Narrow Bus Encoding for Low Power Systems

Youngsoo Shin and Kiyoung Choi
School of Electrical Engineering
Seoul National University
Seoul 151-742, Korea

Abstract

High integration in integrated circuits often leads to the problem of running out of pins. Narrow data buses can be used to alleviate this problem at the cost of performance degradation due to wait cycles. In this paper, we address bus coding methods for low power core-based systems incorporating narrow buses. Although the conventional Bus-Invert code performs well for completely random patterns, we show that transition signaling combined with Bus-Invert, which we call BITS coding, can achieve much more power saving for data patterns of typical DSP applications. The application of BITS coding to a real circuit design is limited by the overhead of the encoder and decoder circuits and the extra bus line introduced. We propose an approximate version of BITS coding, which do not require the extra bus line while retaining the advantage of BITS coding.

1 Introduction

Recently, a core-based design has become prevalent in digital systems design to cope with ever increasing time-to-market pressure. A system designed with cores often contains a lot of components, such as core processors, DSPs, and ASICs. As an example of QUALCOMM MSM3000, which is widely used in wireless systems, ARM7TDMI microprocessor core, a DSP, a CDMA and a DFM processor, and several peripheral interfaces are integrated into the same chip. The number of pins, which directly contributes to cost of a chip, is one of problems for such a high integration because it easily increases with the increasing number of components integrated into the same chip [1]. Communication with off-chip devices via narrow data buses is one of solutions to reduce the number of pins at the cost of reduced performance due to increased wait states. For the example of ARM7TDMI, which is a 32-bit microprocessor core, three kinds of data transfer sizes (Byte, Halfword, and Word) are provided to allow the core to be interfaced to 8, 16, or 32-bit wide memory systems.

The power consumption is also one of problems for such a high integration, especially for portable systems such as cellular phones and PDAs. It is a well known fact that a lot of power is consumed during off-chip bus driving due to the large off-chip driver, the pad capacitance, and the large off-chip capacitance [2]. Specifically, the capacitive load at the input/output of a chip is usually much larger (around three orders of magnitude) than that of the internal nodes. As a consequence, a considerable amount of power can be saved by a bus encoding, which reduces power consumption of a bus through encoding information transferred on the bus in such a way that the encoded version has less transitions compared to the original one.

In this paper, we study bus coding schemes for low power core-based systems incorporating narrow buses[1]. While various bus coding techniques have been proposed to reduce the peak or the average power consumption during off-chip communication [3], [4], [5], [6], [7], [8], [9], none of them has addressed explicitly the problem of coding patterns on a narrow bus. Compared to patterns transferred on the bus of the same width, those on the narrow bus exhibit a different property in that correlations between consecutive patterns, if they exist, are entirely lost. We show that transition signaling combined with Bus-Invert (BI) coding [5], which we call *BITS* coding in this paper, is particularly suitable for this situation.

The power reduction with bus coding is obtained at the cost of power, delay, and area overhead of encoding and decoding circuits. For BITS coding, the overhead mainly comes from a majority voter, which is a circuit for decision making. In order to reduce the overhead while retaining the advantage of BITS coding, we propose an approximate version of BITS coding, called *ABITS* coding. More importantly, ABITS coding does not use any spatial and temporal redundancy thus provides an efficient coding method in a broad-level circuit design. To verify the advantage of ABITS coding in circuit design, we implement coding logics for BI, BITS, and ABITS codings, and compare power, delay, and area overhead.

The rest of the paper is organized as follows. In the next section, we review the related work which focus on the reduction of bus transitions by bus coding. In section 3, we explain BITS and ABITS codings and their advantages, and present experimental results for some examples of patterns. In section 4, we present the implementation results of encoding and decoding circuits for BI, BITS, and ABITS codings and compare their power, area, and delay overhead. In section 5, we draw conclusions.

2 Related Work

The BI code [10], [5] is appropriate for uncorrelated data patterns, i.e. for patterns randomly distributed both in time and in space. In the BI coding, if the Hamming distance (the number of bits resulting in a transition) between the new pattern to be transferred and the old one currently on the bus (also counting the transition on the extra bus line, called *invert*) is larger than half the bus width, then the new pattern is transferred with each bit inverted. The *invert* line is used to inform the receiver side whether the pattern is inverted or not. However, it is not effective for correlated data patterns and for buses of larger widths.

