Narrow Bus Encoding for Low-Power DSP Systems

Youngsoo Shin, Kiyoung Choi, and Young-Hoon Chang

***Abstract*—High levels of integration in integrated circuits often lead to the problem of running out of pins. Narrow data buses can be used to alleviate this problem provided that the degraded performance due to wait cycles can be tolerated. We address bus coding methods for low-power core-based systems incorporating narrow buses. We show that transition signaling combined with bus-invert coding, which we call BITS coding, is particularly suitable for the data patterns of typical DSP applications on narrow data buses. The application of BITS coding to real circuit design is limited by the extra bus line introduced, which changes the pinout of the chip. We propose a new coding method, which does not require the extra bus line but retains the advantage of BITS.**

***Index Terms*—CMOS, low-power, switching-activity, system-level.**

I. Introduction

RECENTLY, core-based design of digital systems has become prevalent, in order to cope with ever decreasing time to market. A system designed with cores often contains a lot of components, such as core processors, digital signal processors (DSPs), peripheral interface circuits, and application specific integrated circuits (ASICs). The number of pins, which directly contributes to the size and cost of a chip, is one of the problems with such a high level of integration because pin-count naturally increases with the number of components integrated into the chip. Communication with off-chip devices by means of narrow data buses is one way to reduce the number of pins, at the cost of reduced performance due to an increased number of wait states. Narrow buses are also used to build low-cost high-volume embedded systems. For example, a system requiring a 16-bit wide external memory can use an 8-bit wide narrow bus if the increased delay for memory transfer can be tolerated. The cost of the system is reduced because a cost-effective 8-bit wide memory and bus can be used and fewer pins are required.

Power consumption is another problem with high integration, especially in portable systems such as cellular phones and personal digital assistants (PDAs). It is a well-known fact that a lot of power is consumed driving the off-chip bus, due to the large off-chip driver, the pad capacitance, and significant off-chip capacitance [1]. Power consumed by off-chip driving becomes dominant as devices are scaled down, because off-chip capacitance does not depend on process technology, but depends on the package and printed circuit board (PCB) technologies.

Bus encoding reduces the power consumption of a bus by encoding the information transferred on the bus in such a way that the encoded version has fewer transitions than the original. There are various low-power coding methods for data buses: bus invert (BI) code [2], [3] for uncorrelated data patterns, transition signaling [4] for data patterns where the probability of `0` and `1` is very different and probability-based mapping [5], [6], followed by transition signaling for patterns with nonuniform probability densities. In the case of instruction address patterns, gray code [7], `T0` code [8], and inc-xor [6] are efficient. For special-purpose applications, where the information about the sequence of patterns is available *a priori*, the characteristics of patterns can be exploited to reduce bus transitions. The beach solution [9] and the partial bus invert (PBI) code [10] perform well in this case.

In this paper, we study bus coding schemes for low-power core-based systems incorporating narrow buses. [1] Although various low-power bus coding techniques have been proposed, none of them has explicitly addressed the problem of coding patterns on a narrow bus. In contrast to a bus of full width, the narrow bus exhibits one very different property: correlations between consecutive patterns, if they exist, are entirely lost. We show that transition signaling combined with bus-invert (BI) coding [3], which we call *BITS coding* in this paper, is particularly suitable for this situation.

The extra bus line used in BITS coding makes the coding difficult to use in real circuit design because it implies changes to the interface specification of the chip. Furthermore, power reduction is obtained at the cost of the overhead of encoding and decoding circuits, which introduce delay, take up area and, indeed, themselves require power. To overcome the need for the extra bus line and the overhead of encoding and decoding circuits, while retaining the advantage of BITS coding, we propose a new coding method called *half-identity half-reverse and transition signaling* (`hihrTS`) *coding*. In this technique, coding logic is greatly simplified and the overhead of power, delay, and area accompanying the encoder and decoder circuits is much reduced. More importantly, `hihrTS` coding does not use any spatial and temporal redundancy and thus provides a coding method that is efficient in broad class of circuit designs. Furthermore, when a narrow data bus is used between a processor core and an off-the-shelf external memory, data can be stored in the memory in `hihrTs`-encoded form and then used by the processor after loading and decoding, which is difficult with BITS coding.

