Hindawi
VLSI Design
Volume 2017, Article ID 4914301, 7 pages
https://doi.org/10.1155/2017/4914301

Research Article

Comparative Power Analysis of an Adaptive Bus Encoding Method on the MBUS Structure

X. Yang,[1] N. Wu,[2] and J. H. Andrian[3]

[1]*Engineering Department, College of Science and Engineering, University of Houston-Clear Lake, Houston, TX, USA*
[2]*Electrical Engineering Department, Arkansas Tech University, Russellville, AR, USA*
[3]*Electrical and Computer Engineering Department, Florida International University, Miami, FL, USA*

Correspondence should be addressed to J. H. Andrian; andrianj@fiu.edu

Received 30 May 2017; Revised 16 August 2017; Accepted 25 September 2017; Published 23 October 2017

Academic Editor: Xueqing Li

This paper proposes a novel bus encoding method on MBUS in order to reduce the power consumption of system-on-chips (SoCs). The main contribution is to lower the bus activity by an average 64.55% and thus decrease the IO power consumption through reconfiguring the MBUS transmission. This method is effective because field-programmable gate array (FPGA) IOs are most likely to have very large capacitance associated with them and consequently dissipate a lot of dynamic power. Experimental result shows an average 70.96% total power reduction compared with the original MBUS implementation.

1. Introduction

For the energy-limited IoT and wearable devices, low-power technologies to prolong the life of the battery become an important constraint for embedded chip design [1, 2]. A general formula of dynamic power dissipation can be written as

$$P\_\text{dyn} = \alpha \times C \times \left(V^2\right) \times f, \tag{1}$$

where the supply voltage denoted as V and the load capacitance denoted as C are decided by the tape-out technology [3]. The maximum clock or operational frequency, denoted as f, is the reciprocal of the sum of setup time ($t_$setup), critical combinational path delay between flip flops ($t_$propagation), clock to q delay ($t\_\text{clk} - q$), and clock skew ($t_$skew). All these four parameters are constant and cannot be changed after synthesis and technology selection. Hence, as the aforementioned equation (1) referred to, one of the most effective ways to reduce the on-chip power dissipation is reducing the toggle activity, denoted as α, of signals, IOs, and logics.

Moreover, the toggling rate of the on-chip bus becomes one of the main design issues because it dominates the power consumption and degrades the performance due to a complex scalability [4, 5]. For example, the existing on-chip buses, such as AHB [6] and AXI [7] from ARM Holdings, Wishbone from Silicore Corporation [8], and OCP from OCP-IP [9], cost much hardware resource in terms of slice/gate count and energy consumption, due to a large number of IO and signal definitions and complicated structures. They are designed for a broad range of various applications and are characterized by high flexibility, scalability, and universality. Under this context, a cost-effective and power-efficient control bus named as master bus (MBUS) [10] has been proposed for specific IoT applications as our previous work, making a better balance between the limited energy on tiny-size embedded chips and high speed requirement of complex computations. Furthermore, it has been improved in [11] to preselect data sequence for Advanced Encryption Standard (AES) engines, so that the state buffering and rescheduling overhead can be reduced.

Based on the MBUS protocol, this paper presents a novel bus encoding technology on the existing address signal in order to reduce the dynamic power. More specifically, the contributions are as follows:

(1) We represent four different transfer types with two always-zero bits of the address signal, by which the switching activity can be decreased and thus the IO power dissipation can be lowered.

(2) As a case study, we apply this method on a generic control/central bus, MBUS, and introduce the basic idea of the hardware structures. Field-programmable gate array (FPGA) results show an average 70.96% power reduction compared with the original designs.

The rest of the article is structured as follows: in Section 2 we present related work and Section 3 discusses our proposed encoding methodology. In Section 4 we introduce the hardware implementation. Then, we demonstrate and evaluate the approach with simulation, synthesis, and power measurement in Section 5. Last, we summarize our work in Section 6.