For instruction address patterns, where consecutive patterns are often sequential, the Gray code is efficient [3]. In the T0 code [7], the bus transitions are further reduced by freezing the address lines when consecutive patterns are found to be sequential. An extra bus line is employed to inform the receiver side whether or not the current pattern is sequential.

In special-purpose applications, where the information about the sequence of patterns is available a priori, the characteristics

[1] In this paper, we define a *narrow bus* as a bus transferring patterns whose width is larger (typically multiples of 2) than that of the bus, thus transferring each pattern in multiple cycles.

 ISBN 0-7803-5974-7

of patterns can be exploited to efficiently reduce bus transitions. The Beach Solution [8] makes clusters of bus lines based on statistical information of address patterns and then generates an encoding function for each cluster such that the encoded version of each cluster results in less transitions. For data address patterns which are less sequential than instruction address patterns and less random than data patterns, the Partial Bus-Invert (PBI) code [9] performs better. It applies BI coding to a pre-selected sub-group of bus lines thereby avoiding unnecessary inversion of relatively inactive and/or uncorrelated bus lines.

While the conventional level signaling encodes logic `1` as high level voltage and logic `0` as low level voltage (or vice versa), the transition signaling [4] encodes logic `1` as having a transition and logic `0` as lack of transition (or vice versa). For example, if the pattern currently on the bus is `0110` and the one to be transferred is `0101`, it is encoded into `0011`. It is efficient with respect to power consumption when the probability of `0` (or `1`) is very high because the number of transitions becomes that of `1` (or `0`) after transition encoding. We can increase the probability of `0` (or `1`) by applying BI encoding[2].

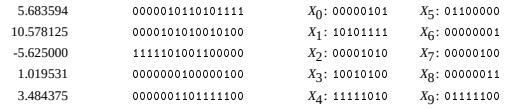

Fig. 1. An example of patterns from human voice. (a) Decimal representation. (b) 16-bit two's complement fixed point representation with 8-bit integral part (the leftmost bit being MSB and the rightmost bit being LSB). (c) The same patterns on 8-bit wide bus with 37 transitions.

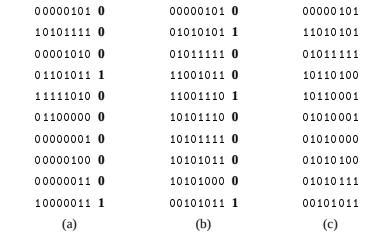

Fig. 2. (a) BI-encoded patterns with 30 transitions. (b) BITS-encoded patterns with 23 transitions. (c) ABITS-encoded patterns with 26 transitions.

3 Coding a Narrow Bus

3.1 BI and BITS codings

If data patterns are randomly distributed in time and mutually independent in space, a transfer via narrow bus, that is transferring each pattern in multiple cycles, does not destroy the original randomness of patterns. BI code still performs well in this case. Actually, its performance increases because BI code performs better for small bus width [5].

When data patterns do not follow this ideal property, as is often the case with data patterns in signal and image processing applications, BI code is not the best choice. Consider an example of decimal representation in Fig. 1(a), which is drawn from samples of human voice. Fig. 1(b) shows a typical 16-bit two's complement fixed point representation with 8-bit integral part and fractional part[3]. As shown in the figure, the least significant (LS) bits tend to be random whereas the most significant (MS) bits tend to have high *spatial* and *temporal* correlations due to sign extension, though the boundary is not clearly defined. This behavior of patterns are commonly observed in DSP applications such as signal and image processing algorithms and even used as a model for power estimation [11]. However, the temporal correlations at the MS bits are lost when the original patterns are transferred on a narrow bus, say 8-bit wide bus in this example, since the MS bits having spatial correlations and LS bits having randomness appears on bus lines alternately, as illustrated in Fig. 1(c), where X_i denotes a pattern at time *i*.