[1] We define a *narrow bus* as a bus transferring patterns whose width is larger than that of the bus, thus transferring each pattern in multiple cycles.

Manuscript received August 31, 1999; revised February 21, 2001. This work was supported in part by the NONDIRECTED RESEARCH FUND, Korea Research Foundation, 1999.

Y. Shin is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA.

K. Choi is with the School of Electrical Engineering and Computer Science, Seoul National University, Seoul 151-742, Korea.

Y.-H. Chang is with the Samsung Advanced Institute of Technology, Yongin-shi, Kyungki-do 449-900, Korea.

Publisher Item Identifier S 1063-8210(01)03839-2.

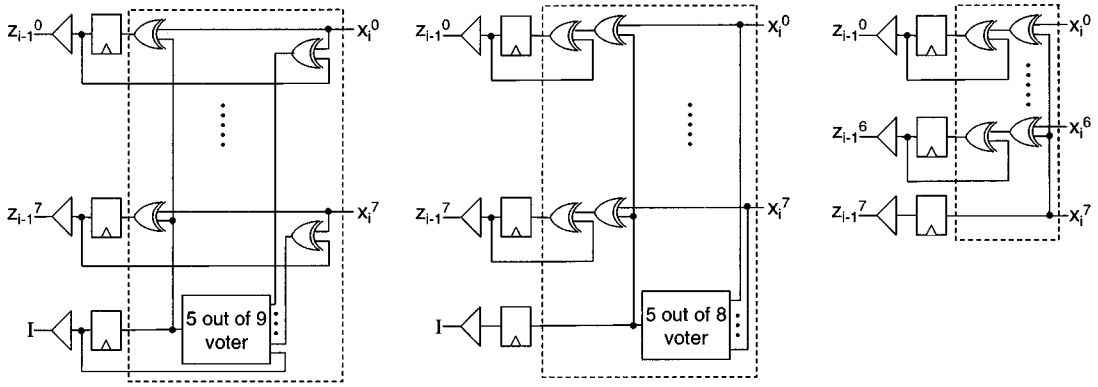

Fig. 1. Schematic diagrams of encoders (inside boxes) for (a) BI; (b) BITS; and (c) `hihrTS`.

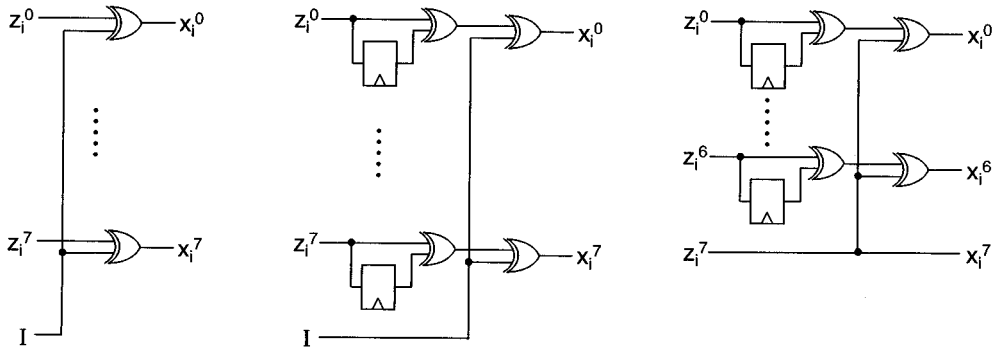

Fig. 2. Schematic diagrams of decoders for (a) BI; (b) BITS; and (c) `hihrTS`.