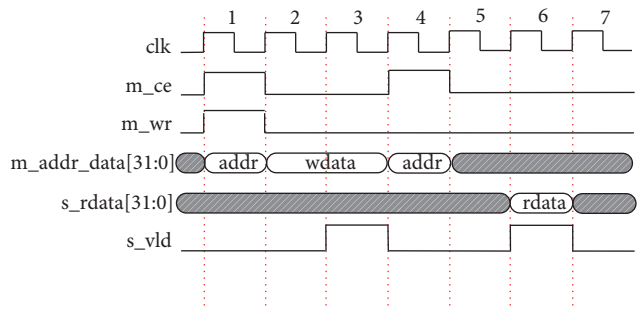


Figure 1: Original MBUS timing diagram.

2. Related Work

In earlier works, there were many bus encoding algorithms aiming to reduce the power dissipation on interfaces by mapping the information on IOs or signals to a form which has less transition activity than the original, such as the bus-invert encoding [12, 13], gray code [14], serial T0, and combined bus-invert and T0 technology [15].

The basic idea of bus-invert encoding is flipping a transmission word when the Hamming distance between the current word and the previous word is greater than a half of the bus size; otherwise, no encoding is applied to the transmission word [12]. In such a way the maximum number of lines that switches can be limited to 50% of the bus size. Gray and T0 encodings are targeted to the situations in which the address to be sent is consecutive to the one sent previously [14]. Furthermore, the bus-invert technique is combined with T0 in [15], thus obtaining more activity reduction compared with single T0 technology.

However, all of these encoding algorithms require redundant control lines and overhead logic to recognise the switching rate between subsequent transmissions. And they are less effective when applied to a single or interleaving transfer mode, because in this case the percentage of in-sequence bursts decreases. To overcome these issues, this paper proposes a new encoding method with the existing address lines. By differentiating and processing different register configuration modes, an average toggle rate will be decreased by using this encoding technology. More important, our proposed encoder employs only concatenation and shifting operators of hardware description language (HDL), which will be converted as just reconnected signals and IOs after synthesis.

3. Proposed Encoding MBUS

In this section, we present an encoding MBUS protocol, capable of lowering the bus activity and thus decreasing the IO energy consumption.

3.1. An Analysis on MBUS Protocol. MBUS is defined as a control bus for functional register configuration [10]. Considering the instruction operations on energy-limited chips, MBUS is created for minimal power consumption and reduced interface complexity so that it only supports single transfer mode with at least one-cycle command and one-cycle data.

As shown in Figure 1, "m_ce" and "m_wr," respectively, represent the data transfer enabled signal and direction, 1 for write and 0 for read. "m_data" is designed as a bidirectional and shared interface with write and read channels so that the wire usage efficiency can be increased and the hardware interconnection can be simplified. Since the MBUS is a single-master and multislave structure, there is no arbitration so that the command stage takes only one master cycle, in which "m_ce," "m_wr," and "m_addr" are sent from master to salve.

From slave to master, a valid signal, denoted as "s_vld," is defined to handshake with the request in order to avoid metastable signals crossing different time domains and overflows of command FIFOs. Additionally a response delay timer is required to detect command errors. If the current response is a timeout, the command is indicated as "error" and must be "retried" or "discarded" by the master.

By analyzing the MBUS protocol, it can be observed that the sequence of command stages can be predicted, exploited, and rearranged using a scheduler or an arbiter [16]. Moreover, the toggle rate reduction on "m_addr" of the command stage can dramatically decrease the entire MBUS IO power due to its multibit definition.

3.2. Functional Register Map. As an example, a digital system built around an 8-bit microprocessor providing 16-bit address lines is applied in our work, which can address up to 64 KB of memory. The hardware of the system is arranged so that devices on the address bus will only respond to particular addresses which are intended for them, while all other addresses are ignored. This is the job of the address decoding circuitry, and that establishes the memory map of the system. For instance, system's register map may look like Figure 2. This memory map contains 16 bases, each for one application-specific module or peripheral, which is also quite common in actual system architectures.