If we apply BI encoding for the patterns in Fig. 1(c), the number of transitions are reduced from 37 to 30 as illustrated in Fig. 2(a), where the rightmost bit indicates the value on the *invert* line. Further reduction can be obtained if we apply BITS encoding as illustrated in Fig. 2(b). This is because it is highly probable that the occurrence of `1` or `0` dominates at each pattern due to sign extension (see Fig. 1(c)), and BITS encodes `0` as having a transition in the former case and `1` as having a transition in the latter case. That is, in BITS encoding, if the number of `1`'s in X_i is larger than half the bus width, each bit of X_i is inverted (with the *invert* line set to `1`) and then transition-encoded. Otherwise, each bit of X_i is transition-encoded as is. More precisely, for a pattern at time *i* (X_i), its BI-encoded version is given by

$$Y_i|I = \begin{cases} X_i|0, & \text{if } w(X_i) \le n/2 \\ \overline{X_i}|1, & \text{otherwise} \end{cases} \quad (1)$$

where $|$ denotes a concatenate operation, I denotes the value at the *invert* line, $w(X_i)$ denotes the number of `1`'s in X_i, and n denotes the bus width. Then, the BITS-encoded version of X_i is given by

$$Z_i|I = TS(Y_i, Z_{i-1})|I, \quad (2)$$

where $TS(x,y)$ denotes a transition encoding of x with respect to y[4].

The decoding process can be carried out by

$$X_i = \begin{cases} Y_i, & \text{if } I = 0 \\ \overline{Y_i}, & \text{otherwise} \end{cases} \quad (3)$$

where $Y_i = TS^{-1}(Z_i, Z_{i-1})$. Note that both $TS(x,y)$ and its inverse $TS^{-1}(x,y)$ can be implemented by XORing x and y.

3.2 ABITS coding

Although substantial power saving can be obtained with BITS coding, its overhead can not be neglected as will be illustrated with the implementation of encoder and decoder circuits in the next section. Furthermore, its application to real circuit design is limited by the extra bus line, which calls for change in pinout and interface specification of the original chip. Because the main overhead comes from a majority voter, that decides whether to

[2] In normal BI encoding (BI encoding with level signaling), the decision for inverting depends on the number of transitions between the pattern to be transferred (X_i) and the one on the bus. In BITS encoding (BI encoding with transition signaling), the decision is made based on the number of `1`'s in X_i.

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

[4] TS is commutative, i.e. $TS(x,y) = TS(y.x)$.

encode 1 or 0 as having a transition (equation (1)), we eliminate the voter but resort to a *guess*. In the proposed ABITS encoding, the guess relies on the MSB of each X_i, denoted by x_i^{n-1}, meaning that MSB takes over the function of invert line thus eliminating the need for the extra bus line. More precisely, in ABITS encoding, Y_i is given by

$$Y_i = \begin{cases} x_i^{n-1} | X_i(n-1), & \text{if } x_i^{n-1} = 0 \\ x_i^{n-1} | \overline{X_i(n-1)}, & \text{otherwise} \end{cases} \quad (4)$$

where $X_i(n-1)$ denotes lower $n-1$ bits of X_i. Then, the ABITS-encoded version of X_i is obtained by

$$Z_i = y_i^{n-1} | TS(Y_i(n-1), Z_{i-1}(n-1)). \quad (5)$$

The application of ABITS encoding for our example is illustrated in Fig. 2(c). The main observation behind ABITS encoding is that it is highly probable that the guess is correct when x_i^{n-1} corresponds to the actual sign of the pattern ($X_0, X_2, X_4, \ldots$ in Fig. 1(c)). Furthermore, even for the remaining patterns ($X_1, X_3, X_5, \ldots$ in Fig. 1(c)), the probability of incorrect guess is kept low. Specifically, if an n-bit wide pattern is completely random, the probability of incorrect guess is given by

$$\frac{2(C_{n-1}^{n/2+1} + C_{n-1}^{n/2+2} \cdots C_{n-1}^{n-1})}{2^n} = \frac{1}{2} - 2^{1-n} C_{n-1}^{n/2}. \quad (6)$$

As a numerical example, the probability is 0.125 for a 4-bit pattern and 0.227 for an 8-bit pattern. The probability is 0 for a 2-bit pattern, meaning that ABITS encoding becomes equivalent to BITS encoding. Although ABITS encoding obtains less transition reduction due to incorrect guess, the overall power consumption (including the power consumed by the encoder itself) is fairly comparable to that of BITS encoding because the ABITS encoder consumes less power than BITS encoder. More importantly, the spatial redundancy is not used in ABITS coding, and as will be presented in the next section, both the delays of the encoder and the decoder for ABITS coding are below 1 *ns*, thus lends itself to adaptation as a coding method in a broad range of circuit design.