II. Coding for a Narrow Bus

A. BI and BITS Codings

If data patterns are randomly distributed in time and mutually independent in space, transferring them across a narrow bus (each pattern in multiple cycles) does not destroy the original randomness of the patterns. BI code (for which an encoder and decoder are shown in Figs. 1(a) and 2(a), respectively) still performs well in this case. In BI encoding, if the number of transitions between the current pattern on the bus, denoted by X_i, and the previous pattern, denoted by Z_{i-1}, exceeds half the bus width, the current pattern is transferred with each bit inverted. An extra bus line, denoted by I in Fig. 1(a), is used to signal the inversion.

When data patterns do not follow this ideal property, as is often the case in telecommunications, speech, and image processing applications, BITS code (for which an encoder and decoder are shown in Figs. 1(b) and 2(b), respectively) can be shown to perform better. In BITS encoding, if the number of `1`s in X_i is larger than half the bus width, then each bit of X_i is inverted (with line I set to `1`) and then transition-encoded. Otherwise, each bit of X_i is transition-encoded without alteration. [2]

As an example, consider the decimal representation in Fig. 3(a), which is drawn from samples of human speech [11]. Fig. 3(b) shows a typical 16-bit two's complement fixed-point representation with an 8-bit integral part and an 8-bit fractional part.[3] As shown in the figure, the low-order bits tend to be random whereas the high-order bits tend to have high *spatial* and *temporal* correlations. However, the temporal correlations of the high-order bits are lost when the original patterns are transferred on a narrow bus (8-bit wide bus in this example) since the high-order and low-order bits appear on the bus lines alternately, as illustrated in Fig. 3(c). Instead, the patterns of high-order bits are more spatially correlated, that is, the probability that the occurrence of `1` or `0` dominates each pattern increases compared to the original patterns.

5.683594	0000010110101111	X_0: 00000101	X_5: 01100000
10.578125	0000101010010100	X_1: 10101111	X_6: 00000001
-5.625000	1111101001100000	X_2: 00001010	X_7: 00000100
1.019531	0000000100000100	X_3: 10010100	X_8: 00000011
3.484375	0000001101111100	X_4: 11111010	X_9: 01111100
(a)	(b)	(c)	

Fig. 3. An example of patterns from human voice. (a) Decimal representation. (b) 16-bit two's complement fixed-point representation with 8-bit integral part. (c) The same patterns on 8-bit wide bus with 37 transitions.

If we apply BI encoding to the patterns in Fig. 3(c), the number of transitions is reduced from 37 to 30, as illustrated in Fig. 4(a), where bold face indicates the value of the signal on line I. Further reduction can be obtained if we apply BITS encoding as illustrated in Fig. 4(b). This is because it is highly probable that the occurrence of `1` or `0` dominates each pattern due to sign extension [see Fig. 3(c)], and BITS encodes `0` as a transition in the former case and `1` as a transition in the latter case.

B. `hihrTS` Coding

Although substantial power saving is possible with BITS coding, its application to real circuit design is severely limited

[2]In the coding framework proposed in [6], BITS encoding is equivalent to $f_1 = \text{inv}$ and $f_2 = \text{identity}$ without function F.

[3]In this example, 8-bit is the minimum width of the integral part required to avoid overflow.

00000101 **0**	00000101 **0**	00000101
10101111 **0**	01010101 **1**	11010101
00001010 **0**	01011111 **0**	01011111
01101011 **1**	11001011 **0**	10110100
11111010 **0**	11001110 **1**	10110001
01100000 **0**	10101110 **0**	01010001
00000001 **0**	10101111 **0**	01010000
00000100 **0**	10101011 **0**	01010100
00000011 **0**	10101000 **0**	01010111
10000011 **1**	00101011 **1**	00101011
(a)	(b)	(c)

Fig. 4. (a) BI-encoded patterns with 30 transitions. (b) BITS-encoded patterns with 23 transitions. (c) `hihrTS`-encoded patterns with 26 transitions.

by the extra bus line, which calls for a change in the pinout and interface specification of the original chip. Another problem is that the overhead of BITS coding cannot be neglected, as will be illustrated in the implementation of the encoder and decoder circuits in the next section. We tackle these two problems by focusing on the first: the solution to the second problem is obtained as a byproduct.