As a case study, the first 4 KB of address space, denoted as Base#0 in Figure 2, may be allotted to random access memory (RAM), another 4 KB of Base#1 to read only memory (ROM), and the remainder to a variety of other devices such as host or device interfaces, timers, counters, and wireless devices. In one of the bases, Base#F used by a host interface, for example, the offset or pointer "0xF18," represents the transfer length, "0x000" indicates the transfer direction, 1 for write

Input: PreBase = sustain the previous value of the base address;
Input: PreOffset = sustain the previous value of the offset address;
Input: CurBase = update the current value as the base address;
Input: CurOffset = update the current value as the offset address;
Output: EncAddr = encoding address;
index = FrameNum;
repeat
  if *Mode#0: Consecutive transactions in the Same Base;* **then**
    EncAddr = {PreBase, PreOffset, 1'b1, 1'b1}; break;
  end
  if *Mode#1: Consecutive transactions in different bases;* **then**
    EncAddr = {CurBase, PreOffset, 1'b0, 1'b1}; break;
  end
  if *Mode#2: Interleaving transactions in the same base;* **then**
    EncAddr = {PreBase, CurOffset, 1'b1, 1'b0}; break;
  end
  if *Mode#3: Interleaving transactions in different bases;* **then**
    EncAddr = {CurBase, CurOffset, 1'b0, 1'b0}; break;
  end
  index = *index* − 1;
until index = 0;

Algorithm 1: Algorithm for the encoding MBUS address.

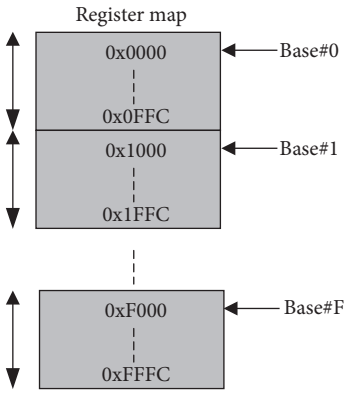


Figure 2: Functional register map.

m_addr[15:12]	m_addr[11:2]	m_addr[1]	m_addr[0]
Base Address	Offset	Same Base flag	Consecutive Address flag

Figure 3: Encoding MBUS address field.

and 0 for read, and "0xF0C" performs the transfer enable. Hence CPU can send 512 B data out using the host interface by sequentially writing "0x200" to the memory location "0xFF18," "0x1" to the memory location "0xF000," and "0x1" to the memory location "0xFF0C." Then the address switching rages are 37.5% for both of the two subsequent operations.

An alternative way to drive the bus is in the order of "0xF000," "0xFF18," and "0xFF08." In such a way the toggle rate of the first transmission will not be changed but the second sequence switching rate will be decreased to 6.25%. Furthermore, since the consecutive or increment register configuration, such as "0xF000," "0xF004," "0xF008," and "0xF00C" in sequence, is very commonly used by software configuration, an encoding bus can be employed to reduce the toggle rate instead of directly sending the original addresses.

3.3. The Address Encoding Method. In our study, the data bus is word size so that the least significant 2 bits of MBUS address is always 0. Instead of filling out 2-bit 0, we redefine this 2-bit field as 2 flags, "Same Base" and "Consecutive Address."

As shown in Figure 3, the "Same Base" field represents that the previous and current addresses on bus are from the same base, 1 for valid and 0 for invalid. Likewise, the "Consecutive Address" field indicates that the two addresses are consecutive in the memory location. In other words, when the "Consecutive Address" flag is asserted the present address will be the previous address plus 4 as a word-size bus.

More specifically, Algorithm 1 introduces the encoding procedure and Figure 4 explains the encoding method with four different test cases.

