iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** V **Month of publication:** May 2020

DOI: http://doi.org/10.22214/ijraset.2020.5243

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

Application and Analysis of Approximate Bus and Shift-Invert Encoding Technique for Image Data Transfer over Serial Bus

V. Priyadarshini[1] , S. R. Malathi[2]
[1, 2]*Department of ECE, Sri Venkateswara College of Engineering, Tamilnadu, India 602117*

Abstract: ***Reduction of power utilization in computing devices has gained more attention recently. Several analysis works have centered on reducing power consumption within the off- chip buses as they consume a major quantity of total power. Since the bus power consumption is proportional to the shifting activity, reducing the bus shift is an effective way to cut back bus power. There are various techniques for reducing bus power in address buses compared to data buses. It has been proved that approximate serial bus encoding technique reduces the signal transitions over a serial bus to a significant amount. It is also found that shift-invert bus encoding technique reduces significant amount of signal transitions over parallel buses at the cost of additional lines for encoding. Here we have applied shift- invert coding for image data over serial bus where no additional bus lines for encoding are used and performance analysis between approximate bus encoding and shift-invert encoding technique is done on image data over a serial bus.***

Keywords: ***Low Power; Bus Encoding; Approximate bus encoding; Sensor data; Shift-invert coding***

I. INTRODUCTION

The last decade has seen a faster accretion of wearable and implantable devices like action cameras, health tracking bands, smart watches, and pacemakers. Contradictory to traditional computing platforms, the above mentioned devices squander important amounts of time reading sensors like microphones, image sensors, accelerometers, electrocardiogram (ECG) sensors, etc., and operating the detected data (Fig. 1). Frequent or continuous sensing poses a replacement challenge to energy potency as a result of information transfer from off-chip sensors consumes momentous energy. Disparate to on-chip buses that have benefited from enhancements in semiconductor development technology, off-chip buses, and especially serial interfaces that are normally used with sensors (e.g., Serial Peripheral Interface (SPI), InterIntegrated Circuit (I2C), MIPI Camera Serial Interface (CSI)), have improved far more slowly. As a result, up to thirteen percent of energy dissipation in such systems will be attributed to information transfer from off-chip sensors [1].

After all sensors convert attributes of the physical world, like light, temperature, or vibration, into a digital value, their outputs inherently add noise. Thus, the algorithms that develop afferent information are composed to tolerate noisy inputs. For example, little amount of noise within the readings from an accelerometer in a pedometer will not result in a significant error in the step count, which is what the user is ultimately interested in. Employing traditional off-chip serial buses for reading information from such sensors in primary amounts to carry noisy information accurately, which is arbitrary and devastates energy.

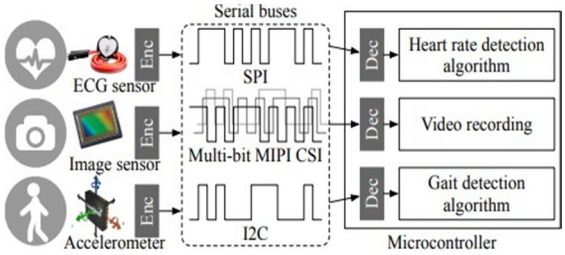


Fig. 1. Serial buses between a microcontroller and various sensors[3]

Approximate measuring applies the error resiliency of applications to avoid wasting energy by reclining the orderliness demand and manufacturing results that are just adequate [2]. Adopting this broad approach, approximate serial bus techniques are planned to conceal information in a lossy manner to scale back bit transition counts (TC), resulting in reduced shifting power in the serial bus [5], [8]. Approximate serial bus encoding techniques need to address three main challenges. First, the encoding scheme should

conceal as many information and values as possible using energy- efficient bit patterns, minimizing the transmission of high- energy cost bit patterns. Second, the encoding scheme should provide the application with a control knob for quality configuration as a result of the notion of a good enough result is application-specific and contextual. Third, the encoding and decoding got to be extremely energy-efficient to stop the energy overheads from nullifying the energy savings obtained from the bus.

A novel quality-configurable approximate bus known as AXSERBUS (Approximate Serial Bus)[5] handles the above mentioned challenges. AXSERBUS encodes data in three modes: (i) If the deviation between the current and previous data values is very low, the result is zeroed out, and a 0-TC (no intra-word TC) pattern is transmitted, and the receiver reuses the previous data value (this is similar to [6]). (ii) If the deviation is in average range (defined by adjustable thresholds), an approximate value of the differential is encoded as a 1-TC (single intra-word TC) pattern, which incurs minimal energy dissipation since it contains only a single bit-transition. The employment of approximate deviations permits a broader dimension of differences to be encoded in this energy-efficient encoding mode. An interesting property of this encoding scheme is that the error due to approximation scales with the magnitude of the differentials. (iii) Finally, if the deviation is high, we transmit the complete value and not the difference. The energy cost for transmission of the absolute value is relatively high because it may contain several signal transitions, but only a small fraction of data are encoded in this mode. So we can see that AXSERBUS is mainly based on its transition counts along with its encoding modes. The low difference encoding mode is not complex since it has only least counts it is not taken into account and the value is made to zero and given as output.