The advantage of BITS coding mainly comes from the fact that it is highly probable that the occurrence of `1` or `0` dominates each pattern. This is graphically shown in Fig. 5(a), where the distribution of patterns on an 8-bit wide narrow bus for noisy speech signals, which are represented by 16-bit two's complement fixed-point numbers with an 8-bit integral part and an 8-bit fractional part, are shown. Note that the values along the horizontal axis show the values of the 8-bit patterns represented in decimal as if they were two's complement signed integers. Fig. 5(b) shows distributions for music signals (16-bit two's complement signed integers). Although different kinds of signals are used for Fig. 5(a) and (b), distributions of 8-bit patterns look very similar: patterns around `00 000 000` occur most frequently. This implies that, during BITS encoding, most inversions occur for the patterns near the left-hand side of `00 000 000`, where patterns with high weight (the number of `1`s) are concentrated.

Recognizing that a coding method that does not require spatial redundancy (redundant bus lines) implies a one-to-one and onto mapping from a space consisting of all n-bit patterns to the same space, we construct a mapping called an `hihr` mapping, which uses an identity function for the set of patterns on the right-hand half (from `00...0` to `01...1`). We map the set of patterns on the left-hand half (from `10...0` to `11...1`) to the same set with orders reversed (e.g., `10...0` is mapped to `11...1`). Thus, patterns near to the left-hand side of `00...0`, which have high weight and thus are candidates for inversion in BITS encoding, are mapped to those having low weight. Since the patterns have high frequency, reducing the weights of those patterns effectively reduces the total weight.

Note that the majority voter used in BITS encoding to decide whether to encode `1` or `0` as having a transition, which is the main overhead, is not needed in `hihrTS` encoding (`hihr` mapping followed by transition signaling.[4]) In other words, `hihrTs` encoding [see Fig. 1(c)] resorts to the MSB for a *guess*, instead of counting the number of `1`s to determine whether to invert a pattern or not. Specifically, in `hihrTS` encoding, the `hihr`-mapped version of X_i, denoted by Y_i, is obtained by

$$Y_i = \begin{cases} x_i^{n-1}|X_i(n-1), & \text{if } x_i^{n-1} = 0 \\ x_i^{n-1}|\overline{X_i(n-1)}, & \text{otherwise} \end{cases} \tag{1}$$

where

x_i^{n-1} MSB of X_i;

$|$ concatenate operation;

$X_i(n-1)$ lower $n-1$ bits of X_i.

Then, the `hihrTS`-encoded version of X_i, denoted by Z_i, is obtained by

$$Z_i = y_i^{n-1}|TS(Y_i(n-1), Z_{i-1}(n-1)) \tag{2}$$

where $TS(x, y)$ denotes a transition encoding of x with respect to y.

The decoding process of `hihrTS` can be carried out by

$$X_i = \begin{cases} y_i^{n-1}|\overline{Y_i(n-1)}, & \text{if } y_i^{n-1} = 0 \\ y_i^{n-1}|Y_i(n-1), & \text{otherwise} \end{cases} \tag{3}$$

where $Y_i = z_i^{n-1}|TS^{-1}(Z_i(n-1), Z_{i-1}(n-1))$. The application of `hihrTS` encoding for our example is illustrated in Fig. 4(c).

The main observation behind `hihrTS` encoding is that it is highly probable that the guess is correct when the MSB corresponds to the actual sign of the pattern [$X_0, X_2, X_4, \ldots$ in Fig. 3(c)]. Furthermore, even for the remaining patterns [$X_1, X_3, X_5, \ldots$ in Fig. 3(c)], the probability of an incorrect guess[5] is kept low. Specifically, if an n-bit wide pattern is completely random, it can be shown that the probability of an incorrect guess is given by

$$\frac{2\left(C_{n-1}^{n/2+1} + C_{n-1}^{n/2+2} \cdots C_{n-1}^{n-1}\right)}{2^n} = \frac{1}{2} - 2^{1-n} C_{n-1}^{n/2}. \tag{4}$$