(1) Mode#0: generally software configurations are consecutive and increment in the memory location. In such case the base and offset fields sustain the previous values and the 2-bit identifiers of the "Same Base" and "Consecutive Address" are asserted. As an example shown in Figure 4, since the higher 14 bits of the 16-bit current address is not changed in Mode#0, the IO switching power will be dramatically decreased. On the decoder side, one 10-bit adder (the least significant 2 bits is always 0 and the most significant 4 bits is directly connected) is required to compute the original address.

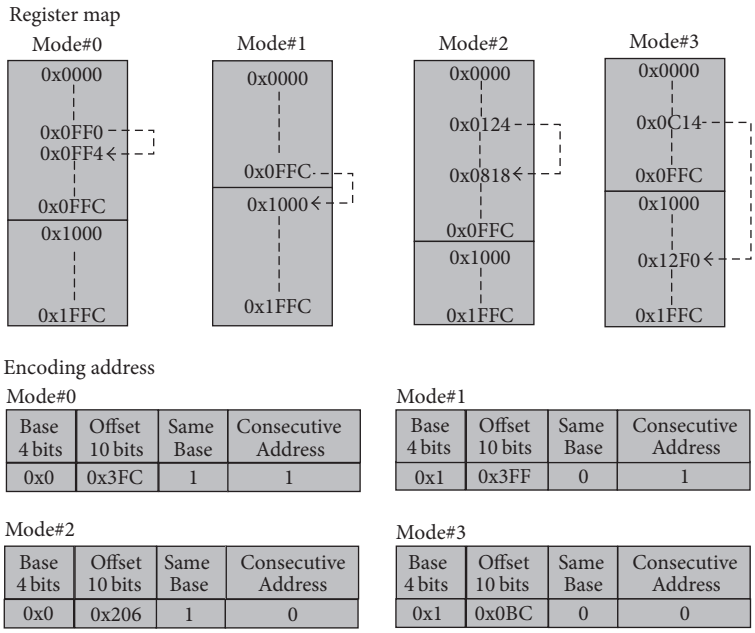


Figure 4: Examples of different register configuration modes.

(2) Mode#1: when the transfers are consecutive and base-crossing, the current "Based Address" field should be updated and the "Same Base" flag has to be deasserted. Because base-crossing results in a large percentage of IO flipping, the power dissipation can be reduced a lot by using this method. As an example shown in Figure 4, the "Offset Address" field sustains the previous address so that the flipping rate is reduced from 68.75% (0x0FFC – 0x1000) to 12.5% (0x0FFC – 0x1FFD). Different from Mode#0, on the salve side the original address can be simply decoded by concatenating 4-bit "Base Address" with the initial offset of the current base, in this case they are 12-bit 0.

(3) Mode#2: memory addresses are often interleaved in the same base. In this case the 2-bit identifiers are set to be binary "10," and the power consumption is similar to the traditional address transmission due to the similar toggle rate. As shown in Figure 4, in the third case the number of switching bits is changed from 6 to 7.

(4) Mode#3: in Mode#3, the switching rate is not modified in the case of base-crossing and address-interleaving. For instance, the least significant 2 bits, "Same Base" and "Consecutive Address," is 0 in Figure 4 and the fields "Based Address" and "Offset Address" have to be updated, so that the dynamic IO power will be the same as that of the conventional bus communication.

4. Implementation of the Encoding MBUS Structure

Address encoding concepts are introduced in Section 3; we then implement the design under test (DUT) and evaluate the performance using a performance evaluation methodology [17].