AXSERBUS does not require extra bits to indicate the encoding mode because the mode information is implicit in the TC of each codeword.

1) AXSERBUS achieves more than 80% reduction in TC compared to a traditional (exact) serial bus and up to 17.8% reduction in TC compared to approximate bus [3] given the same accuracy requirement.

2) The tradeoff between application-level quality and energy savings is demonstrated using an optical character recognition (OCR) application. AXSERBUS achieves 79.4% reduction in dynamic power dissipation whereas maintaining OCR accuracy above 95%[5].

II. LITERATURE REVIEW

Low-power bus encoding mainly aims at reducing TC on the bus, thereby reducing the dynamic energy consumed for switching highly capacitive off-chip interconnects. A typical approach to reduce TC is to exploit the temporal or spatial locality of the data. Gray encoding and T0 encoding are applicable when most of the data are consecutive, such as in memory address buses. However, these encoding schemes aim to reduce inter-word (word-to-word) TC on parallel buses. Modern low-power sensors predominantly employ a serial bus, where intra-word (bit-to-bit) TC is more relevant than interword TC, hence these parallel bus encoding schemes are not applicable.

Several serial bus encoding techniques have been proposed [1], [5], [11] which discusses about the performance of a tightly coupled on-chip buses connecting several embedded cores can be improved using multithreshold comparators at the receiver end. For example, Differential Bar Encoding (DBE) converts small differentials into a 1-TC codeword, where the number of consecutive 1's signifies the exact magnitude of the differentials. However, the energy savings of these encoding schemes is inherently limited because they are designed for lossless data transfer, which, as mentioned before, is often an overkill.

Shift-Invert encoding techniques[6], [15] deals with the communication between parallel buses in which logic switching is a significant factor in system power consumption. These schemes are also better suited for buses with arbitrary widths. There is no need of shifting data left or right individually. This combines all features together and runs the system without any error. In future, data reversal technique can be implemented in it to investigate the merits of this technique by generating other coding schemes resulting from a combination of the shift and invert operations.

Approximate Differential Encoding (ADE) and Serial T0

[8] are differential encoding schemes that exploit the locality of the data. In Serial T0, a special code with no intra-word TC represents a negligibly small differential from the previous value, and exact values are transmitted only if the differential is significant. However, the tradeoff between energy and quality is not fully exploited due to the single threshold approach wherein differentials are either zeroed out (incurring no energy cost) or accurately transferred (incurring a high energy cost).

Data encoding techiques using adders details about the precise adders that take along delay and large power consumption to obtain accurate results. Compared to the conventional ripple carry adder and carry-lookahead adder, these adders with block size of 4 reduces power-delay product by 66% and 32%, respectively, for a 32-bit addition [4].

A novel method [7] is introduced for designing approximate circuits by fabricating and exploiting false timing paths, i.e., critical paths that cannot be logically activated. An adaptive power gating technique is applied locally to a 32-bit Kogge Stone adder, and evaluated at the 16 nm FinFET technology node. This high granularity adaptive power gating approach employs a local controller to lower energy use and reduce circuit overhead[10]. In [13], the authors present a radix-4 static CMOS full adder circuit that reduces the propagation delay, PDP, and EDP in carry-based adders compared with using a standard radix-2 full adder solution. A novel gate-level strategy for designing Carry-Select adders is proposed. The strategy is more general than the previously proposed techniques, and accounts for the dependence of multiplexer delay on its fan-out[14].

III. SHIFT-INVERT CODING [15]

A. Bus Invert Coding

The basic idea behind Bus Invert Coding originated by noting that a lot of power is wasted during data transmission in off-chip bus lines. This is because of the frequent change or shifting of high capacitance lines, and thus the power can be saved by reducing the count of deviations happening on these bus lines.

Let N be the number of transitions between the data D^{k+1} at time k+1 and the value on the bus B^k, (i.e) N represents the total number of bit-positions in which the new data and the existing value on the bus differ. Let $D^{k+1(INV)}$ be the inverted data. (i.e) for all i, $0 \le i \le w'$. The Bus Invert Coding technique chooses either D^{k+1} or $D^{k+1(INV)}$ to be transmitted on the bus. If $N \le N^{INV}$, then D^{k+1} is transmitted on the bus; otherwise, the inverted data $D^{k+1(INV)}$ is sent on the bus. In default Bus Invert Coding, $w' = w + 1$. For variations of Bus Invert Coding, w′ can be greater than w+1.

Recently, a number of techniques have been proposed for bus encoding. Most of these techniques either center on the original bus-invert coding scheme or assume some special data conditions.