For example, the probability is 0.227 for an 8-bit pattern and 0.125 for a 4-bit pattern. Although `hihrTS` encoding obtains less transition reduction due to incorrect guesses, the overall power consumption (including the power consumed by the encoder itself) is comparable to that of BITS encoding because the `hihrTS` encoder consumes less power than the BITS encoder. More importantly, the extra bus line is not used in `hihrTS` coding and, as we see in the next section, both the delays of the encoder and the decoder for `hihrTS` coding are below `1 ns`,

[4]In the coding framework proposed in [6], `hihrTS` encoding is equivalent to $f_1 =$ `hihr` mapping and $f_2 = \text{identity}$ without function F.

[5]Because inversion is not determined by the weight but by the MSB of a pattern in `hihrTS` encoding, even a pattern with its weight less than half the bus width is inverted if its MSB is equal to one. This is an incorrect decision because the number of transitions is increased rather than decreased.

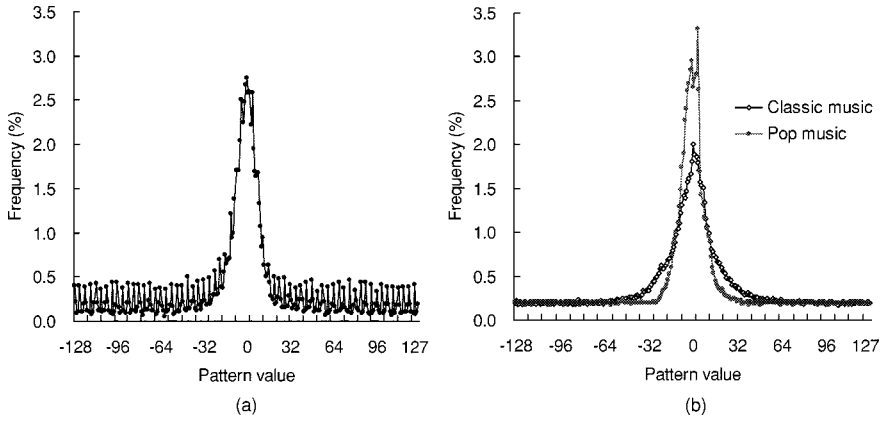

Fig. 5. Distributions of patterns on 8-bit wide narrow bus for (a) speech signals and (b) music signals.

thus lends itself to adaptation as a coding method in a broad range of circuit designs.

C. Experimental Results

To evaluate the efficiency of `hihrTS` encoding in bus transition reduction, we perform experiments for the following set of sample patterns.

- `noisy_speech` and `speech_output` : input and output signals from a noise canceller [11], which receives two signals (noisy speech and reference noise signals) as inputs and produces a noise-cancelled speech signal as an output. Each signal is encoded as a 16-bit two's complement fixed-point with an 8-bit integral part and an 8-bit fractional part.
- `pop_music` and `classic_music` : music signals from WAVE files, which store 16-bit PCM samples as two's complement signed integers.
- `fft_rdata` and `fft_idata` : patterns of real and imaginary parts of data processed by a 128-point complex FFT processor, which is one of the blocks in an audio decoder that is designed with VHDL [12]. Each pattern is encoded as a 20-bit two's complement signed integers, and is extracted through VHDL simulation.
- `dha` : the temporary result of speech processing from a DSP core, which is a part of an industrial example, a digital hearing aid (DHA) system [13], to be stored in external SRAM. The pattern encoded as 16-bit two's complement signed integers is extracted through VHDL simulation.

For off-chip communication using a narrow bus, we assume that two cycles are needed to transfer each pattern, meaning that the width of the narrow bus is half the width of each pattern.

The resulting percentage reduction in the number of transitions compared to the unencoded case for BI, BITS, and `hihrTS` is shown in Table I. BITS encoding obtains a substantial reduction compared to unencoded patterns and BI-encoded patterns. `hihrTS` encoding performs as well as BITS, although it relies on a fairly simple encoding mechanism and does not use the extra bus line. This is an important result because, if we take the power consumed by coding logic into account, `hihrTS` coding obtains similar power saving (in some cases, more power saving) without the extra bus line. We will elaborate on this in more detail in the next section.