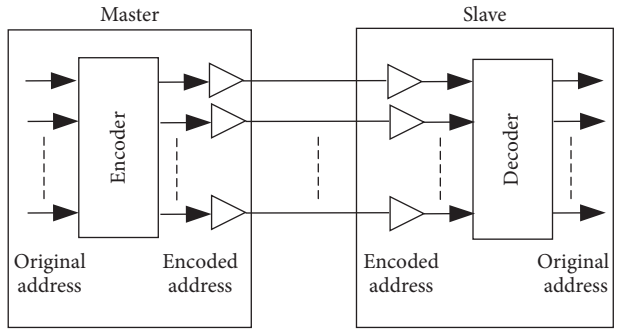
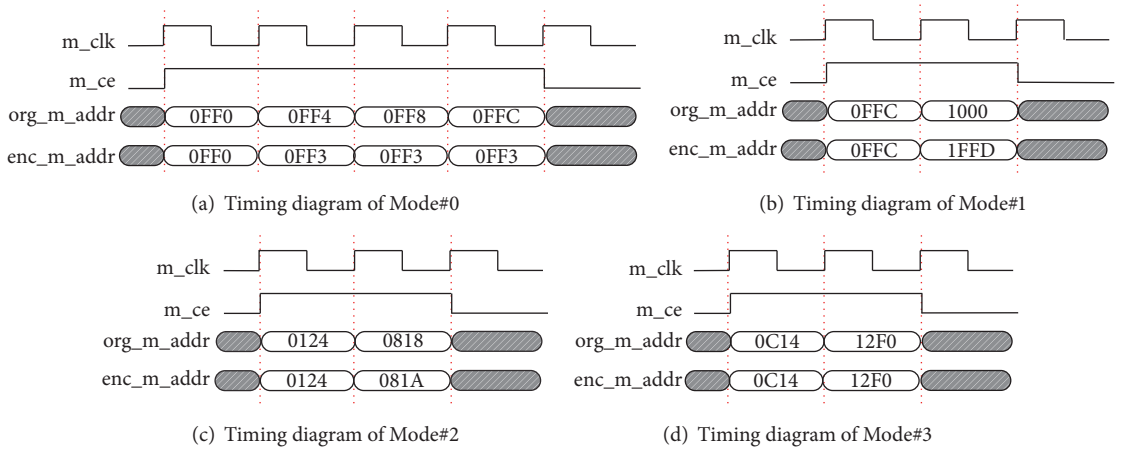


Figure 5: Encoding MBUS structure.

As a case study, we consider that MBUS address will be of 16-bit wide for both original and encoding tests. The DUT structure is shown in Figure 5. First we design an MBUS master using Verilog HDL. The basic module of the master is a bus encoder. Then, we implement a decoder on the slave side as a verification intellectual property (VIP).

In our work, we use Mentor Graphic ModelSim 10.4d as the simulator. After simulation we can obtain the waveforms (VCD) and after the synthesis we can get the net lists (NCD). These files are required to analyze the power consumption using XPower Analyzer (XPA). Moreover, Xilinx ISE 14.7 is employed as the synthesis tool with the target device Spartan6-XC6SLX4L in our study.

We perform 10 us simulation to understand the tests. For example, Figure 6(a) shows a vector of 4-beat increment transfer. The addresses on the encoding MBUS are "0x0FF0," "0x0FF3," "0x0FF3," and "0x0FF3." Notice that the last three transactions are not modified so that the switching activity,



(a) Timing diagram of Mode#0

(b) Timing diagram of Mode#1

(c) Timing diagram of Mode#2

(d) Timing diagram of Mode#3

Figure 6: Timing diagrams of four different modes.

calculated as $2/(16 \times 4)$ in this case, is lower than the toggle rate of the original MBUS. Since the original MBUS addresses should be modified in each transaction, in the order of "0x0FF0," "0x0FF4," "0x0FF8," and "0x0FFC," the toggle rate should be computed as $3/(16 \times 4)$. Although the improvement is limited for this 4-beat transfer, the IO power dissipation will be reduced much more than this case when the number of transactions is large.

In Figure 6(b), the power saving is more than the first case, because all the 10-bit offset field sustains the previous value. The switching rate is only 2/16 for the encoding MBUS; however, the original bus costs 11/16 bus toggle rate. For Mode#2 and Mode#3, the power consumption of the encoding bus is similar to the original MBUS; hence, the power saving is very low.