B. Shift-Invert Coding

In shift-invert coding, the main idea is to optionally shift the data bits by one-bit position (either left-shift or right-shift) if the shifting reduces the number of bus transitions. We define the left- shift operation on a w-bit data as follows. Let D represent d_{w-1}, $d_{w-2}\ldots d_0$ represent a binary data string of width w.

$$d_i^{(LS)} = d_{i-1}, \qquad 1 \le i \le w',$$
$$d_0^{(LS)} = d_{w'-1}$$

(i.e) we perform a circular left-shift. This will guarantee that we do not lose any information from the original data. The right-shift operation is defined similar to the left-shift, as follows.

$$d_i^{(RS)} = d_{i+1,} \qquad 0 \le i < w'-1,$$
$$d_{w'-1}^{(RS)} = d_0$$

In any coding scheme based on bus inversion, the value of the i^{th} bit, bi on the bus will be either the data value d_i or $1-d_i$. Thus, for all i, $0 \le i < w'$,

$b_i = d_i$, uninverted bit (or)

$b_i = d_i - 1$, inverted bit

Consider the following example. The data string at any time instant k can be represented as, $D^k = d_{w-1}{}^k, d_{w-2}{}^k, \ldots d_0{}^k$. B^k is the data transmitted on the bus at time k. Note that the bit width w′ of the bus (i.e) the encoded data that gets transmitted on the bus) could be greater than w depending on the coding scheme used. B^k and D^{k+1} are the inputs whereas B^{k+1} is the outpt.

Let

B^k = 0010 1011 (assume a 8-bit bus)

and, the new data at time k+1,

D^{k+1} = 1000 1111

therefore, the inverted data

$D^{k+1(INV)}$ = 0111 0000

For this example, the number of transitions N between B^k and D^{k+1} is 5. N^{INV} is the number of transitions between $D^{k+1(INV)}$ and B^k. In the case of Bus Invert Coding, we would try to see whether it is beneficial (i.e) whether the number of 0 to 1 and 1 to 0 transitions are reduced) to send $D^{k+1(INV)}$ over the bus. The number of transitions N^{INV} between B^k and $D^{k+1(INV)}$ is 3. Since $N^{INV} < N$, in the Bus Invert Coding technique, $D^{k+1(INV)}$ will be sent

over the bus at time k+1.

When we left-shift the data D^{k+1} once as defined above, we denote the left-shifted data at time k+1 as $D^{k+1(LS)}$.

$D^{k+1(LS)}$ = 0001 1111

Comparing to,

B^k = 0010 1011

the number of conversions N^{LS} between B^k and $D^{k+1(LS)}$ is just 1, which is better than the 3 transitions that one gets from the inverted data $D^{k+1(INV)}$. Thus, in this case, it is clear that by sending the left-shifted data, we can reduce the number of transitions even further than the reduction obtained from sending the inverted data. The reason behind applying the shift operation is straightforward. By changing the information at time k+1, it is doable that the bit values might match in more places with the existing data on the bus at time k than if the data bits were either upturned or ported stable. The matching of bits in additional places implies fewer transitions, thereby giving a higher resolution.

The provided example clarifies how a left-shift procedure is preferred rather than inversion. Further more, one can formulate examples to show how a right-shift operation on the data lowers the number of shifts. Therefore, we terminate that one can further decrease the number of shifts either by carrying out a left shift or a right-shift process.

It is obvious that shifting left or right move will not always reduce the number of transitions. Confiding on the values of B^k and D^{k+1}, it is doable that either the inverted information of $D^{k+1(INV)}$ or could also be even the unmodified/original input of Dk+1 provides the smallest amount of transitions once sent on the bus.

For every new data that must be sent over the bus, we assess the changeover of N, $N^{INV,}$ N^{LS} and NRS between Bk and D^{k+1}, $D^{k+1(INV)}$, $D^{k+1(LS)}$, and $D^{k+1(RS)}$ respectively. The encoding that appears in the least number of changeover is preferred.

Note that the data that gets sent over the bus, Bk+1 can be one of D^{k+1}, $D^{k+1(INV)}$, $D^{k+1(LS)}$, $D^{k+1(RS)}$. Thus,we oblige to identify the bus with a pair of extra bits that denote the coding that was utilized. This can be accustomed to decode the bus value accordingly at the receiving end. Thus, in ShiftInvert coding, the width of the bus w′ = w + 2, where w is the width of the data vector and we use pair of extra bits as compared to one extra bit in default Bus Invert Coding.

C. Approximate Serial Bus Encoding [5]

The encoding modes based on the transmission is selected among the low difference (LD) encoding mode, intermediate difference (MD) encoding mode and high difference (HD) encoding mode based on ‘i’. The following subsections describe the encoding and decoding mechanisms in each mode. Δ is the sum of current difference and error of the previous transmission. D_0 is the minimum threshold value and D_m is the maximum threshold value.