TABLE I
COMPARISON OF PERCENTAGE TRANSITION REDUCTION

Pattern name	BI	BITS	pbm	hihrTS
noisy_speech	18.6	34.3	37.8	26.5
speech_output	15.8	47.3	46.1	40.6
pop_music	18.4	36.5	35.6	29.6
classic_music	19.0	34.0	31.8	27.4
fft_rdata	17.9	52.5	42.6	44.2
fft_idata	17.7	52.2	43.0	46.1
dha	20.8	47.1	40.0	41.5
average	18.3	43.4	39.6	36.6

We also report the result obtained with probability-based mapping (pbm) [5], [6] in the fourth column, although it is of limited practical use due to the severe complexity of the associated encoding and decoding logic. Because pbm requires a representative data set to obtain a mapping function, we use one half of each pattern to obtain the mapping function, and the other half to obtain the result of percentage reduction in the number of transitions compared to unencoded case.

III. IMPLEMENTATION OF CODING LOGIC

Bus coding inherently introduces area, delay, and power overheads due to the encoding and decoding circuits. This overhead should be kept as low as possible in order for the coding to be used in a broad class of circuit designs. We now present the implementation of encoding and decoding circuits for BI, BITS, and `hihrTS` codings for an 8-bit wide bus, and compare their power, area, and delay overheads. The coding circuits are designed with VHDL and synthesized using the synopsys design compiler. Layouts are obtained using the cadence silicon ensemble. The circuits are mapped on to a $0.35-\mu$m, 3.3-V gate library developed for the TSMC $0.35-\mu$m CMOS process. We assume a 100-MHz clock frequency.

Note that the latches and output drivers on the encoder side, except for those involved in line I of BI and BITS encoders (see

TABLE II
COMPARISON OF ENCODERS AND DECODERS

	Encoder			Decoder		
	BI	BITS	hihrTS	BI	BITS	hihrTS
Area (μm^2)	19076	18626	4659	2662	11392	9968
Delay (ns)	3.29	3.87	0.38	0.15	0.38	0.38
Power (μW)	2309	2409	411	120	2102	1618

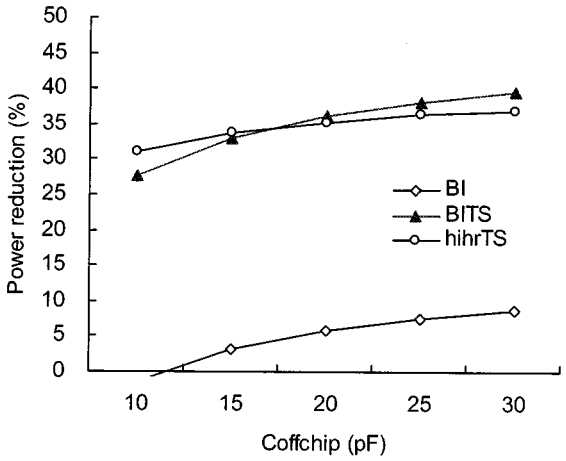

Fig. 6. Comparison of the overall power saving.

Fig. 1), are also present in the unencoded case and, thus, do not contribute to the overhead of the encoder and decoder circuits.

The area, delay, and power consumption of the encoder and decoder for each coding method are summarized in Table II. The power is simulated using IRSIM, with the pattern `speech_output` used as input vectors. The delay is measured using HSPICE. The delay of BITS encoding may limit the application of BITS coding to systems that are already delay optimized, or clocked at very high speed. `hihrTS` takes much less than `1 ns` for each encoding and decoding operation.