3.3 Experimental Results

To evaluate the efficiency of the ABITS encoding in bus transition reduction, we perform two experiments. The first experiment is with a noise canceller, an example from Hyper [12], which receives two signals (noisy speech and reference noise signals) as inputs and produces a noise-cancelled speech signal as an output. We encode each signal as 16-bit two's complement fixed point with 8-bit integral part. We assume 8-bit wide bus for off-chip communication thus 2 cycles for transferring each pattern. The results are shown in Table 1 for two patterns, named `noisy_speech` and `speech_output`. The result of ABITS encoding is compared to those of BI and BITS encodings. Note that BITS encoding is a lower bound to ABITS encoding with respect to the number of transitions.

The second experiment is with data patterns extracted from an audio decoder, which is designed with VHDL and supports MPEG-2 audio and AC-3 standard with programmability [13]. Through VHDL simulation, we extract 40-bit wide data patterns (20-bits of real and imaginary parts, respectively) handled by an *FFT processor*, which computes 128-point complex FFT. We assume one 10-bit wide bus for off-chip communication for each of real and imaginary parts. The results are shown in Table 2 for two patterns, named `fft_rdata` and `fft_idata`.

Table 3. Comparison of area, delay, and power of encoders and decoders

	Encoder			Decoder		
	BI	BITS	ABITS	BI	BITS	ABITS
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

The results show that BITS encoding obtains substantial reduction compared to unencoded patterns and BI-encoded patterns. The difference between BITS and ABITS encodings is not that significant, though the latter resort to fairly simple prediction mechanism.

4 Implementation of Coding Logic

Bus coding inherently introduces area, delay, and power overhead due to encoding and decoding circuits. Thus, the overhead should be kept as low as possible in order to be used in broad class of circuit design. In this section, we present the implementation results of encoding and decoding circuits for BI, BITS, and ABITS codings for 8-bit wide bus and compare their power, area, and delay overhead. The coding circuits are designed with VHDL followed by functional simulation and synthesized using Synopsys Design Compiler. Layouts are obtained using Cadence Silicon Ensemble. The circuits are mapped onto a 0.35 μm, 3.3 V gate library developed for the TSMC 0.35 μm CMOS process. We assume a 100 MHz clock frequency.

The area, delay, and power of the encoder and decoder for each coding method are summarized in Table 3. The power is simulated using IRSIM with the pattern `noisy_speech` in Table 1 used as input vectors. The delay is measured using HSPICE. Although the delay of BITS encoding is much less than the clock period (10 *ns* in the 100 MHz system), it may limit the application of BITS coding to systems which are already delay-optimized, which are clocked at very high speed, and so on. ABITS takes less than 1 *ns* for each of its encoding and decoding.

To evaluate the overall power consumed during off-chip driving, which includes the power consumed by encoder, latches, and output drivers[5], each output driver is loaded with *output capacitance*, which is the sum of pad capacitance and off-chip capacitance. Then the entire circuit of each coding method is simulated with IRSIM applying the same patterns in Table 1 as input vectors. The results are summarized in Table 4. We vary the output capacitance[6] and compare the overall power consumption. The third column corresponds to the power consumption of circuits containing only latches and output drivers. Compared to Table 1, the difference between BITS and ABITS is very small due to less power consumption of the ABITS encoder.

[5]The latches and output drivers at the encoder side and the input drivers at the decoder side (except for those involved in the *invert* line) are also present in the unencoded case.

[6]The power consumed by off-chip driving consists of the power used to drive off-chip capacitance, bonding wires, and the pad capacitance and the power consumed by the driver [2]. In the experiment, we vary the output capacitance from 10 pF, which is typical for multichip module technology, to 30 pF, which is for advanced package and advanced PCB.

Table 1. Comparison of the total number of bus transitions

Pattern Names	# patterns	Unencoded # trans.	BI encoding		BITS encoding		ABITS encoding	
			# trans.	% red.	# trans.	% red.	# trans.	% red.
noisy_speech	3276	13133	10686	18.6	8632	34.3	9650	26.5
speech_output	3276	12375	10423	15.8	6518	47.3	7348	40.6

Table 2. Comparison of the total number of bus transitions

Pattern Names	# patterns	Unencoded # trans.	BI encoding		BITS encoding		ABITS encoding	
			# trans.	% red.	# trans.	% red.	# trans.	% red.
fft_rdata	1566	7767	6375	17.9	3690	52.5	4337	44.2
fft_idata	1566	7714	6351	17.7	3684	52.2	4160	46.1

Table 4. Comparison of the overall power consumption during off-chip driving

Pattern Names	Output cap. (pF)	Unencoded power (mW)	BI encoding		BITS encoding		ABITS encoding	
			P (mW)	Red. (%)	P (mW)	Red. (%)	P (mW)	Red. (%)
noisy_speech	10	29.5	29.1	1.4	23.0	22.0	23.1	21.7
	15	40.4	38.0	5.9	30.2	25.2	31.1	23.0
	20	51.4	46.9	8.8	37.3	27.4	39.1	23.9
	25	62.4	55.8	10.6	44.5	28.7	47.2	24.4
	30	73.3	64.7	11.7	51.7	29.5	55.2	24.7
speech_output	10	27.9	28.2	-1.1	18.1	35.1	18.0	35.5
	15	38.2	36.9	3.4	23.5	38.5	24.1	36.9
	20	48.5	45.6	6.0	28.9	40.4	30.2	37.7
	25	58.9	54.3	7.8	34.3	41.8	36.3	38.4
	30	69.1	63.0	8.8	39.7	42.5	42.4	38.6

5 Conclusion

In this paper, we address bus coding methods targeting highly integrated low power systems incorporating narrow buses. When the original data patterns are completely random, the conventional Bus-Invert (BI) coding still performs well for narrow buses. Regarding data patterns found in typical DSP applications, we show that transition signaling combined with Bus-Invert (BITS) coding can achieve significant power saving for narrow buses. Since the application of BITS coding in circuit design is limited by the overhead of the encoder and decoder circuits and the extra bus line, we propose ABITS coding, which is an approximate version of BITS coding. Although the ABITS coding employs a much simpler encoder circuit, overall power saving is shown to be comparable with that of BITS.

Acknowledgment

The authors would like to thank Seokjun Lee and Prof. Wonyong Sung for providing us with example patterns of an audio decoder and Kangyun Lee for providing us with a gate library developed for the TSMC 0.35 μm CMOS process.

References

[1] O. Gunasekara, "Developing a digital cellular phone using a 32-bit microcontroller," Tech. Rep., Advanced RISC Machines Ltd., 1998.

[2] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, June 1994.

[3] 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*, Feb. 1994, pp. 209–214.

[4] M. R. Stan and W. P. Burleson, "Limited-weight codes for low-power I/O," in *Proc. Int'l Workshop on Low Power Design*, Apr. 1994, pp. 209–214.

[5] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. on VLSI Systems*, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[6] M. R. Stan and W. P. Burleson, "Low-power encodings for global communication in CMOS VLSI," *IEEE Trans. on VLSI Systems*, vol. 5, no. 4, pp. 444–455, Dec. 1997.

[7] L. Benini, G. De 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 Symposium on VLSI*, Mar. 1997, pp. 77–82.

[8] L. Benini, G. De Micheli, E. Macii, M. Poncino, and S. Quer, "System-level power optimization of special purpose applications: The Beach Solution," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1997, pp. 24–29.

[9] Y. Shin, S. Chae, and K. Choi, "Partial bus-invert coding for power optimization of system level bus," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1998, pp. 127–129.

[10] R. J. Fletcher, "Integrated circuit having outputs configured for reduced state changes," May 1987, U.S. Patent 4667337.

[11] P. Landman and J. Rabaey, "Architectural power analysis: the dual bit type method," *IEEE Trans. on VLSI Systems*, vol. 3, no. 2, pp. 173–187, June 1995.

[12] J. Rabaey, C. Chu, P. Hoang, and M Potkonjak, "Fast prototyping of datapath-intensive architectures," *IEEE Design & Test of Computers*, pp. 40–51, June 1991.

[13] S. Lee and W. Sung, "A parser processor for MPEG-2 audio and AC-3 decoding," in *Proc. Int'l Symposium on Circuits and Systems*, June 1997, pp. 2621–2624.