1) *LD Encoding Mode:* If | Δ | is very small, i.e., $\Delta < D_0$, where D_0 is the zero-out threshold, we approximate Δ as zero. We call the error introduced by this approximation the zero-out error. In this case, Δ = 0 is encoded as a 0-TC codeword, which is the most low-energy codeword. In l-bit codewords, there exist two 0-TC codewords (all 1’s and all 0’s), and we use all 1’s to represent Δ = 0. This is preferable in some serial bus physical-layer standards that have asymmetric static power consumption where a bit 0 consumes more power than a bit 1 because the line is pulled low by an open-drain driver, such as in I2C.
2) *MD Encoding Mode:* If $D_0 \leq |\Delta| < D_m$, where Dm is the mode threshold, | Δ | is encoded as a 1-TC codeword. There exist 2(l − 1) 1-TC codewords, and l is the length of the codeword which is used to represent small (non-zero) differences in an approximate manner, with non-uniform approximation bounds. Let us assume $\Delta \geq 0$. If $\Delta < 2l-1$, Δ is approximated as

$$\Delta' = 2^{[\log 2\ \Delta]} + 2^{[\log 2[\Delta/2]]}$$

In other words, all Δ between 2r and 2r+1−1 are approximated as 2r + 2r−1, which is the median of the range. Then, Δ is encoded as a run of l − r bits of 1’s, followed by r bits of 0’s. In case that Δ < 0, | Δ | is encoded in the same manner, but the order of bit 1’s and bit 0’s is reversed. The error due to this approximation is called the approximation error. This error is added to the next δ for error compensation, where δ_i is the difference between current s_i and previous source transmission s_{i-1}. At the receiver side, t is decoded as and added to the previous s. Codewords with a greater number of bits 1’s are assigned to lower Δ because bit 1 is preferred for low power, as mentioned above.

3) *HD Encoding Mode:* If | Δ | is greater than the mode threshold (i.e., | Δ | ≥ Dm), the absolute value of s, instead of the difference, is encoded as a multi-TC (two or more intra- word TCs) codeword. If tc(s) is already two or more, the source data is not modified and transmitted as its binary representation, i.e., t = s, where t is the encoded word. However, if tc(s) is 0 or 1, it cannot be transmitted without modification because 0-TC and 1-TC codewords are reserved for the LD and MD mode, respectively. This collision of codewords could be avoided by adding extra bits to indicate the encoding mode, but this would

incur a heavy overhead. Instead, knowing that an additional minor error will not affect the overall quality and that the HD encoding mode will not be used frequently, we intentionally increase tc(t) to 2 by flipping a minimum number of bits so that the decoder can identify the encoding mode from tc(t). To minimize the value deviation, the last one or two least significant bits (LSBs) are flipped. The maximum error due to the bit flips is only ±2. No decoding is required at the receiver side because t is already the binary representation of the exact value or an approximate value of s.

IV. IMPLEMENTATION

AXSERBUS is validated by implementing hardware for the encoding and decoding logic. Figure 2 is the high-level hardware block diagram of the encoder and the decoder to show how the encoding and decoding modes are selected. We describe the detailed implementation of the major blocks.

A. TC Counter Block

The TC counter block is common in the encoder and the decoder, and it determines whether the input word w (s or t) is a 0-TC, 1-TC, or multi-TC word. Determining whether w has 0 TC or 1 TC is relatively simple since the 0-TC and 1- TC words are limited. Two 0-TC words, i.e., all 1's and all 0's, can be detected by AND and NOR gates, as shown at the bottom of Figure. The logic for detecting 1-TC words is shown at the top of Fig. LH, RL, LL, and RH stand for "left high", "right low", "left low", and "right high", respectively. They indicate whether an input word w has a specific bit pattern. For example, LH[3] is true if w starts with three 1's, i.e., if w is 111XXXXX, where X is a 'don't care'. Similarly, RL[5] is true if w ends with five 0's, i.e., XXX00000. Therefore, if both RH[3] and LL[5] are true, w is 11100000, which has a TC of 1. Other 1-TC words can be detected by applying the same method.

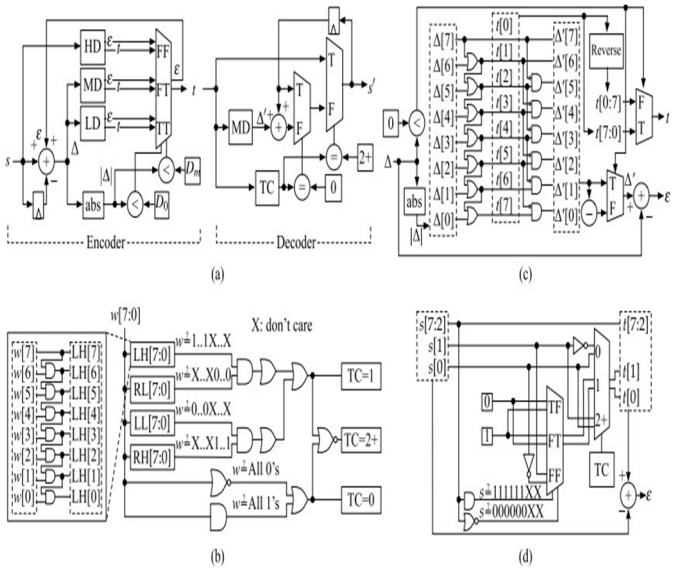


Fig. 2. (a) Encoder and decoder block diagrams, (b) TC block in the encoder and the decoder, (c) MD encoder block in the encoder, and (d) HD encoder block in the encoder [5]

B. LD Encoder Block

If $|\Delta| < D0$, the LD encoder block generates the 0-TC code as described in Section IV-A1. The codeword is always l bits of 1's, and the error is equal to $-\Delta$. On the decoder side, if the TC of the received codeword t is 0, it simply recirculates the previous s .

