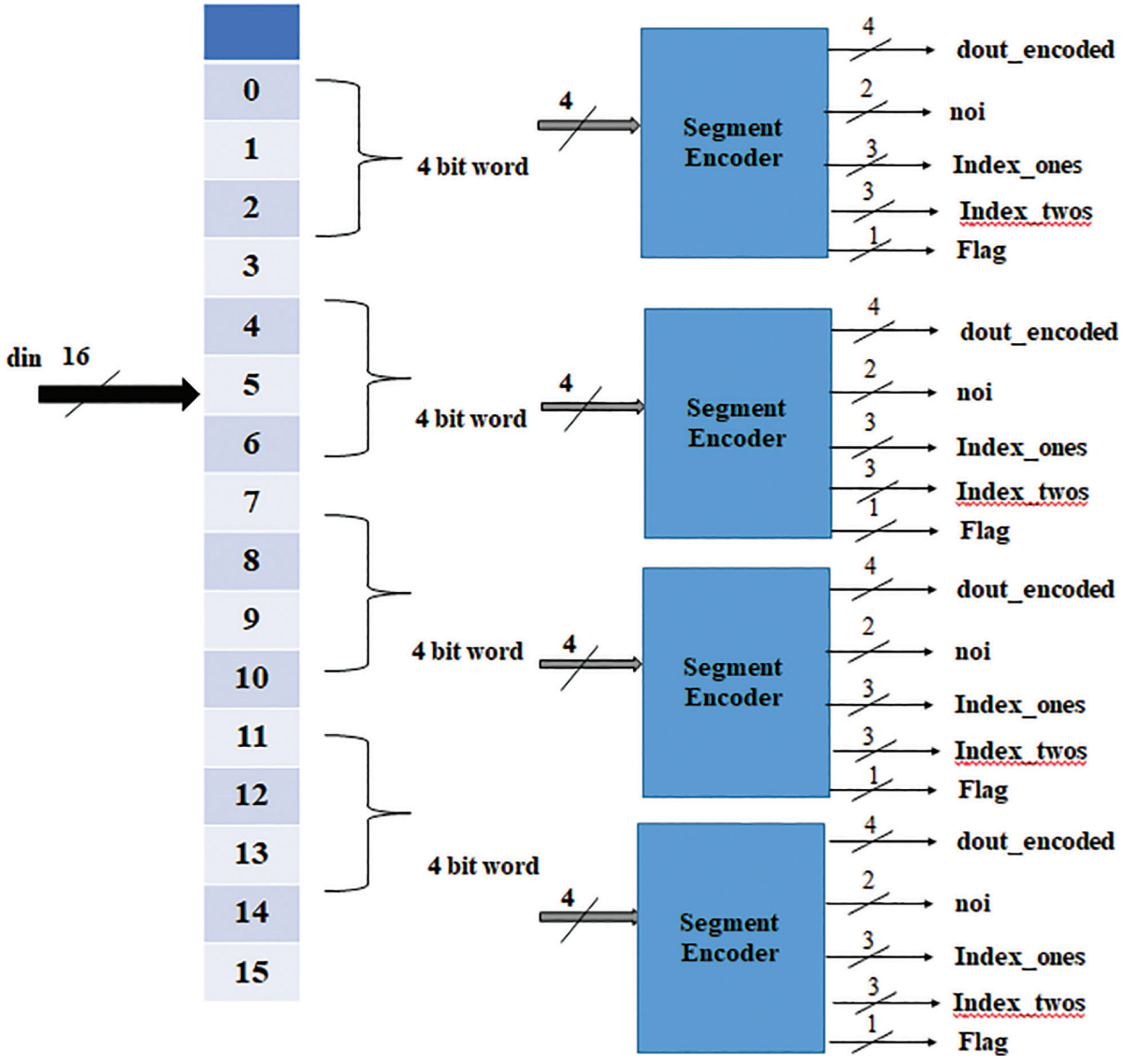

Figure 8: Segmentation/bit splitter.

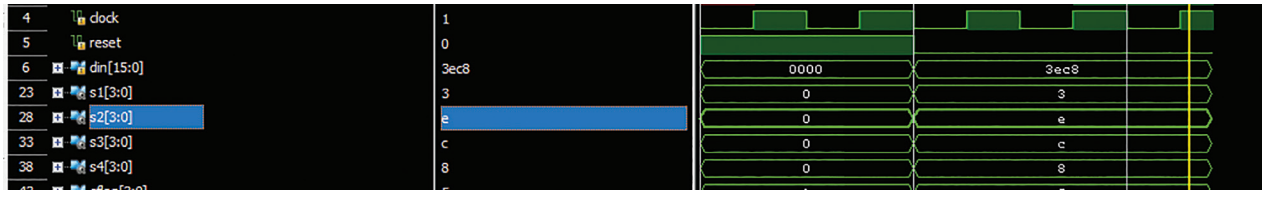

Figure 9: Simulation result showing data segmentation for input 3EC8.

c. Index coding

Index coding is a methodology employed in digital communication systems to enhance transmission efficiency by minimizing the number of pulses. In this approach, a series of binary symbols is translated into a sequence of pulses or waveforms suitable for transmission over a communication channel. The generation of this pulse sequence is contingent

Table 2: Encoded data and number of 1's

Regular data	Encoding/ inversion	Number of 1's	Flag
0000	0000	0	0
0001	0001	1	0
0010	0010	1	0
0011	0011	2	0
0100	0100	1	0
0101	0101	2	0
0110	0110	2	0
0111	1000	1	1
1000	1000	1	0
1001	1001	2	0
1010	1010	2	0
1011	0100	1	1
1100	1100	2	0
1101	0010	1	1
1110	0001	1	1
1111	0000	0	1

upon a pre-established codebook or index code. To generate pulses based on an index code, the following steps can be taken:

- **Index Codebook:** Comprising a collection of pre-defined pulse sequences, the index codebook assigns a unique pulse sequence to each distinct binary symbol.
- **Mapping Binary Symbols to Pulse Sequences:** The binary symbol sequence designated for transmission undergoes mapping to a sequence of pulse sequences sourced from the codebook. Each binary symbol is substituted with the corresponding pulse sequence derived from the index codebook.
- **Generating Pulse Sequence:** Following the mapping of the binary symbol sequence to pulse sequences, the concrete pulse sequence intended for transmission is generated. This involves concatenating the pulse sequences associated with each binary symbol in the mapped sequence.
- **Transmitting Pulse Sequence:** Subsequently, the generated pulse sequence is transmitted over the communication channel.

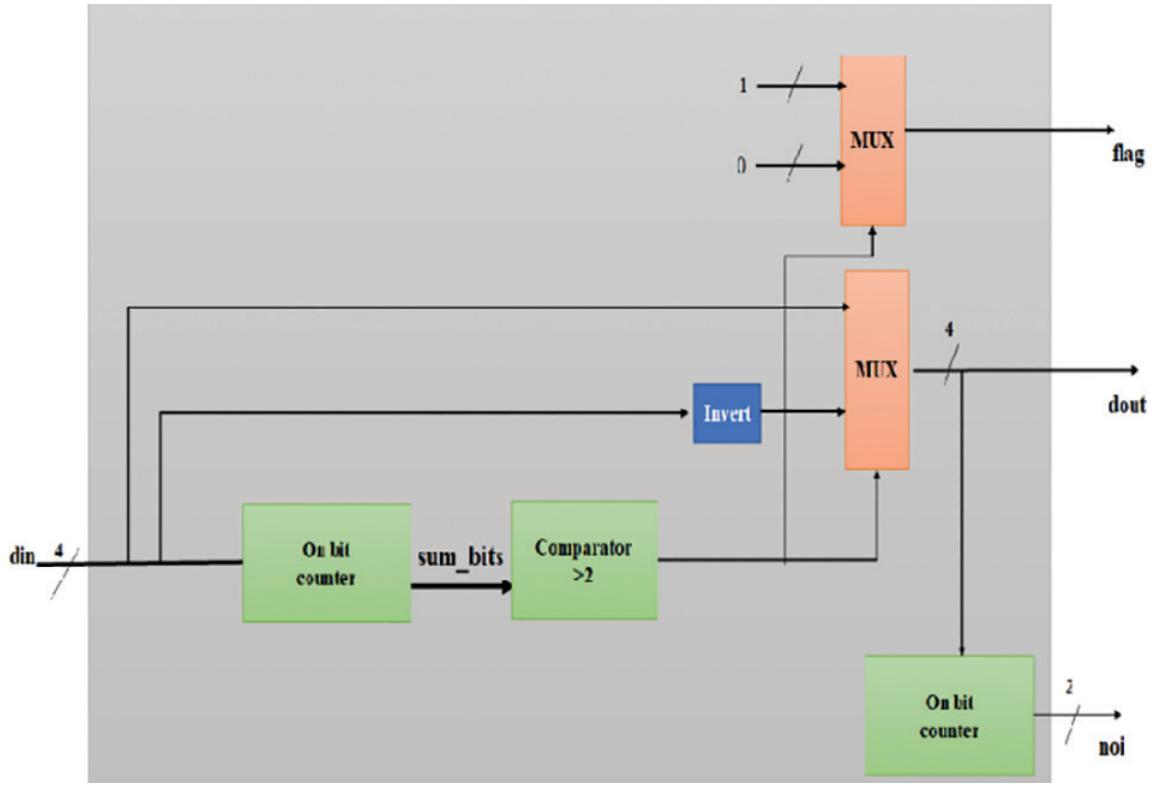

Figure 10: Segment encoder.

Figure 11: Simulation results showing Index codes and NOIs.

Table 3: Index coding

Possible data after encoding	No. of pulses (existing) [8, 9]	Number of 1's	Index code	No. of pulses (proposed)
0001	1	1	001	1
0010	2	1	010	2
0100	3	1	011	3
1000	4	1	100	4
0011	3	2	001	1
0101	4	2	010	2
0110	5	2	011	3
1001	5	2	100	4
1010	6	2	101	5
1100	7	2	110	6

Table 3 gives the index code of encoded data for number 1's = 1 and number of 1's = 2. The index code varies based on the number of 1's in the given data. Figure 12 depicts how index code is generated for the four bit encoded data.

Utilizing index coding enhances transmission efficiency by enabling the transmission of multiple binary symbols within a single pulse sequence. This results in a reduction of the required pulses for transmission, as illustrated in Table 3, thereby diminishing the

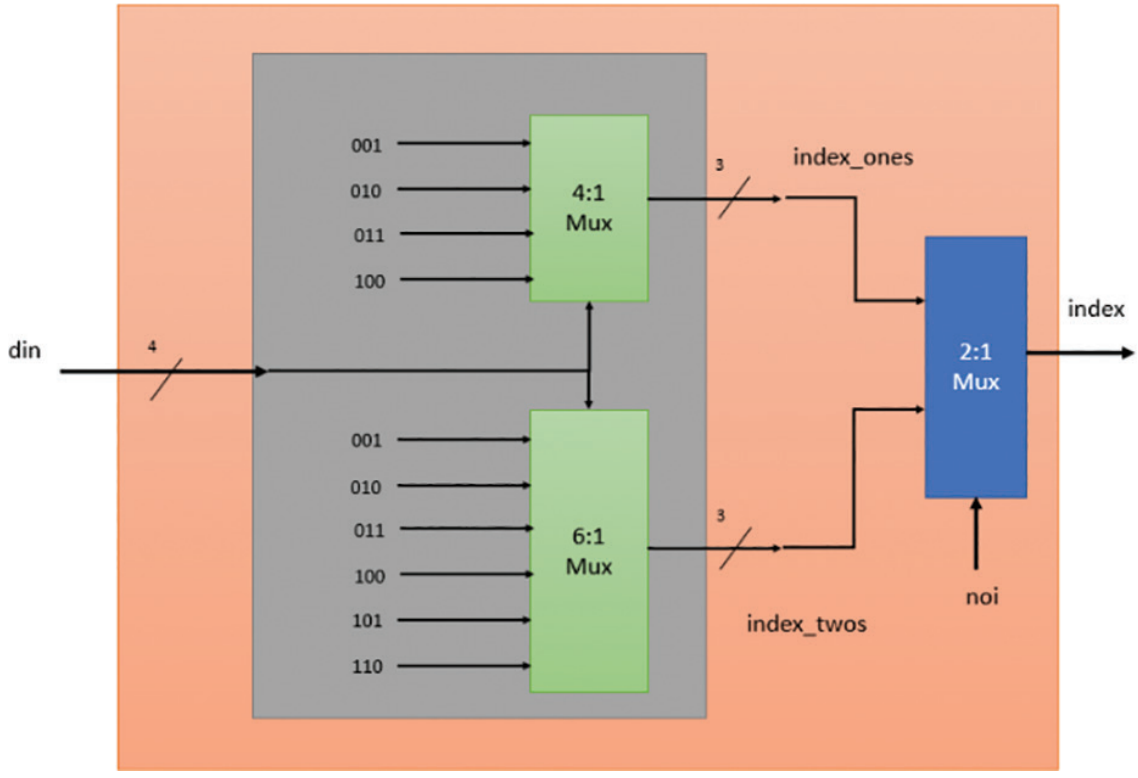

Figure 12: Index code generation.

Figure 13: Simulation result showing encoded data.

transmission time and bandwidth requirements. In summary, following the segmentation of 16-bit data, each segment encoder generates encoded data, a flag bit, NOIs, and an index code for the count of "1"s equal to 1 and 2.

Figures 13–16 show the final output of segment encoder for each of the 4 bits. It computes Dout encoded, NOI, Index ones, Index twos, and Flag.

d. Data packing

Following the segmentation and encoding process, all the informational components are packed together to form a packet. Each constituent of the packet is subsequently transmitted as a pulse stream, with the pulse count aligning with the decimal number represented by that specific piece. An optimal inter-symbol delay α separates all the pulse streams. Notably, the suggested ECS utilizes both rising and falling edges for transmitting an edge stream, deviating from the conventional approach of transmitting a pulse stream. The DDR-ECS packet, as proposed, is depicted in Figure 17.

where,

ON BITS- contains the information for the number of ON bits in the segment

Figure 14: Simulation result showing Index one output.

Figure 15: Simulation result showing Index two output.

Figure 16: Simulation result showing Flag bit.

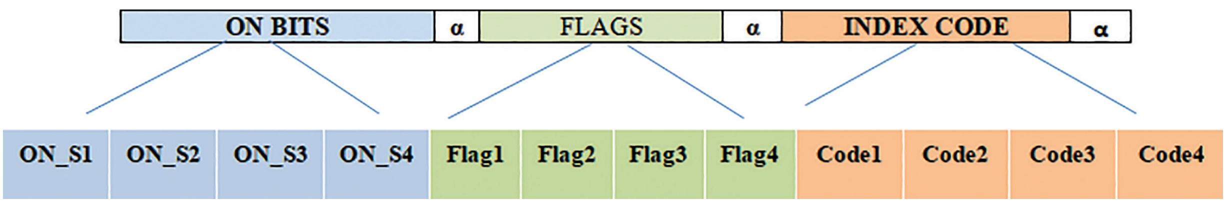

Figure 17: Proposed DDR-ECS packet. DDR, double data rate; ECS, edge coded signaling.

α - Inter-symbol separator of a α clock cycles
Flags- Flags will be set 1 if bits are inverted
Index Code - Index code for each segment as per Table 2.

As an illustration, consider the binary data sequences: 1100, 1000, and 0000. The count of active bits in these sequences is 2, 1, and 0, respectively. No encoding occurs for the sequence with 3 active bits, resulting in a flag of 0. Conversely, for the sequence 1110, encoding takes place, transforming the data to 0001, and the flag bit is set to 1. The transmission of flags serves to notify the receiver about the encoding

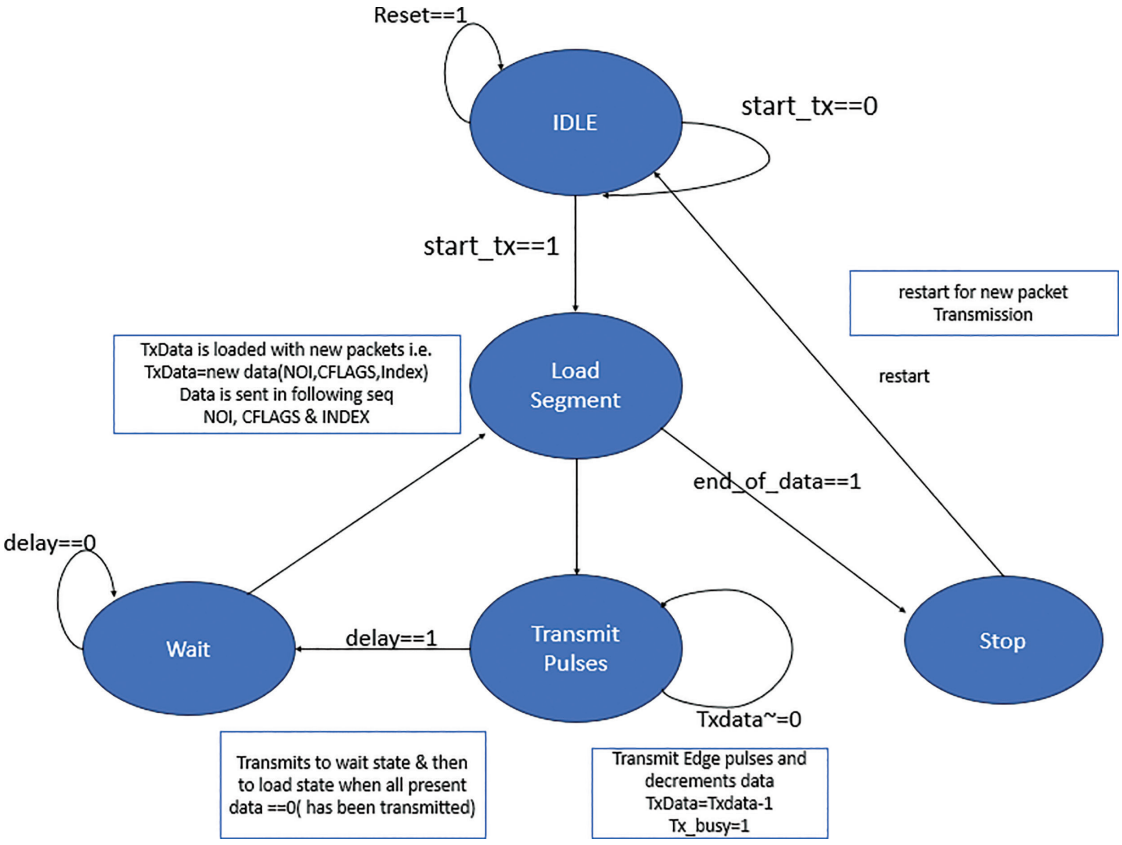

Figure 18: FSM implementation of proposed DDR-ECS transmitter. DDR, double data rate; ECS, edge coded signaling.

process. To commence transmission, the ECS transceiver initiates a series of pulses, followed by a four-clock cycle inter-symbol separator α. Subsequently, the transmitter dispatches multiple pulse streams, concluding each with α. The pulse count within each stream corresponds to the count of "ON" bits in the segments. This transmission process is repeated for the other segments. After transmitting all segments, the flags are conveyed. Similar to previous steps, they are sent as a pulse stream followed by α, with the pulse count equal to Flags + 1. Finally, the index code (Table 2) is applied to each segment for the ultimate packet formation, and pulses are generated for transmission.

e. Finite state machine

The primary role of the FSM is to oversee the data and operation flow. It is tasked with selecting the suitable design flow for all modules and receives a 16-bit data input for potential transmission. Additionally, the FSM conducts tests on the encoders to encode segments and guides other circuits in progressing edge computations for the designated transmitter. Upon achieving each edge stream, there exists the potential to introduce an inter-symbol delay, optimizing the transmitter setup for subsequent iterations.

Consequently, the FSM-designed transmitter functions as a system utilizing a state machine to govern the data and operation flow. It ensures adherence to the appropriate design flow for all modules, encoding segments, advancing edge computations, and introducing inter-symbol delays for effective transmission in subsequent iterations, as depicted in Figure 18.

f. Toggle counter

A toggle counter is a circuit designed to tally the occurrences of a toggle switch being flipped or toggled between its two positions (on and off). Typically, the operation of a toggle counter involves linking the toggle switch to the input of the counter, and with each toggle of the switch, the counter increments its count by 1. In the context of ECS, a toggle counter serves the purpose of generating pulse edges, denoting

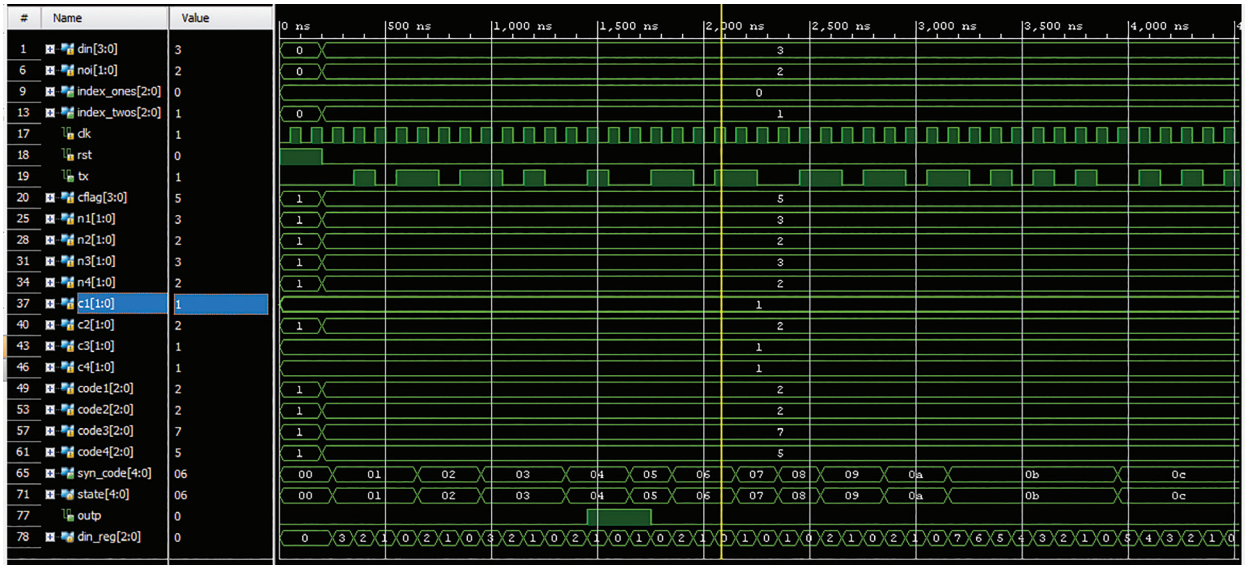

Figure 19: Simulation result showing subsequent iterations.

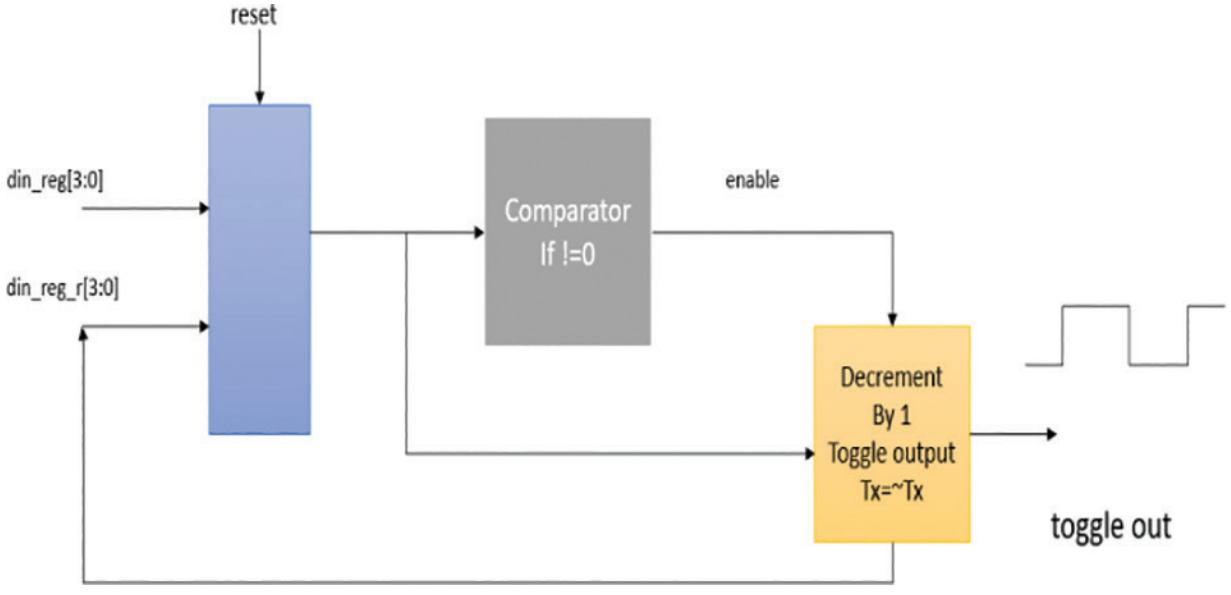

Figure 20: Toggle counter.

transitions from one logic level to another. Essentially, a toggle counter produces a sequence of pulses, each spaced a fixed number of clock cycles apart.

This is accomplished by connecting the counter's output to a flip-flop and subsequently linking the flip-flop's output back to the counter's input. In this configuration, each activation of the toggle switch results in the counter incrementing by 1, and the flip-flop's output changing state, thereby generating a pulse. Upon the subsequent activation of the toggle switch, the counter increments by 1 again, but this time, the input to the counter assumes the opposite state from the previous cycle due to the feedback from the flip-flop, as shown in Figure 20. Consequently, the counter counts up to the next even number and then backs down to the original value, generating another pulse. Toggle counters find utility in various applications where it is essential to monitor the number of

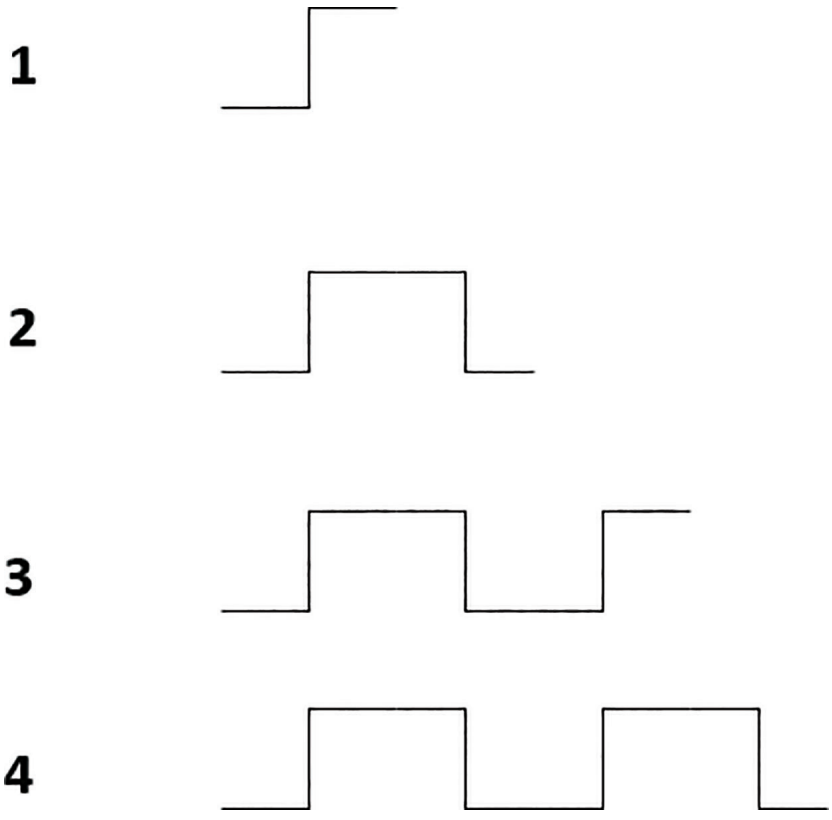

Figure 21: Example of DDR-ECS pulse generation using toggle counter. DDR, double data rate; ECS, edge coded signaling.

times a toggle switch is flipped. Such applications include industrial automation, security systems, and laboratory equipment.

As a result, the toggle counter transmits data in the form of transitions, denoted as Tx (Transmission pulses), as depicted in Figure 19. For instance, if the input is 1, only one transition is transmitted, and if the input is 2, two transitions are sent, as illustrated in Figure 21.

In the transmission phase of ECS, a busy signal is employed to indicate that the transmitter is presently incapable of transmitting data. The busy signal generation circuit within the transmitter employs various techniques to identify instances when data transmission is not possible. For instance, if the transmitter's buffer is at full capacity and cannot accommodate new data, the busy signal generation circuit detects this condition and generates a busy signal, signifying the current unavailability of the transmitter for data transmission.

The busy signal generation circuit may also serve to prioritize data transmissions based on factors such as importance or urgency. For example, when multiple data packets are queued for transmission, the circuit may prioritize the transmission of higher-priority packets, generating a busy signal for lower-priority packets that must await their turn. Once the busy signal is generated, it is transmitted alongside the data signal to the receiver. The receiver can then utilize the busy signal to ascertain when the transmitter is ready to accept new data for transmission. Figure 22 shows the simulation result of DDR-ECS Transmitter for the input 3EC8.

Algorithm for DDR-ECS Transmission

1: for j = 1,2 do
2: sendPulsesWithSeparator(*NOIj* + 1, α)
3: sendPulsesWithSeparator(*Flags* + 1, α)
4: for each ON bit in S_{2j-1}
5: sendPulsesWithSeparator(*Index code*, α)
6: end for
7: for each ON bit in S_{2j}
8: sendPulsesWithSeparator(*Index code*, α)
9: end for
10: end for

Figure 22: Simulation result showing generation of transmission pulse for the input 3EC8.

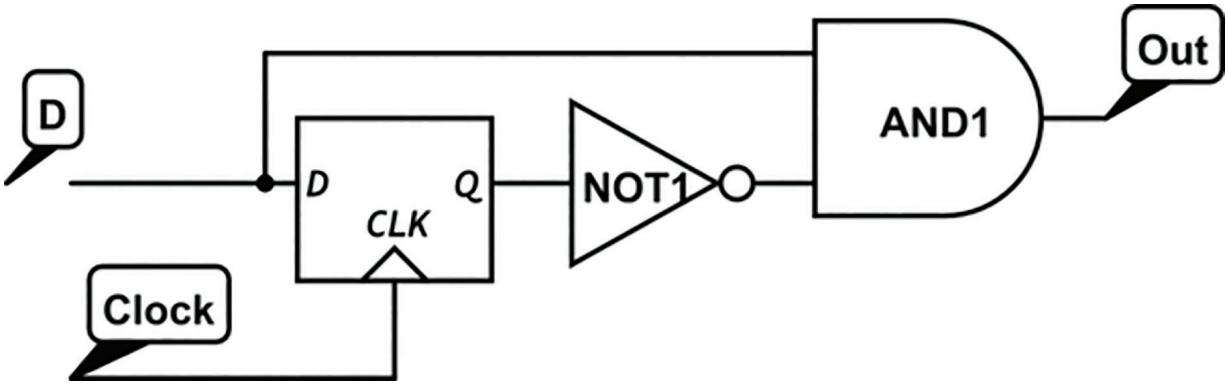

Figure 23: Positive edge detector.

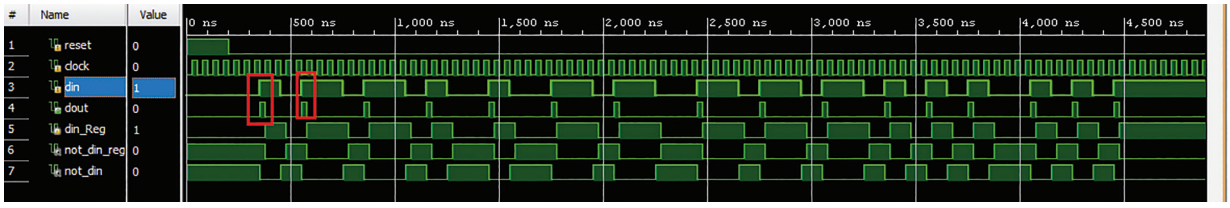

Figure 24: Positive edge detection.

IX. DDR-ECS Reception

The proposed ECS receiver mainly consists of important modules, namely, Edge Detector, Counter, Memory, Data Decoding, and Finite state-machine.

a. Edge detectors

Within ECS, the identification of edges in the received signal, representing encoded data, relies on the use of positive edge detectors. These detectors are specifically engineered to discern solely the rising edges of the signal, mitigating the risk of false detections caused by noise or interference. Precise edge detection holds paramount importance in ECS, as any errors in detecting the edges could result in inaccuracies during data recovery. Positive edge detectors excel in swiftly and accurately detecting edges within the received signal. Figure 23 and 24 shows simple circuit and simulation result of Positive edge detector respectively.

In ECS, the role of a negative edge detector is to identify the falling edge within the received signal, indicative of encoded data. Negative edge detectors are engineered to exclusively detect the falling edge of the signal, minimizing the likelihood of false detections caused by noise or interference. Precise edge detection is pivotal in ECS, as errors in detecting edges can lead to inaccuracies during data recovery. Negative edge detectors complement positive edge detectors, enabling the detection of both rising and falling edges within the signal. Moreover, negative edge detectors can also regenerate the signal by reshaping the edges of the received signal. This reshaping process enhances signal quality and mitigates the impact of noise and interference. Figures 25 and 26 shows simple circuit and simulation result of Negative edge detector respectively.

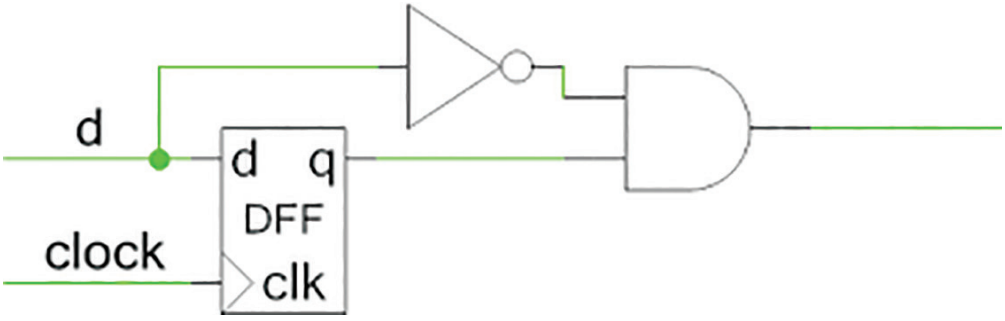

Figure 25: Negative edge detector.

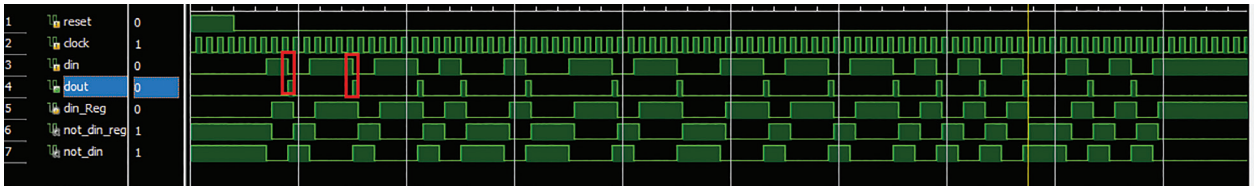

Figure 26: Negative edge detection.

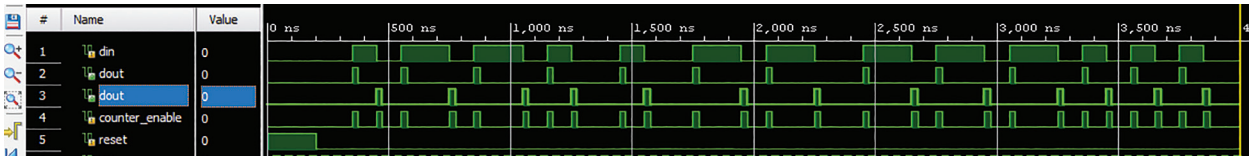

Figure 27: Counter enable generation.

b. Count enable generation

Count enable generation serves as a mechanism that allows a counter to tally pulses only when necessary, essentially providing a means of regulating the counting process within a counter circuit. In ECS, count enable generation specifically involves the creation of a control signal that activates or deactivates the counting function in a digital circuit based on the edge of a clock signal. Typically, a count enable signal is generated by identifying the rising (or falling) edge of a clock signal and using it to enable or disable the count input of a counter circuit. This ensures that the counter engages in counting only when the clock signal edge is detected, contributing to error reduction in the count and enhancing the overall accuracy of the system. Figure 27 shows the simulation result of count enable generation where it OR's both positive and negative edges.

c. Detection of edges using digital counter

Counter 1: In the realm of ECS, counter 1 is a digital circuit designed to enumerate the occurrences of rising edges (or falling edges) within a signal, generating an output signal that mirrors the count. Figure 28 shows simple representation of rising and falling edge of a pulse. The counter operates by identifying each rising edge in the input signal and incrementing a binary counter accordingly. The counter's output takes the form of a binary number, serving as a representation of the tally of rising edges. This output signal holds versatility, finding applications such as regulating the timing of other circuits or gauging the frequency of the input signal. For instance, in a digital clock system, an up counter may be employed to track the count of seconds, minutes, and hours. With each occurrence of a rising edge in the input signal (typically

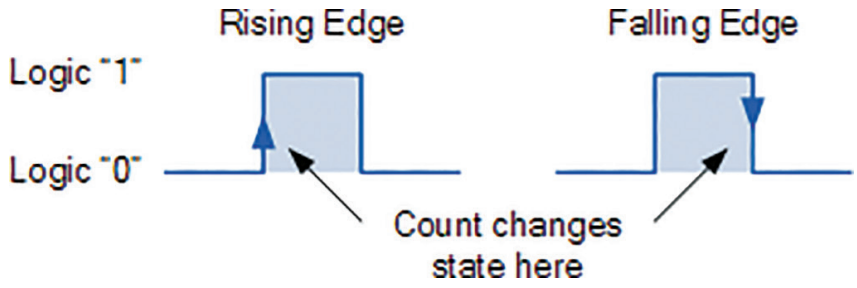

Figure 28: Rising and falling edge of a pulse.

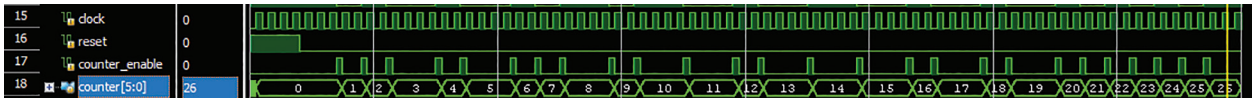

Figure 29: Simulation result of counter 1.

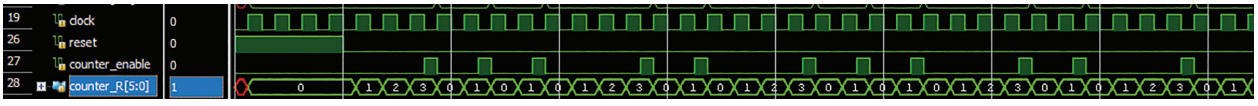

Figure 30: Simulation result of counter 2.

sourced from a crystal oscillator), the up counter increments its count. Subsequently, the output of the up counter undergoes decoding and is displayed on the clock face.

Figure 29 shows the output of counter 1; it is an enabled based synchronous counter which is incremented based on counter enable signal.

Counter 2: The application of this counter is commonplace in the receiver architecture of ECS, serving to extract the clock signal from the incoming data stream. Given that edge transitions, rather than signal levels, are utilized for data communication, it becomes imperative for the receiver to derive the clock signal from the incoming data stream for accurate data sampling. This counter plays a crucial role in generating a reference clock signal synchronized with the incoming data stream. In ECS, the information is conveyed through transitions in signal edges. Additionally, a counter 2 can be effectively employed in ECS to monitor and keep a count of the signal transitions that occur. Counter 2 is a free running counter, unlike counter 1; counter 2 resets on enable pulse.as it can be seen from the Figure 30, whenever enable pulse is high counter value is reset to 0.

d. Memory unit

In the receiver architecture of ECS, a memory unit is frequently utilized to store the received signal data before undergoing decoding. This memory unit serves the purpose of enabling the receiver to capture and retain the signal data for subsequent processing, particularly when the received signal exhibits a speed surpassing the processing capability of the receiver. Upon receiving a signal, it is captured and stored within the memory unit. Subsequently, the signal undergoes decoding through an analysis of transitions between high and low states. The decoding process may encompass comparing time intervals between these transitions with a reference time interval, commonly established by the system's clock frequency.

Figure 31 shows the simulation of memory unit. The input to this is the output of counter 1 and the address generation is based on the counter 2 value. So all the accumulated values in that timer interval are

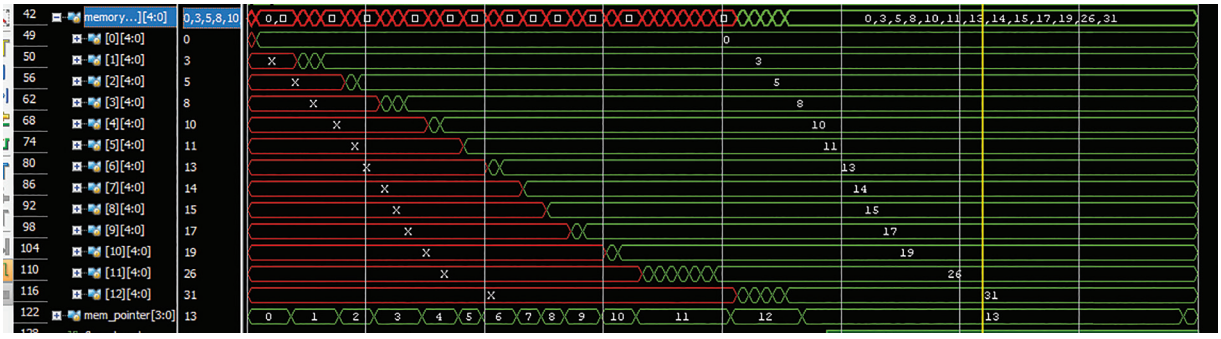

Figure 31: Simulation result of memory unit.

stored in the memory, which will be retrieved and decoded by the data decoder.

e. Data correction

Enhancing the accuracy of received signal data in ECS is achievable through incorporation of a data correction circuitry within the receiver. This circuitry is designed to identify and rectify potential faults that may occur during signal transmission. A common approach employed in data correction circuitry involves the utilization of error detection and correction codes. These codes are embedded into the signal during transmission, and the receiver leverages them to detect and rectify any conceivable transmission errors. The Hamming code, a frequently used example in digital communication systems, serves as an illustration of such a code. In the context of ECS, the presence of data correction circuitry holds a pivotal role within the receiver architecture. This circuitry empowers the receiver to identify and rectify errors present in the received signal data, thereby enhancing the accuracy and reliability of the transmitted information.

g. Acknowledging Logic in ECS receiver

Within the receiver architecture of ECS, a busy generation logic circuit can be employed to communicate to the transmitter when the receiver is incapable of receiving additional data. This situation may arise when the receiver is actively processing the received signal data, and the data buffer or memory unit has reached its capacity. The functioning of the busy generation logic circuit revolves around monitoring the memory unit dedicated to storing the received signal data. Upon detecting a full memory unit, the busy generation logic circuit generates a busy signal, which is then transmitted back to the transmitter. This busy signal serves as an indication that the receiver is currently unable to accommodate more data. Consequently, the transmitter can temporarily halt transmission until the receiver is prepared to receive additional data.

h. Data decoding

Within the receiver architecture of ECS, the data decoding process serves to retrieve the original transmitted data from the received signal. This process entails analyzing the transitions between high and low states within the received signal and leveraging this information to reconstruct the initial data. The precise methodology of the data decoding process is contingent upon the encoding scheme employed in the ECS system. Additionally, it may be influenced by the incorporation of error correction codes or other techniques aimed at enhancing the accuracy and reliability of the decoded data. Figure 32 shows the simulation output of data decoder.

Proposed DDR-ECS Receiver Algorithm

1: for i = 1 to 2 do
2: NOI_i = PulseStreamReceiver()*–1
3: $S_{2i-1} = S_{2i} = 0$
4: for j = 1 to NOI do
5: index = PulseStreamReceiver()–1
6: S_{2i-1}[index] = 1
7: end for
8: for j = 1 to NOI do
9: index = PulseStreamReceiver()–1
10: S_{2i} [index] = 1
11: end for
12: end for
13: Flags = PulseStreamReceiver()–1
14: Data = {S4 ⊕ {4{Flags[3]}}, S3 ⊕{4{Flags[2]}}, S2 {4{Flags[1]}}, S1 ⊕ {4{Flags[0]}}}
[* PulseStreamReceiver() is the pulse counter for each input pulse stream]

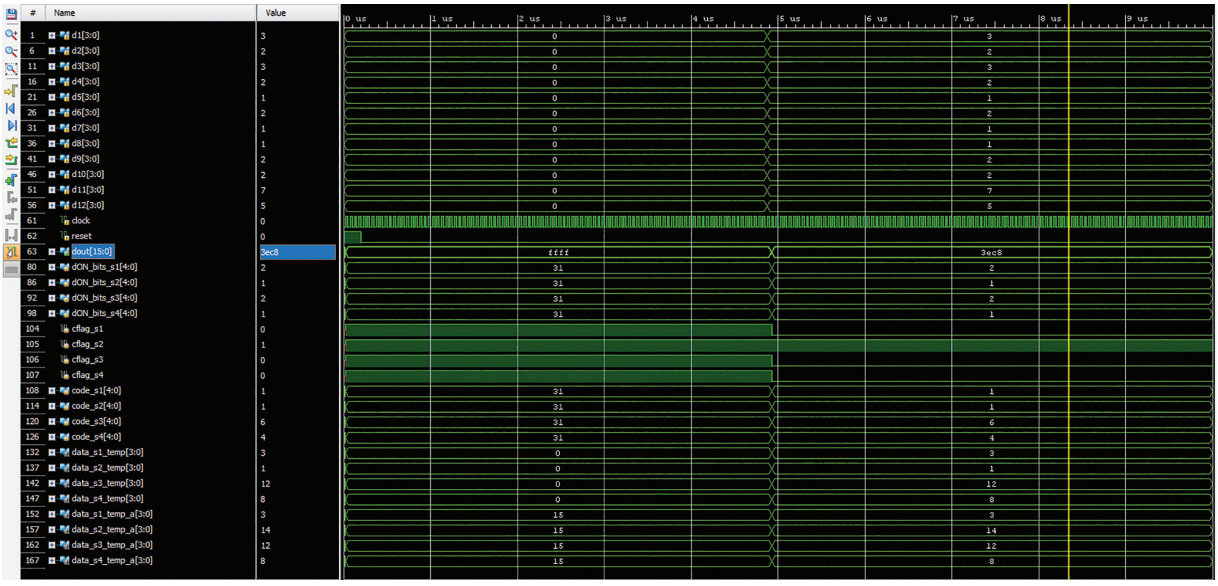

Figure 32: Simulation result of data decoder.

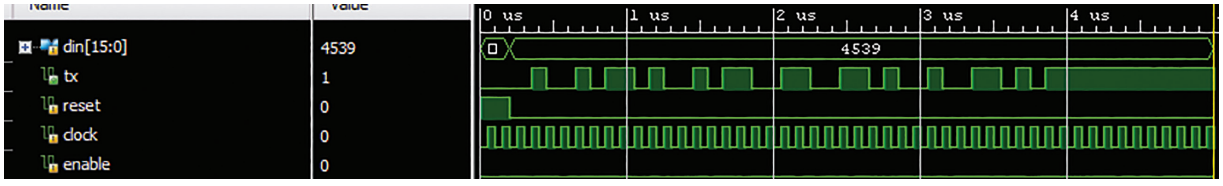

Figure 33: Simulation output of 16-bit DDR ECS transmitter module with data input 4539.Double data rate ECS.

X. Results and Discussion

a. Simulation

The designed DDR-ECS architecture was developed using Verilog. The Verilog hardware description language is also used to implement the submodules of this architecture. The speed of the clock signal was considered as 30 MHz and is managed at the coupled ends of the communication designed link. The complete DDR-ECS architecture was simulated using the Cadence Incisive tool for 16-bit data, with the transmission of the number of bits to the next module also being observed. Finally in order to know the complete design, the implementation of the synthesis was done through the Genus tool for the optimization power, area, and delay which is suitable for the IoT device. The simulation results below represent the transmit information based on segment bits, as well as the entire design component of the transmitter synthesis results.

Consider that we need to transmit a 16 bit data 4,539 (decimal equivalent). Next, DDR-ECS transmitter breaks the data into four segments and undergoes encoding process. A flag bit will be set if segment is encoded. Then NOI (number of ON indices) is identified in the segments and incremented. Finally all the information pieces are packeted together to construct a packet. Each piece in the packet is then transmitted in the form of a pulse stream where each pulse stream is separated by inter-symbol delay α. Pulse equivalent to 4,539 is generated as shown in Figure 33. Using decoding process, an original transmitted data 4,539 is obtained at the ECS receiver as shown in Figure 34.

b. Synthesis

Moore's law says that the performance of an integrated circuit (IC), including the number of components on it doubles every 18–24 months with the same chip price. For Moore's law to continue

Figure 34: Simulation output of 16-bit DDR ECS receiver module with data input 4539. Double data rate ECS.

Table 4: DDR-ECS transceiver synthesis results using 65 nm technology

Name of the module	Power (μW)	Area (μm²)	Data rate (Mb/s)
Proposed 16 bit DDR-ECS	≈13	1,755	12–73.5
DDR-ECS [10]	≈19	1,943	7.8–44.4
ECS [9]	≈19	≈2,098	4.2–26.7
PDC [4]	≈25	≈2,150	4.8–12.9
PIC [3]	≈26.6	≈2,356	3.1–8.5

DDR, double data rate; ECS, edge coded signaling; PDC, pulsed decimal communication; PIC, pulsed index communication.

pragmatically valid, new process technologies must provide more than the projected increase in density and chip capacity. Optimizing all the Power, Area and Performance at a time is an impossible task. So, based on specifications given by the customer, the flow is tweaked to achieve the required specifications. The Power, Area and Timing report of the proposed DDR-ECS transmitter is shown in Table 4.

XI. Conclusion

To sum up, the suggested DDR-ECS is a unique single-channel communication protocol that is well-suited to satisfy a variety of requirements, such as secure transmission, high data rates, low power consumption, and dependability, especially for direct device-to-device communication among constrained edge nodes. One important benefit of DDR-ECS is that it can significantly lower silicon area and power consumption (13 μW) because it does not require power- or space-intensive circuits for CDR. This efficiency is made possible by the fact that DDR-ECS ignores pulse width in favor of counting the rising edges of transmitted pulses for packet receipt and decoding. Resilient to clock skews, jitters, and fluctuations, DDR-ECS combines the best features of PIC, PDC, and ECS in terms of data rate (12–73.5 Mbps), dependability, packet security, and power efficiency. Across the summary, DDR-ECS is the recommended choice for limited devices across a range of applications, such as body-area networks, implanted devices, wireless sensor nodes, and tiny transceivers. Additionally, DDR-ECS's usefulness goes beyond traditional radio frequency communication to include other media like photonics, infrared,

and visible light, providing flexible applications in a variety of contexts. By contrast, the suggested DDR-ECS, essentially doubles the data rate without causing a major increase in the power consumption or area requirements of ECS, which is a significant improvement over normal single-ECS.

In conclusion, DDR-ECS proves to be an excellent choice for constrained devices in various applications due to its innovative data transmission approach, which offers significant benefits in power efficiency, reliability, and versatility. Its ability to operate across different communication media further enhances its applicability and potential impact. As the IoT landscape evolves, the demand for efficient, reliable, and secure communication protocols will grow. DDR-ECS is a promising solution that addresses these needs, paving the way for more advanced and capable IoT systems.

The proposed architecture leverages Edge technologies and design principles to balance power efficiency and high data throughput. By addressing the unique characteristics of IoT environments, such as intermittent connectivity, limited computational resources, and diverse application scenarios, our approach offers a comprehensive solution for improved data transmission and reception in IoT systems. Rigorous testing and validation demonstrate this low-power architecture's potential to enhance IoT device performance, supported by detailed analysis and empirical data. This contribution advances IoT technology, enabling more efficient and effective deployments across various sectors.

Future Scope

Ongoing research and development of DDR-ECS are poised to drive further innovations in communication technologies, propelling the IoT era forward. The success and potential of DDR-ECS open exciting avenues for additional research and development. Future efforts could focus on refining and optimizing the protocol to improve its real-time performance, reliability, security, and scalability, ultimately leading to a robust platform capable of efficiently managing the vast amounts of data transmitted in an IoT-centric world.

One promising area for future exploration is integrating DDR-ECS with advanced machine learning algorithms for predictive maintenance and real-time analytics in industrial IoT applications. Additionally, investigating the integration of DDR-ECS with blockchain technology could enhance security and trust in IoT networks, addressing concerns about data integrity and unauthorized access.

Conflicts of Interest

The authors declare no conflict of interest and the information presented is free from any biases influenced by potential conflicts.

Author Contribution

All authors have made substantial contributions to the conception and design, acquisition of data, and analysis and interpretation of data. All were involved in drafting the manuscript and revising it critically for important intellectual content and gave final approval of the version to be published.

References

[1] Sayeed et al. "IoT Raspberry Pi Based Smart Parking System with Weighted K-Nearest Neighbours Approach ", Civil Engineering Journal (Iran) Vol.9 No.8 2023.

[2] X. Widmer and P. A. Franaszek. 1983. A DC-balanced, partitioned-block, 8B/10B transmission code. IBM J. Res. Dev. 27, 5 (Sept. 1983), 440–451.

[3] S. Muzaffar and A. Shabra, J. Yoo, and I. M. Elfadel. 2015. A Pulsed-index technique for single-channel, low-power, dynamic signaling. In *Proceedings of the Design, Automation and Test in Europe (DATE'15)*. 1485–1490.

[4] S. Muzaffar and I. M. Elfadel. 2017. A Pulsed-decimal technique for single-channel, dynamic signaling for IoT applications. In *Proceedings of the 25th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC'17)*. 1–6.

[5] S. Muzaffar, O. T. Waheed, A. Aung, and I. M. Elfadel. 2017. Single-clock-cycle, multilayer encryption algorithm for single-channel IoT communications. In *Proceedings of the IEEE Conference on Dependable and Secure Computing (DSC'17)*. 153–158.

[6] Chunjiang Jia, DingWu, I. Hawkins, and A. Forsyth. 2012. One-wire communication system for cryogenic converter control. In Proceedings of the 6th IET International Conference on Power Electronics, Machines and Drives (PEMD'12)0.1–5.

[7] Maxim Integrated Products Inc. 2008. Overview of 1-Wire Technology and Its Use. (2008).

[8] Freescale Semiconductors. 2008-09. UICC—Contactless Front-end (CLF) Interface, Technical Specification, Version 7.3.0. (2008–09).

[9] S. Muzaffar and I. M. Elfadel, "Dynamic Edge-coded Protocols for Lowpower, Device-to-Device

Communication," ACM Transactions on Sensor Networks (TOSN), 2020

[10] S. Muzaffar and I. M. Elfadel, "Double Data Rate Dynamic Edge-Coded Signaling for Low-Power IoT Communication", 2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration

[11] Jaewook Byun, Seong Hoon Kim, and Daeyoung Kim. Lilliput "Ontology-based platform for IoT social networks",In: Proc. of IEEE International Conference on Services Computing, pages 139–146, June-July 2014.

[12] Shi Dayu, Xu Huaiyu, Su Ruidan, and You Zhiqiang, "A geo-related IoT applications platform based on google map", e-Business Engineering (ICEBE), pages 380–384, November 2010.

[13] Jenq Muh Hsu and Chin Yo Chen, "A sensor information gateway based on thing interaction in IoT-IMS communication platform", In: Proc. of IEEE International Conference on Services Computing, pages 835–838, August 2014.

[14] Maxim Integrated Products Inc. (2008) Overview of 1- Wire Technology and Its Use. [Online]. Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/1796

[15] 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 onewire vehicle network",In: Proc. of Solid-State Circuits Conference, pages 228–231, September 1998.

[16] B. Alqahtani and B. AlNajrani, "A Study of Internet of Things Protocols and Communication," 2020 2nd International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2020, pp. 1-6, doi: 10.1109/ICCIS49240.2020.9257652.

[17] N. Aini and G. Wibisono, "Method Comparison for Increasing Data Rate on 5G-IoT Technology," *2021* International Conference on Artificial Intelligence and Computer Science Technology (ICAICST), Yogyakarta, Indonesia, 2021, pp. 129-134, doi: 10.1109/ICAICST53116.2021.9497815.

[18] M. Ibro and G. Marinova, "Review on Low-Power Consumption Techniques for FPGA-based designs in IoT technology," 2021 16th International Conference on Telecommunications (ConTEL), Zagreb, Croatia, 2021, pp. 110-114, doi: 10.23919/ConTEL52528.2021.9495970.

[19] N. Lata and R. Kumar, "Internet of Things: A Review of Architecture and Protocols," 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 2020, pp. 1027-1031, doi: 10.1109/DASA51403.2020.9317091.

[20] Kuzlu, M., Pipattanasomporn, M. & Rahman, S. 2015. Review of communication technologies for smart homes/building applications. Innovative Smart Grid Technologies-Asia (ISGT ASIA), 2015 IEEE: 1–6.

[21] Raza, U., Kulkarni, P. & Sooriyabandara, M. 2017. Low Power Wide Area Networks: An Overview. IEEE Communications Surveys & Tutorials.

[22] López, P., Fernández, D., Jara, A. J. & Skarmeta, F. 2013. Survey of Internet of Things technologies for clinical environments. Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on: 1349–1354.

[23] Samuel, S. S. I. 2016. A review of connectivity challenges in IoT-smart home. Big Data and Smart City (ICBDSC), 2016 3rd MEC International Conference on: 1–4.

[24] Gomez, C. & Paradells, J. 2010. Wireless home Automation networks: A survey of Architectures and technologies. IEEE Communications Magazine, 48(6).

[25] Azamuddin Bin Ab Rahman, R. J. 2015. Comparison of Internet of Things (IoT) Data Link Protocols.

[26] Martha Zemede, K. T. 2015. Explosion of the Internet of Things: What does it mean for Wireless devices? Key sight Technologies.

[27] Tabish, R., Mnaouer, A. B., Touati, F. & Ghaleb, A.M. 2013. A comparative analysis of BLE and 6LoWPAN for U-HealthCare applications. GCC Conference and Exhibition (GCC), 2013 7th IEEE: 286–291.

[28] Hughes, J., Yan, J. & Soga, K. 2015. Development of wireless sensor network using Bluetooth low energy (BLE) for construction noise monitoring. International Journal on Smart Sensing and Intelligent Systems, 8(2):1379– 1405.

[29] Aragues, A., Del Valle, P., Muñoz, P., Escayola, J. & Trigo, J. D. 2012. Trends in entertainment, home automation and e-health: Toward cross-domain integration. IEEE Communications Magazine, 50(6).

[30] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. & Ayyash, M. 2015. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys \& Tutorials, 17(4): 2347–2376.

[31] Frenzel, L. 2012. The Fundamentals of Short Range Wireless Technology. ELECTRONIC DESIGN, Oct, 11.