The power consumed by off-chip driving, denoted by P_{enc}, consists of two parts [1]. One is the power used to drive off-chip capacitance, bonding wires, and the pad capacitance. The other is the power consumed by the encoder, latches, and output drivers. To evaluate P_{enc}, each circuit in Fig. 1 is loaded with the total off-chip capacitance, denoted by C_{offchip}. Then the circuits are simulated with IRSIM, applying example patterns as input vectors. The power consumed by the decoder side, denoted by P_{dec}, is also obtained by applying the encoded version of example patterns as input vectors to each circuit in Fig. 2. The results are shown in Fig. 6 for `speech_output`. We vary C_{offchip} from 10 pF, which is typical for multichip module technology, to 30 pF, which is for an advanced package and PCB [1], and compare the percentage reduction in the total power consumption ($P_{\text{enc}} + P_{\text{dec}}$) for each coding method, compared to the uncoded case.

Compared to Table I, which only counts the number of transitions and, thus, does not take the effect of encoders and decoders into account, the difference between BITS and `hihrTS` is very small, due to the reduced power consumption of the `hihrTS` encoder. In particular, `hihrTS` gets more power saving when C_{offchip} is smaller than 20 pF, meaning that `hihrTS` can be used even for on-chip data buses having moderate capacitive loading.

IV. CONCLUSION

In this paper, we address bus coding methods, targeting highly integrated low-power systems incorporating narrow buses. For data patterns found in typical DSP applications, we show that transition signaling combined with BITS coding can achieve significant power saving for narrow buses. Since the application of BITS coding in circuit design is limited by the extra bus line and the overhead of the encoder and decoder circuits, we propose `hihrTS` coding, which does not require the extra bus line but retains the advantage of BITS coding. Although `hihrTS` coding employs a much simpler encoder circuit, overall power saving is shown to be comparable with BITS.

ACKNOWLEDGMENT

The authors would like to thank K. Lee, Seoul National University, Korea, for assistance in the experiments.

REFERENCES

[1] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE J. Solid-State Circuits*, vol. 29, pp. 663–670, June 1994.
[2] R. J. Fletcher, "Integrated circuit having outputs configured for reduced state changes," U.S. Patent 4 667 337, May 1987.
[3] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 49–58, Mar. 1995.
[4] ——, "Limited-weight codes for low-power I/O," in *Proc. Int. Workshop Low-Power Design*, Napa, CA, Apr. 1994, pp. 209–214.
[5] D. W. Faulkner, "PCM signal coding," U.S. Patent 5 062 152, Oct. 1991.
[6] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "A coding framework for low-power address and data busses," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 212–221, June 1999.
[7] C. L. Su, C. Y. Tsui, and A. M. Despain, "Low power architecture design and compilation technique for high-performance processors," in *Proc. IEEE COMPCON*, San Francisco, CA, Feb. 1994, pp. 209–214.
[8] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems," in *Proc. Great Lakes Symp. VLSI*, Urbana-Champaign, IL, Mar. 1997, pp. 77–82.
[9] L. Benini, G. D. Micheli, E. Macii, M. Poncino, and S. Quer, "System-level power optimization of special purpose applications: The Beach Solution," in *Proc. Int. Symp. Low-Power Electronics and Design*, Monterey, CA, Aug. 1997, pp. 24–29.
[10] Y. Shin, S. Chae, and K. Choi, "Partial bus-invert coding for power optimization of system level bus," in *Proc. Int. Symp. Low-Power Electronics and Design*, Monterey, CA, Aug. 1998, pp. 127–129.
[11] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast prototyping of datapath-intensive architectures," *IEEE Design Test Comput.*, pp. 40–51, June 1991.
[12] S. Lee and W. Sung, "A parser processor for MPEG-2 audio and AC-3 decoding," in *Proc. Int. Symp. Circuits and Systems*, Hong Kong, June 1997, pp. 2621–2624.
[13] H. Jang, S. Kim, and Y. Chang, "A digital signal processor for low power," in *Proc. IEEE Asia Pacific Conf. Application Specific Integrated Circuits (ASICs)*, Seoul, Korea, Aug. 1999.

Youngsoo Shin, photograph and biography not available at the time of publication.

Kiyoung Choi, photograph and biography not available at the time of publication.

Young-Hoon Chang, photograph and biography not available at the time of publication.