C. MD Encoder/Decoder Blocks

If $D0 \leq |\Delta| < Dm$, the MD encoder block generates a 1- TC code as described in Section IV-A2. Fig. 5(c) shows the logic to generate a 1-TC code t and the error for intermediate differences. First, a 1-TC codeword t is generated for | | using the cascaded OR gates that detect the most significant 1 and make all less significant bits to 1's. If $D0 < 0$, the bit order of t is reversed by flipping the vector (Reverse block in the Fig. 5(c). To compute the error, the encoder decodes t to using the cascaded AND gates that finds the most significant 1 and makes the next less significant bit 1, followed by 0's, as presented in Table II. The MD decoder block is identical to the part of the MD encoder. Note that is not maintained on the decoder side.

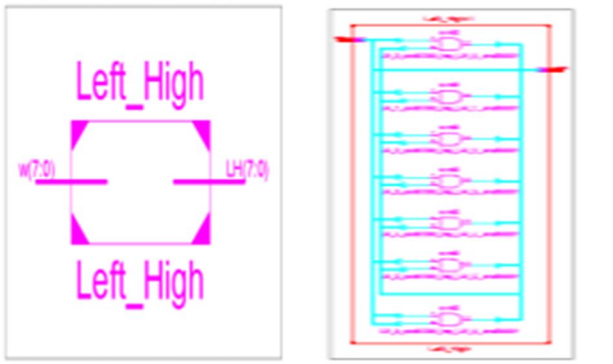
D. HD Encoder Block

If | Δ | ≥ Dm, the HD encoder block, shown in Fig. 5(d), generates a multi-TC code as described in Section IV-A3. The HD encoder block modifies only lower two bits to ensure that t has two or more TCs. The TC counter block is used to determine how the lower two bits should be modified. Since 2-TC words are interpreted by the decoder as an absolute value, the error is simply the difference between s and t. On the decoder side, there is no HD decoder for multi-TC codewords since they are absolute values to be used as s as they are.

V. SIMULATION RESULTS

The outputs of the left high in transition count block with encoder and decoder and high difference encoder block in the encoder are represented according to the varying inputs. The logic for detecting 1 transition count words are left high, left low, right high, right low. The output given below represents the block diagram (Fig. 3(a)), circuit diagram (Fig. 3(b)) and the output (Fig. 4) if left high is used in the transition count block. The following output represents the block diagram (Fig. 5(a)), circuit diagram (Fig. 5(b)) and the output (Fig. 6) of the transition count block when both encoding and decoding is done. The given output represents the block diagram (Fig. 7(a)), circuit diagram (Fig. 7(b)) and the output (Fig. 8) of the high encoder block that includes transition count block along with multiplexers. The HD encoder block modifies only lower two bits to ensure that there are two or more TCs. The Transition Counter block is used to determine how the lower two bits should be modified.

The proposed Shift-Invert coding & AXSERBUS Encoding technique was implemented in Verilog code. Fig. 9 shows the output of the image data passed through high difference encoder. Fig. 10 is the output of the shift-invert coding block and Fig. 11 is the output of the image data when passed through the shift-invert coding block. Image data is nothing but when the image is being converted into binary data and stored. Then it is saved using a memory file which is being read later.



(a) (b)

Fig. 3. Left High block (LH Block) (a) RTL Diagram of Left High block (b) Schematic Diagram of Left High block

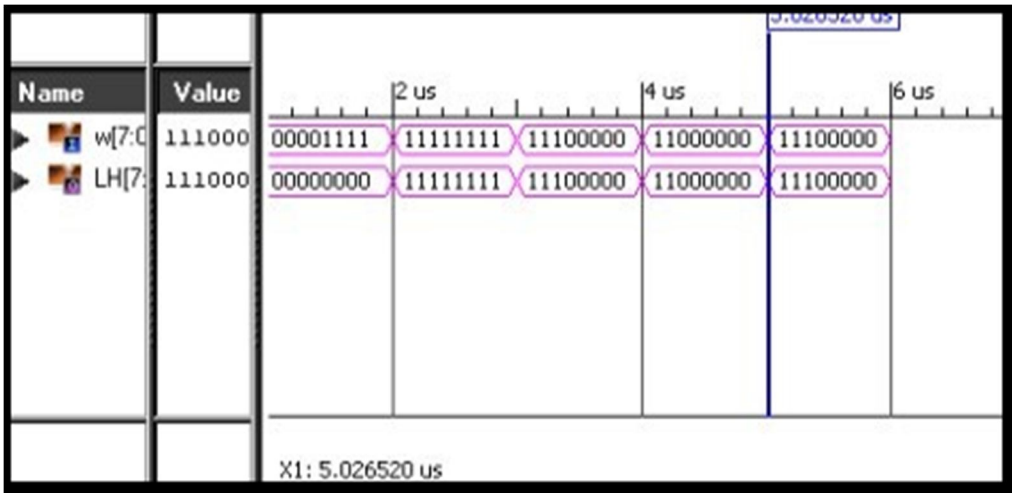
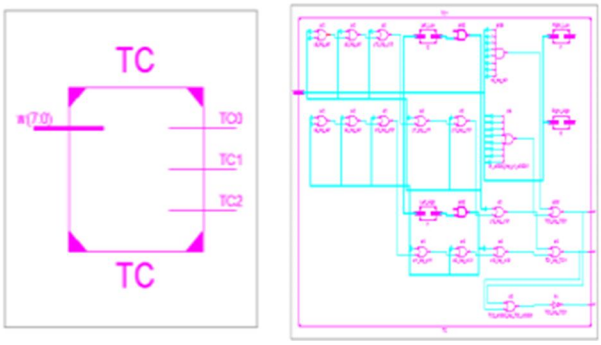