In what follows, the power consumption reports and corresponding signal rate information are generated by the XPA tool. We claim and experimentally show that there is almost a linear relationship in signal rate and power reductions for bus transfers. As an example shown in Table 1, the IO power dissipation is 16.04 mW with 12.2% signal rate on the original MBUS (as shown in the third row), and the IO power cost is reduced to 2.53 mW due to the very low signal rate (1.9%) on the encoding MBUS (as shown in the fourth row).

Furthermore, we summarize the total power dissipation, including both static and dynamic power, in Table 2. As an example shown in the third and fourth rows, in Mode#0 the power consumed by our proposed encoding method is reduced by $(15\,\text{mW}/29\,\text{mW}) \times 100 = 51.7\%$ compared with the original bus.

In the same way we can estimate power dissipation of the other test modes. Generally, in Mode#1 (as shown in the fifth and sixth rows) up to 70.9% of power is saved but in Mode#2 (as shown in the seventh and eighth rows) and Mode#3 (as shown in the ninth and tenth rows) the power dissipations are very similar to the original MBUS, which proves our expectation and analysis in Section 3.

Table 1: Signal rate and power dissipation.

Cases	Address signal		
	Signal rate	% high	Power (mW)
Mode#0			
Encoding MBUS	1.9	37.9	2.53
Original MBUS	12.2	61.9	16.04
Mode#1			
Encoding MBUS	13	68.1	17.14
Original MBUS	68.1	34	89.72
Mode#2			
Encoding MBUS	43.3	24.8	57.02
Original MBUS	37.1	18.6	48.94
Mode#3			
Encoding MBUS	49.6	30.9	65.33
Original MBUS	49.6	30.9	65.33

5. Results and Analysis

In this section, the system performance of speed and power consumption is analyzed and concluded.

In general, it can be observed that our proposed work significantly reduces the total power consumption in Mode#0 and Mode#1 by lowering the IO power, actually by decreasing the switching activity. The toggle rate and IO power decreasing can reach up to 18.9% in the best case of Mode#1, and thus in the same test the total power reduction is achieved by 29.1%.

The worst case occurs at Mode#2, where the power consumption of our design is increased to 1.1 times compared with the original MBUS. Assuming the percentage of each test mode is 25%, the proposed method can reduce the system power to an average 70.96% compared with the original bus. Since the consecutive (Mode#0) and bank/page/base-crossing (Mode#2) cases frequently occur at the register configuration, our proposed work is suitable for the control/central buses such as MBUS.

Table 2: Power dissipation of original and encoding MBUS.

Cases	Power consumption (mW)		
	Static power (mW)	Dynamic power (mW)	Total power (mW)
Mode#0			
Encoding MBUS	11	4	15
Original MBUS	11	18	29
Mode#1			
Encoding MBUS	11	19	30
Original MBUS	12	91	103
Mode#2			
Encoding MBUS	11	59	70
Original MBUS	11	51	62
Mode#3			
Encoding MBUS	11	67	78
Original MBUS	11	67	78

Table 3: Maximum operational frequency.

Delay (ns)		Maximum operational frequency (MHz)
Minimum period	3.094	323.217
Minimum input arrival time before clock	4.526	
Maximum output required time after clock	6.540	
Maximum combination path delay	7.646	

Moreover, the system speed is estimated in Table 3. The maximum operational frequency can reach 323.217 MHz with the critical path as 3.094 ns, which is shown in the second row. As shown in the other rows, the DUT meets all the timing constrains such as setup and hold time.

6. Conclusions

This paper proposes a novel address encoding method and applies it to the MBUS protocol as a case study. Power analysis results show an average 70.96% power reduction with equal weight of different test modes and up to 70.9% power decreasing in the specific case of base-crossing and increment addresses.

So far the encoder is considered as the design under test, and in the future we will focus on the performance evaluation for the whole system involving both masters and slaves. Compared to the master design with just concatenation operators, the decoder implementation will cost more slices and power due to the complex address decoding structure, involving one 10-bit adder, one 4-channel multiplexer, 14 latches, and some flip flops for a state machine.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] N. Shafiee, S. Tewari, B. Calhoun, and A. Shrivastava, "Infrastructure Circuits for Lifetime Improvement of Ultra-Low Power IoT Devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2598–2610, 2017.

[2] S. Lin, P. Chen, and Y. Lin, "Hardware Design of Low-Power High-Throughput Sorting Unit," *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1383–1395, 2017.

[3] S.-J. Ruan, S.-F. Tsai, and Y.-T. Pai, "Design and analysis of low power dynamic bus based on RLC simulation," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: Emerging VLSI Technologies and Architectures, ISVLSI'07*, pp. 113–118, bra, March 2007.

[4] J. Nunez-Yanez and G. Lore, "Enabling accurate modeling of power and energy consumption in an ARM-based System-on-Chip," *Microprocessors and Microsystems*, vol. 37, no. 3, pp. 319–332, 2013.

[5] Y.-G. Chen, W.-Y. Wen, Y.-T. Wang, Y.-L. Lee, and S.-C. Chang, "A novel low-cost dynamic logic reconfigurable structure strategy for low power optimization," in *Proceedings of the 21st Asia and South Pacific Design Automation Conference, ASP-DAC 2016*, pp. 250–255, mac, January 2016.

[6] *AMBA AHB specification*, Axis, Sunnyvale, CA, USA, 1999.

[7] *AMBA AXI protocol specification*, Axis, Sunnyvale, CA, USA, 2003.

[8] *Wishbone bUS*, Silicore Corp., Corcoran, MN, USA, 2003.

[9] *Open Core Protocol Specification*, OCP Int. Partnership, Beaverton, OR, USA, 2001.

[10] X. Yang and J. H. Andrian, "A High-Performance On-Chip Bus (MSBUS) Design and Verification," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 7, pp. 1350–1354, 2015.

[11] X. Yang and W. Wen, "Design of a pre-scheduled data bus for advanced encryption standard encrypted system-on-chips," in *Proceedings of the 22nd Asia and South Pacific Design Automation Conference, ASP-DAC 2017*, pp. 506–511, jpn, January 2017.

[12] K. Patil, S. Madabal, and A. Motagi, "Signal transforms with bus invert encoding for low power applications," in *Proceedings*

of the 10th International Conference on Intelligent Systems and Control, ISCO 2016, ind, January 2016.

[13] S.-F. Tsai and S.-J. Ruan, "DS2IS: Dictionary-based Segmented Signal Inversion Scheme for low power dynamic bus design," in *Proceedings of the 9th International Conference on Information Technology, ICIT 2006*, pp. 293–296, ind, December 2006.

[14] H. Chung, S. Hong, B. K. Lim, L. Ma, and B. K. Yi, "Performance analysis and multidimensional gray coding scheme for a real-number M-Ary QAM," in *Proceedings of the 2016 International Conference on Information and Communication Technology Convergence, ICTC 2016*, pp. 954–959, kor, October 2016.

[15] D. J. Pagliari, E. Macii, and M. Poncino, "Serial T0: Approximate bus encoding for energy-efficient transmission of sensor signals," in *Proceedings of the 53rd Annual ACM IEEE Design Automation Conference, DAC 2016*, usa, June 2016.

[16] M. Fan, Q. Han, and X. Yang, "Energy minimization for on-line real-time scheduling with reliability awareness," *The Journal of Systems and Software*, vol. 127, pp. 168–176, 2017.

[17] X. Yang, N. Wu, and J. H. Andrian, "A novel bus transfer mode (AS transfer) and a performance evaluation methodology," *Integration, the VLSI Journal*, vol. 52, pp. 23–33, 2016.