Fig. 4. Output of Left High block (LH Block) in Fig. 2(b)



(a) (b)

Fig. 5. Transition Count block (TC Block) (a) RTL Diagram of TC block (b) Schematic Diagram of TC block

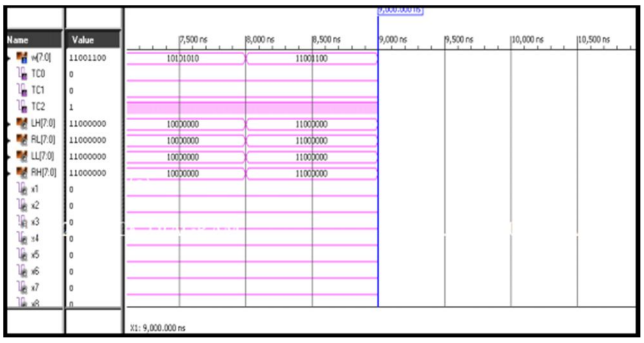
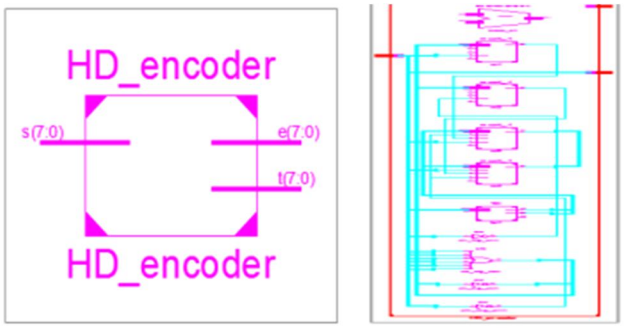


Fig. 6. Output of Transition Count Block (TC Block) in Fig. 2(b)



(a) (b)

Fig. 7. High Difference Encoder Block (HD Block) (a) RTL Diagram of HD Encoder block (b) Schematic Diagram of HD Encoder block

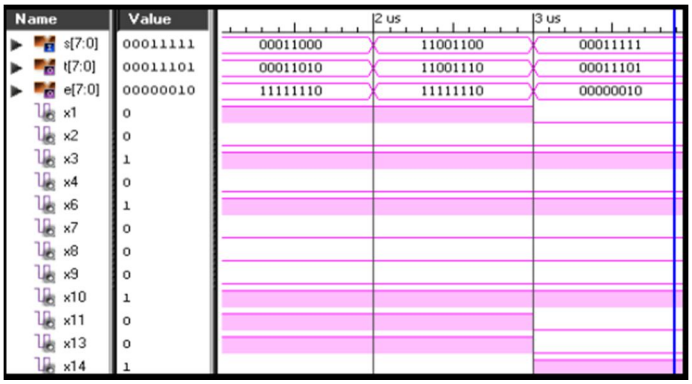


Fig. 8. Output of HD Encoder Block (HD Block) in Fig. 2(d)

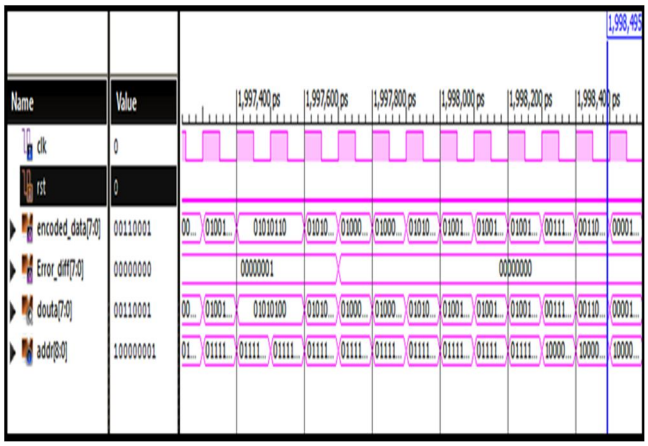


Fig. 9. Image Data of High Difference (HD Block) Block

Fig. 10. Output of Shift-Invert Coding Block

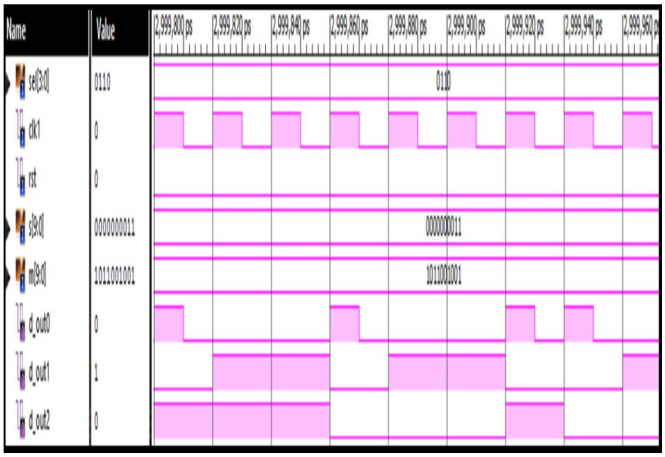


Fig. 11. Image Data of Shift-Invert Coding Block

Table 1 summarizes the results for buses with average number of ransitions per cycle whose bit widths were considered. For individual bit-width pattern, 10000 simulation cycles were conducted. It shows that in all cases AXSERBUS encoding reduces the average number of bus transitions per cycle considerably. Further when compared with the Bus Invert and Shift-Invert coding, AXSERBUS is clearly the winner and also the number of lines are irrespective of bus-width.

TABLE 1 Simulation results for buses with arbitrary width

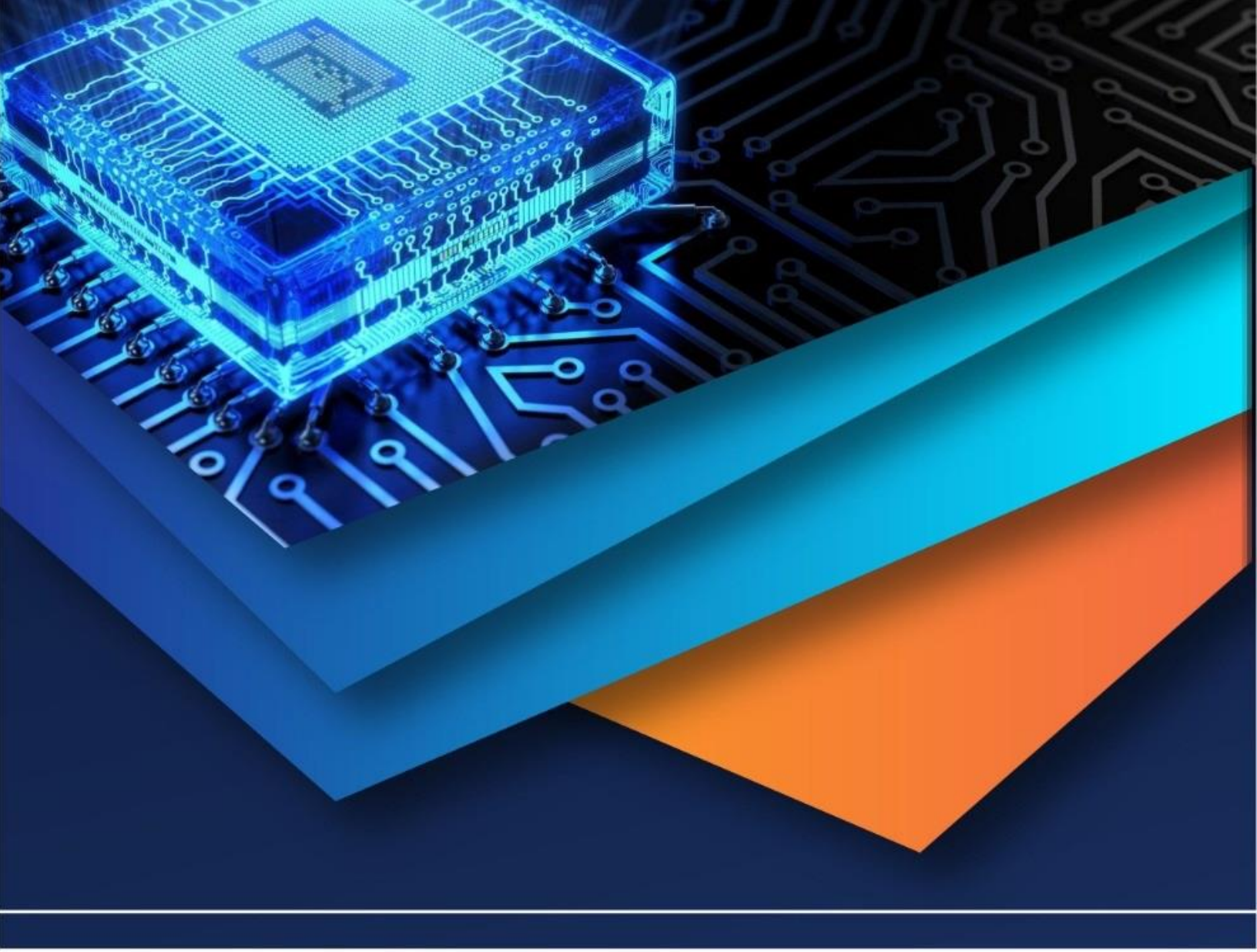
Bit-width	Average Number of transitions per cycle (Bus Invert Coding) BIC	Average Number of transitions per cycle (Shift-Invert Coding) SINV	Average Number of Transitions per cycle (AXSERBUS Coding) AXSERBUS
9	4.50	3.61	3.24
13	6.50	5.31	5.16
21	10.50	8.83	7.92
29	14.50	12.42	12.23
35	17.50	15.12	14.15
43	21.50	18.80	16.90

VI.CONCLUSION

A new method lnown as approximate serial bus encoding technique called AXSERBUS was implemented to increase energy efficiency by reducing power consumption. The power consumption is being tackled here by reducing the signal transitions for efficient data transfer during transmission of data from a sensor to a microcontroller through off chip buses. AXSERBUS technique is simple yet efficient scheme that enhances other well-known Bus Invert and Shift-Invert encoding techniques. AXSERBUS achieved a significant reduction in transition count while maintained a high application-level quality both for natural images and text images when compared to other two techniques. The reduction in Transition Count was evaluated by encoding an image using Optical Character Recognition application. On completely random data, the simulation results suggest that the AXSERBUS reduces the average number of transitions over and above the Shift-Invert bus encoding technique. Further, the AXSERBUS technique does not use extra lines as used in Shift-Invert technique. From the experimental data in Table 1, we can see that the transition counts are less for smaller bus-widths and more for larger bus-widths and also it is clear that AXSERBUS has the least signal transitions by maintaining highest accuracy and quality of images.

REFERENCES

[1] Urs Niesen, Shrinivas Kudekar, “Joint Crosstalk-Avoidance and Error-Correction Coding for Parallel Data Buses”, IEEETransaction on Information Theory, vol. 65, no. 3, Mar. 2019.
[2] Shady Soliman, Mohammed, A.Jaelaa, Abdelrhman, M.Abotalebd, “FPGA implementation of dynamically reconfigurable IoT security module using algorithm hopping”, in Proc. Design FPGA, 2019, pp. 108-121.
[3] Zhen Wang, Tao Wang, Jianhui Jiang,” Failure probability analysis and critical node determination for approximate circuits”, in Proc. VLSI,2019, pp.421-426.
[4] Junjun H, Zhijing Li, Meng Yang, “A high-accuracy approximate adder with correct sign calculation”, in Proc. IEEE Int. Conf.,2019, pp. 634-637.
[5] Younghyun Kim, Setareh Behroozi, Vijay Raghunathan, Anand Raghunathan, “AXSERBUS:A Quality-Configurable Approximate Serial bus for Energy-Efficient Sensory Data Transfer,” IEEE Journal on Emerging and selected topics in Circuits and Systems, vol. 8, no. 3, Sep. 2018.
[6] Kumud Kumar Bhardwaj, Vandana Khare, “Design of Shift-Invert Coding Using Barrel Shifter”, International Journal of Pure and Applied Mathematics, vol. 118, no. 14, 2018, pp. 247-252.
[7] Vincent Camus, Mattia Cacciotti, Christian Enz, “Design of Approximate Circuits by Fabrication of False Timing Paths: The Carry Cut-Back Adder”, in Proc. IEEE/ACM Int. Symp. Low Power Electron.Design(ISLPED), Jul. 2017,pp. 1-6
[8] Daniele Jahier Pagliari, Enrico Macii, Massimo Poncino, “Zero-Transition Serial Encoding for Image Sensors”,IEEE sensors journal, vol. 17, no. 8, Apr. 15, 2017.
[9] Michael Gautschi, Pasquale Davide Schiavone, Andreas Traber, “Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices”, IEEE Transactions on Very Large Scale Integration (VLSI) systems, vol. 25, no. 10, Oct. 2017.
[10] Alexander E. Shapiro, Francois Atallah, Jihoon Jeong, Jeff Fischer, “Adaptive power gating of 32-bit Kogge Stone adder”, in Proc. IEEE VLSI, Jul 2016, pp.124-129.
[11] Kedar Karmarkar, Spyros Tragoudas, “Error Correction Encoding for Tightly Coupled On-Chip Buses”,IEEE transactions on very large scale integration (VLSI) systems, vol. 22, no. 12, Dec. 2014.
[12] Samaneh Babayan-Mashhadi, Reza Lotfi, “Analysis and Design of a Low-Voltage Low-Power Double-Tail Comparator”, IEEE Transactions on very large scale integration (vlsi) systems, Vol. 22, no. 2, February 2014.
[13] Shahzad Asif, Mark Vesterbacka, “Performance analysis of radix-4 adders”, IEEE Trans. Circuits Syst., vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[14] Massimo Alioto, Gaetano Palumbo, Massimo Poli, “Optimized design of parallel-carry select adders”, INTEGRATION the VLSI Journal 44 (2011).
[15] Jayapreetha Natesan, Damu Radhakrishnan, “Shift- Invert Coding (SINV) for Low Power VLSI”, Proceedings of the EUROMICRO Systems on Digital System Design, vol. 11, no. 4, pp. 24-30, 2004.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY