Transactions Briefs

Partial Bus-Invert Coding for Power Optimization of Application-Specific Systems

Youngsoo Shin, Soo-Ik Chae, and Kiyoung Choi

***Abstract*—This paper presents two bus coding schemes for power optimization of application-specific systems: *Partial Bus-Invert* coding and its extension to *Multiway Partial Bus-Invert* coding. In the first scheme, only a selected subgroup of bus lines is encoded to avoid unnecessary inversion of relatively inactive and/or uncorrelated bus lines which are not included in the subgroup. In the extended scheme, we partition a bus into multiple subbuses by clustering highly correlated bus lines and then encode each subbus independently. We describe a heuristic algorithm of partitioning a bus into subbuses for each encoding scheme. Experimental results for various examples indicate that both encoding schemes are highly efficient for application-specific systems.**

***Index Terms*—Digital complementary metal–oxide–semiconductor (CMOS, low-power dissipation, memory, switching activity, system level, tradeoffs.**

I. Introduction

Recently, power consumption has been a critical design constraint in the design of digital systems due to widely used portable systems. Although the power consumption of a system can be reduced at various phases of the design process from system level down to process level, optimization at a higher level can often provide more power savings. Among the architectural components at the system level, buses that interconnect subsystems are important components, which consume a significant power. Especially, a great deal 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 [1]. Power consumed by off-chip driving becomes more dominant as devices are scaled down because the off-chip capacitance does not depend on process technology but depends on the package and PCB technologies. It becomes even more dominant if costs must be lowered by employing cheaper package. Therefore, a considerable amount of power can be saved by reducing power consumption in the bus.

In this paper, we propose a new bus coding scheme, called *Partial Bus-Invert (PBI)* coding, where the conventional bus-invert (BI) coding [2] technique is used but it is applied only to a selected subset of bus lines. We can select such a subset statically if the information about the sequence of memory access patterns is available after the algorithm of an application is specified. Consequently, we focus on data address buses of application-specific systems such as signal and image processing applications. We propose a heuristic algorithm that exploits both transition correlation and transition probability in order to find a subset of bus lines such that the total number of bus transitions are minimized. We also investigate the overhead effect of encoding/decoding circuits and propose a method of incorporating them in selecting a subbus for PBI coding. We also propose a variant of PBI coding, called *Multiway Partial Bus-Invert (MPBI)* coding, which selects multiple subbuses and encodes each subbus with BI coding independently. We present several experimental results of both PBI and MPBI codings and compare them with those of other coding schemes.

Manuscript received April 4, 1998; revised April 10, 2000. This work was supported in part by the Korea Research Foundation under a Nondirected Research Fund.

Y. Shin is with the Center for Collaborative Research and Institute of Industrial Science, University of Tokyo, Tokyo 106-8558, Japan.

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

Publisher Item Identifier S 1063-8210(01)00690-4.

II. Related Work and Motivation

There are various low-power coding methods for *data buses*: BI code [2] for uncorrelated data patterns and probability-based mapping [3] for patterns with nonuniform probability densities. For *instruction address patterns*, Gray code [4], T0 code [5], and inc-xor [3] are efficient. Working zone encoding [6] is well suited both for instruction and data address patterns. In application-specific systems, where the information about the sequence of patterns is available *a priori*, the characteristics of patterns can be exploited to efficiently reduce bus transitions. The Beach Solution [7] performs well in this case.

The behavior of *data addresses* is somewhat different from that of data itself or instruction addresses. First, they are less sequential than instruction addresses. In case of some memory-intensive applications such as image processing algorithms, it is mostly out of sequence. Second, we can hardly assume that data addresses are random even though they are more random than instruction addresses. Usually, the signal probability and/or transition probability of some of bus lines are biased toward 0 or 1, that is, some of the bus lines are far from random. Consequently, we are encouraged to exploit statistical information in order to efficiently reduce transitions on the data address buses.

The motivation of PBI coding is based on the observation that all the previously proposed coding schemes take the entire bus line into account for bus coding. However, the overhead of the encoding/decoding circuits increases with the number of bus lines involved in bus encoding. In PBI coding, we attain two goals at the same time: *minimizing the number of bus lines involved in bus coding* thereby minimizing the overhead and *minimizing the total number of bus transitions*.

III. Partial Bus-Invert Coding

A. Problem Formulation

BI coding requires one extra bus line, called *invert*, to inform the receiver side whether a current pattern is inverted or not. In BI coding, if the Hamming distance (the number of bits resulting in a transition) between the present pattern and the last pattern of the bus (also counting the transition on the *invert* line) is larger than half the bus width, the present pattern is transmitted with each bit inverted.

Now, consider that we encode only m lines out of total n bus lines leaving the remaining lines unencoded. For the patterns randomly distributed in time and mutually independent in space, the more bus lines are encoded with BI coding, the more reduction in bus transitions can be obtained. Specifically, let $E(m)$ be the expected number of transitions per encoded pattern when we take m out of n bus lines for BI coding while leaving the remaining bus lines unencoded. Then it can be shown that

$$E(m) = \frac{n}{2} - \sum_{i=m/2+1}^{m} (2i - m - 1) C_m^i \left(\frac{1}{2}\right)^m. \quad (1)$$

Fig. 1 shows graphically $E(m)$ versus m for a 16-b wide bus. $E(m)$ monotonically (but not strictly) decreases with m but the amount of

1063–8210/01$10.00 © 2001 IEEE

Authorized licensed use limited to: Seoul National University. Downloaded on August 24, 2009 at 01:55 from IEEE Xplore. Restrictions apply.

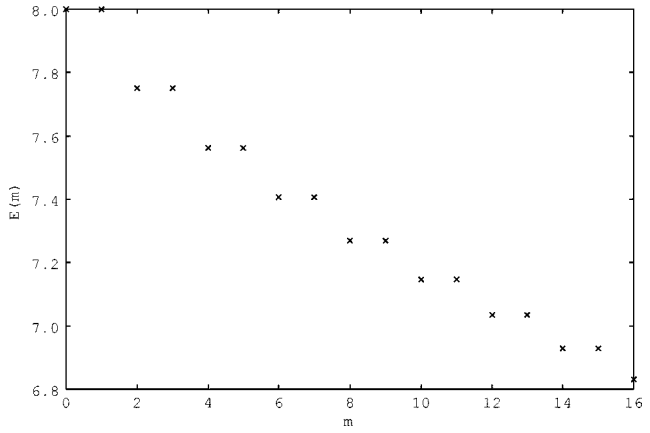

Fig. 1. The expected number of transitions for a 16-bit random pattern when m bus lines are involved in BI coding.

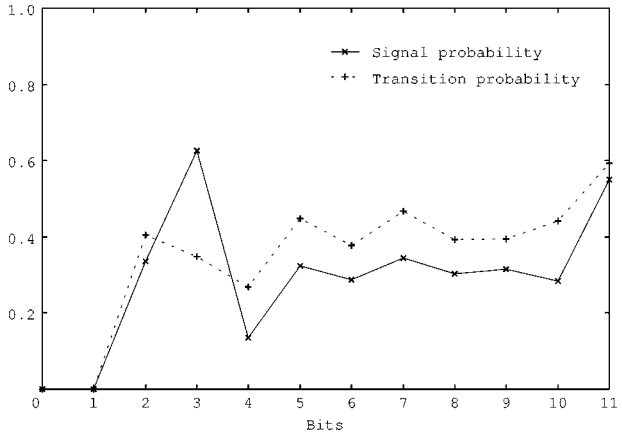

Fig. 2. Statistics of data address patterns from Linear Prediction algorithm.

decrease becomes smaller as m increases. In summary, we can obtain the maximum transition reduction when all the bus lines are involved in BI coding.

However, the monotonicity does not hold when the behavior of patterns deviates from random distribution and mutual independence. In other words, the minimum of the expected number of transitions may occur at any $m \in [0, n]$, which depends on the behavior of the patterns. This is especially the case for patterns on *data address buses*. In data address buses, some of the bus lines are usually *far* from random, meaning that it may be inefficient to encode those lines with BI coding. However, it is difficult to quantitatively determine the criterion of how far from random the bus lines should be if they are not to be included in BI coding. Let us take an example of *linear prediction* [8]. Fig. 2 shows the statistics of signal probability and transition probability of data address patterns obtained from typical runs of the linear prediction program. Obviously it is inefficient to include bit 0 and 1 of the bus for encoding because they are far from random. However, it is not easy to answer the following questions. *Does it help to include bit 4 or 11 for encoding? Which set of bus lines results in the minimum bus transitions when only the set is included in bus encoding?* This decision problem with the nonmonotonicity of $E(m)$ forms the optimization problem:

- given data address patterns of application-specific systems,
- select a subgroup of bus lines for BI coding such that the total number of transitions in the bus is minimized with/without including that of encoding/decoding circuits.

B. Overview

In PBI coding, we partition a bus $\mathcal{B}$ into two subbuses based on the behavior of patterns transferred. More precisely, we are given a bus $\mathcal{B} = (b^0, b^1, \ldots, b^{n-1})$, which transfers a sequence of patterns $B_i = (b_i^0, b_i^1, \ldots, b_i^{n-1})$, where i is the time index, n is the bus width, and b_i^j is the value of a bus line b^j at time i. We partition $\mathcal{B}$ into a selected subbus $\mathcal{S}$ and the remaining subbus $\mathcal{R}$ such that $\mathcal{S}$ contains bus lines having higher transition correlation and/or higher transition probability and $\mathcal{R}$ contains the remaining bus lines. Because the bus lines in $\mathcal{R}$ have low correlation with those in $\mathcal{S}$ and low transition activity, inverting those in $\mathcal{R}$ may increase rather than decrease the transition activity. Therefore, by applying BI coding only to the subbus $\mathcal{S}$, we can reduce the hardware overhead as well as decrease the total number of bus transitions.

Once $\mathcal{B}$ is partitioned, PBI coding is performed as follows: We compute the Hamming distance between S'_{i-1} and S_i (also counting a transition at the *invert* line), where S'_{i-1} is an encoded version of S_{i-1}. If it is larger than $|\mathcal{S}|/2$, set the *invert* line to 1 and invert the lines in S_i without inverting the lines in R_i. Otherwise, set *invert* $= 0$ and let B_i uninverted.

C. Selection Algorithm of the Subbus

The performance of PBI coding depends on the selection of the subbus $\mathcal{S}$. Unfortunately, it is intractable[1] to find an optimum set $\mathcal{S}_{\text{opt}} \subset \mathcal{B}$ such that PBI coding for $\mathcal{S}_{\text{opt}}$ results in the minimum number of total transitions. Thus, we propose a linear-time heuristic algorithm that explores only n configurations to find the one which results in the minimum number of total transitions by exploiting both transition correlation and transition probability.

For jth bus line, the *transition encoding* is defined as

$$t_i^j = \begin{cases} 1, & \text{if } b_{i-1}^j \neq b_i^j \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The *transition correlation coefficient* or simply *correlation coefficient* for two bus lines (jth and kth) is defined by

$$\rho_{jk} = \frac{K_{jk}}{\sigma_j \sigma_k} \tag{3}$$

where σ_j is the standard deviation of t^j. K_{jk} is the covariance of t^j and t^k and defined by

$$K_{jk} = E\{t_j t_k\} - m_j m_k \tag{4}$$

where $E\{x\}$ is the expected value of x and m_j is the mean of t_j.

The selection algorithm is outlined in Fig. 3. Initially, we select the line with the highest transition probability (L5) and then make the first configuration (L6). At each iteration of the **while** loop (L7), we select a line b^j that maximizes the sum of the transition probability of b^j and the average correlation coefficient between b^j and the lines already selected (L8) and then make a configuration (L9). Among the resulting configurations, we select the one that yields the minimum number of total bus transitions (L12).

As a selection metric, we use the transition probability together with the average of correlation coefficients with the bus lines already selected (L7 of Fig. 3), based on the observation that *the maximum gain can be obtained if we invert bus lines with high probability of having transitions together*. Fig. 4 shows the result of the algorithm for the 32-b wide data address patterns used in a lowpass filter [9]. The figure

[1]There are $C_n^0 + C_n^1 + \cdots C_n^n = 2^n$ possible configurations for PBI coding, where a *configuration* is defined as an ordered pair $(\mathcal{S}, \mathcal{R})$

Authorized licensed use limited to: Seoul National University. Downloaded on August 24, 2009 at 01:55 from IEEE Xplore. Restrictions apply.

begin

L1: Compute the transition probability of each bus line, $p_j, j = 0, 1, \ldots, n-1$;

L2: Compute the correlation coefficient of each pair of bus lines, $\rho_{jk}, j = 0, 1, \ldots, n-1, k = 0, 1, \ldots, n-1$;

L3: $\mathcal{S} = \{\}, \mathcal{R} = \{b^0, b^1, \ldots, b^{n-1}\}$;

L4: Initialize the configuration set $C = \{\}$;

L5: Select b^i with the highest transition probability;

L6: $\mathcal{S} = \{b^i\}, \mathcal{R} = \mathcal{R} - \{b^i\}, C = C \cup \{(\mathcal{S}, \mathcal{R})\}$;

L7: **while** $\mathcal{R} \neq \{\}$ **do**

L8: Select b^j such that $\frac{\sum_{b^k \in \mathcal{S}} \rho_{jk}}{|\mathcal{S}|} + p_j$ is a maximum;

L9: $\mathcal{S} = \mathcal{S} \cup \{b^j\}, \mathcal{R} = \mathcal{R} - \{b^j\}, C = C \cup \{(\mathcal{S}, \mathcal{R})\}$;

L10: **end do**

L11: Count the number of total transitions after PBI coding for each configuration in C;

L12: Select the configuration that yields the minimum number of total transitions;

end

Fig. 3. Selection algorithm of the subbus.

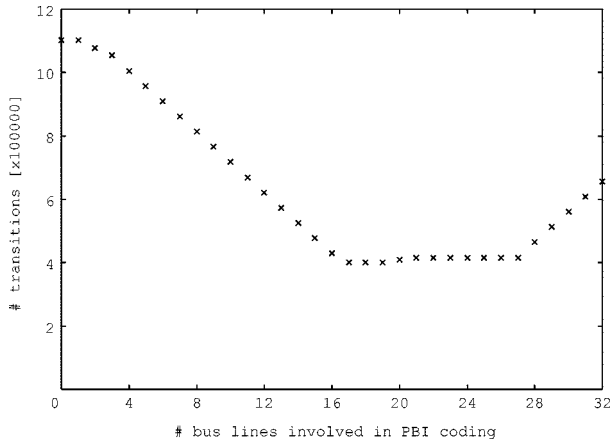

Fig. 4. The number of total transitions versus the number of bus lines involved in PBI coding in an example of low-pass filter.

indicates that as we add bus lines for PBI coding with the heuristic algorithm, transitions are reduced in a way similar to those of random patterns as expected by (1), which is plotted in Fig. 1. However, if we select more than 27 bus lines, the number of transitions increases sharply. This sharp increase is contributed by the bus lines whose transitions occur in a way relatively opposite to already selected bus lines and/or whose switching activity is very small.

Another fact, observed in Fig. 4, is that there is an interval where the variation of transitions is not significant meaning that the number of transitions is relatively independent on the number of bus lines selected for PBI coding. Consequently, if we take internal transitions of encoding/decoding circuits into account for power optimization, the best configuration may be found near the leftmost point of the interval rather than the point which yields the least transitions. In complementary metal–oxide–semiconductor (CMOS) circuits, the dynamic power is proportional to load capacitance and switching activity. Based on this property, we define the *total effective bus transitions*, denoted by T_{eff}, as follows:

$$T_{\text{eff}} = T_{\text{bus}} + \frac{C_{\text{int}}}{C_{\text{bus}}} T_{\text{int}} \tag{5}$$

where

- T_{bus} total bus transitions;
- T_{int} total number of transitions in the encoding/decoding circuits;
- C_{int} average capacitance of the node in the internal circuits;
- C_{bus} total off-chip capacitance per bus line.

By using (5), we count the number of effective transitions at L11 of Fig. 3 to include the effect of the encoding/decoding circuits.

While we can obtain the value of T_{bus} by simply counting the number of transitions from the encoded patterns, it is time-consuming to obtain the accurate value of T_{int}. However, such accuracy is not needed for our purpose because T_{int} is multiplied by a relatively small constant before it is added to T_{bus}. We take a probabilistic approach to estimate T_{int}. When we encode m bus lines with BI coding, the encoding logic requires $2m$ XOR gates and a majority voter with $m+1$ inputs (including the *invert* line) and the decoding logic requires m XOR gates. Because the majority voter contributes to most of the total number of transitions at the encoding/decoding circuits (T_{int}), we approximate T_{int} as the number of transitions in the majority voter. This assumption yields

$$T_{\text{int}} = \kappa N(m+1) a_p L \tag{6}$$

Authorized licensed use limited to: Seoul National University. Downloaded on August 24, 2009 at 01:55 from IEEE Xplore. Restrictions apply.

begin

L1: Compute the transition probability of each bus line, $p_j, j = 0, 1, \ldots, n-1$;

L2: Compute the correlation coefficient of each pair of bus lines, $\rho_{jk}, j = 0, 1, \ldots, n-1, k = 0, 1, \ldots, n-1$;

L3: Initialize the configuration set $C = \{\}$;

L4: **for** each $\rho_{th}, \rho_{min} \leq \rho_{th} \leq \rho_{max}, \rho_{th} = \rho_{th} + step$ **do**

L5: $C = C \cup$ *Generate_configuration*(ρ_{th});

L6: **end do**

L7: Count the number of total transitions after MPBI coding for each configuration in C;

L8: Select the configuration that yields the minimum number of total transitions;

end

Algorithm *Generate_configuration*(ρ_{th})

begin

L9: $R = \{b^0, b^1, \ldots, b^{n-1}\}$;

L10: **while** $R \neq \{\}$ **do**

L11: Select $b^i \in R$ such that p_i is a maximum;

L12: $R = R - \{b^i\}$;

L13: Make a sub-bus $\Psi = \{b^i\}$;

L14: **while** true **do**

L15: Select b^j such that $\frac{\sum_{b^k \in \Psi} \rho_{jk}}{|\Psi|} > \rho_{th}$ and $\frac{\sum_{b^k \in \Psi} \rho_{jk}}{|\Psi|}$ is a maximum;

L16: **if** b^j not found **then** exit loop;

L17: **else** $\Psi = \Psi \cup \{b^j\}, R = R - \{b^j\}$;

L18: **end if**

L19: **end do**

L20: **end do**

L21: Return $\bigcup \Psi$;

end

Fig. 5. Bus partitioning heuristic for MPBI coding.

where κ denotes gate equivalents of a full adder and $N(x)$ is the approximate number of full adders used in the majority voter with x inputs, given by

$$N(x) = x - 2. \tag{7}$$

The derivation can be found in Appendix A. a_p is the average transition probability of m bus lines and L is the number of patterns. There are approximately $\kappa N(m+1)$ gates in the majority voter with average input transition probability a_p. Therefore, there are approximately $\kappa N(m+1)a_p L$ transitions for L patterns.

Authorized licensed use limited to: Seoul National University. Downloaded on August 24, 2009 at 01:55 from IEEE Xplore. Restrictions apply.

TABLE I
RESULT FOR BENCHMARK EXAMPLES

Applications	Unenc.	BI	Heu.+PBI		SA+PBI		MPBI	Beach	Unenc.	BI	Heu.+PBI		SA+PBI	
	T_{bus}	Red.	Red.	$\|\mathcal{S}\|$	Red.	$\|\mathcal{S}\|$	Red.	Red.	T_{eff}	Red.	Red.	$\|\mathcal{S}\|$	Red.	$\|\mathcal{S}\|$
`compress`	1756468	39.3	58.9	20	58.9	20	64.8	58.6	1756468	34.8	55.7	16	56.0	15
`laplace`	3928218	39.5	59.2	19	59.2	19	63.1	55.3	3928218	35.0	56.0	16	56.3	15
`linear`	3948001	38.7	68.9	23	68.9	23	69.0	62.1	3948001	34.2	64.7	23	64.7	23
`lowpass`	1101927	40.5	63.7	18	63.7	20	67.8	61.4	1101927	35.9	60.4	17	60.6	16
`sor`	2874978	33.9	53.3	18	53.3	19	58.4	53.7	2874978	29.4	50.1	16	50.1	16
`wavelet`	2197	36.6	71.8	22	71.9	21	72.2	73.1	2197	32.0	67.7	21	68.3	19
Average		38.1	62.6	20.0	62.7	20.3	65.9	60.7		33.6	59.1	18.2	59.3	17.3

IV. MULTIWAY PARTIAL BUS-INVERT CODING

In PBI coding, we partition a bus into two subbuses and then encode only one subbus while leaving the remaining one unencoded. For this two-way partitioning, we heuristically take account of transition probability as well as transition correlation. PBI coding gives good results for a subbus as long as the lines in the subbus are highly correlated with respect to transition. Therefore, we can extend PBI coding for multiple subbuses to obtain more reduction in bus transitions. That is, we can partition a bus into multiple subbuses such that each subbus contains only bus lines that are highly correlated with each other.

In MPBI coding, we partition a bus into multiple subbuses and then apply BI coding independently for each subbus. We need at most k extra *invert* lines if a bus is partitioned into k subbuses. Because of the internal transitions due to encoding/decoding circuits, some of subbuses may increase total effective bus transitions rather than decrease it. Then we do not encode those subbuses at all. Note that the bus is partitioned based on the correlation so that the number of lines in each subbus is not uniform while a bus is uniformly partitioned in the partitioned BI coding [2].

Fig. 5 outlines the bus partitioning heuristic for MPBI coding. A *Generate_configuration* generates a configuration[2] given a threshold of correlation coefficient, denoted by ρ_{th}. It constructs a subbus at each iteration of **while** loop (L10). A subbus starts from a bus line which has the highest transition probability and is in the set (R) of lines that are not included in any clusters yet (L11, L12, and L13). It iteratively selects a line in R whose average transition correlation is maximum and larger than ρ_{th}. If such line dose not exist, a new subbus starts.

The set of resulting configurations highly depends on the value of ρ_{th}. However, the optimum value of ρ_{th}, which can generate the best configuration in terms of total transitions, depends on application. Therefore, we generate the configurations for a range of ρ_{th} (L4 and L5). When we count the number of transitions for each configuration (L7), we also include the effect of internal transitions due to encoding/decoding circuits by computing the total effective bus transitions.

V. EXPERIMENTAL RESULTS

In this section, we examine the efficiency of PBI and MPBI codings with three experiments. For the total effective bus transitions [see (5)], we assume 30 pF for $C_{\rm bus}$, 0.2 pF for $C_{\rm int}$, and 7 for κ [see (6)]. For MPBI coding, we construct 100 configurations for each example with $\rho_{\rm min} = 0.0$, $\rho_{\rm max} = 1.0$, and *step* $= 0.01$, and then select the best configuration that yields the minimum number of total bus transitions.

[2]In MPBI coding, a *configuration* is defined as a partition of the bus, which is a set of subbuses, where a subbus is defined as a set of bus lines.

A. Experiment with Benchmark Examples

We experiment with several benchmark applications [9] collected from typical image or signal processing algorithms, which are frequently implemented as application-specific systems. We assume 32-b wide data address buses for all the applications and extract the data address patterns issued by a SPARC processor. The result is shown in Table I, which is divided in two parts: comparing the total bus transitions $(T_{\rm bus})$ and comparing the total effective bus transitions $(T_{\rm eff})$. For each coding method, we show the percentage of reduction compared to unencoded case. For PBI coding, we also report the number of bus lines selected $(|\mathcal{S}|)$. The column with the heading SA+PBI corresponds to PBI coding after bus lines are selected using simulated annealing [10] instead of the heuristic algorithm. Also shown is the percentage of reduction with the Beach Solution [7], whose performance is better than working zone encoding [6] except for `lowpass`, in our experiments.

The reduction of bus transitions with PBI coding is 62.6% on the average and up to 71.8% compared to unencoded case and this is obtained by encoding only 20 out of 32 bus lines on the average. The result with SA indicates that performance of the heuristic algorithm is very satisfactory. The execution time of the heuristic is less than 3 min on Ultra 1. MPBI coding gives the best results for these examples with the number of subbuses in the range of 3∼5. The second part of Table I indicates that the number of bus lines selected for PBI coding $(|\mathcal{S}|)$ can be reduced further (18 on the average) if we take the effect of internal transitions due to encoding/decoding circuits into account.

B. Experiment with Examples from Audio Decoder

We experiment with data address patterns extracted from a realistic example of *audio decoder* [11], which is designed with VHDL and then synthesized with the LSI 10k gate library. Fig. 6 shows a block diagram of the audio decoder. The block marked *Parser processor* reads input data stored in a frame memory and uses data address of 16-b wide to access the external memory marked *Buffer*. We extract its data address patterns through VHDL simulation. Another patterns are extracted from a block marked *FFT processor* which accesses memory (not shown in the figure) via data address of 7-b wide for 128-point complex FFT.

The result is shown in Table II with the first set of patterns named `parser` and the second set of patterns named `fft`. The result with BI coding is omitted because BI coding has little effect for these examples. However, the reduction with PBI coding and MPBI coding is still substantial. Furthermore, the number of bus lines selected for PBI coding is very small meaning that the overhead due to coding logic is kept small.

Authorized licensed use limited to: Seoul National University. Downloaded on August 24, 2009 at 01:55 from IEEE Xplore. Restrictions apply.

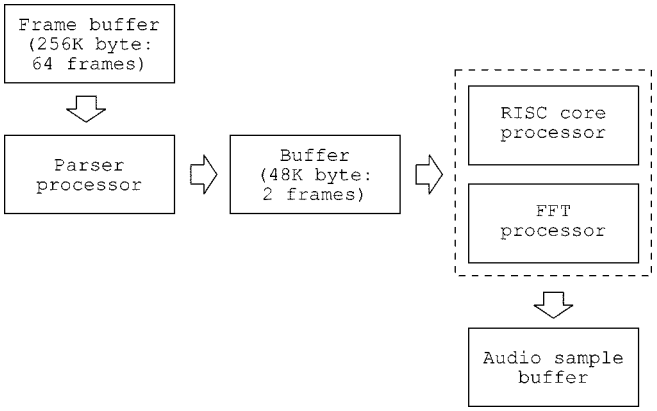

Fig. 6. Block diagram of audio decoder.

TABLE II
RESULT FOR EXAMPLES FROM AUDIO DECODER

Applications		Unenc.	PBI		MPBI	Beach	Unenc.	PBI	
Name	n	T_{bus}	Red.	$\|\mathcal{S}\|$	Red.	Red.	T_{eff}	Red.	$\|\mathcal{S}\|$
parser	16	2563	74.8	7	76.4	75.0	2563	71.1	7
fft	7	1036	21.6	2	27.6	0.9	1036	19.9	2

C. PBI Coding for Data Bus

PBI coding is suitable for data address buses of application-specific systems. However, it can be applied for data buses of some application-specific systems even though the sequence of patterns is not fixed but its statistics such as dynamic range is available. This is because PBI coding relies on correlation of bus lines but not on patterns themselves. For a data bus employing two's complement representation, the least significant (LS) bits tend to be random whereas the most significant (MS) bits are far from random. There is also an intermediate region separating the regions of the LS and MS bits [12]. These characteristics of a data bus fit very well with the optimization problem in PBI coding. Therefore, we can apply PBI coding for the data bus by using a given set of typical data patterns. With the same argument, MPBI coding can also be applied to the data bus.

We experiment with three example patterns: 32-b wide output speech signal from a noise canceller [13], 8-b wide data patterns between *Parser processor* and *Buffer* of the audio decoder, and 40-b wide data patterns between memory and a 128-point complex *FFT processor* of the audio decoder. For each example, one set of patterns is used to select the subbus ($\mathcal{S}$) for PBI coding and to partition a bus into multiple subbuses for MPBI coding. With fixed configuration for each coding scheme, another set of patterns is encoded by each coding scheme. The results are shown together with those of BI coding in Table III with examples named `speech`, `parser`, and `fft`, respectively.

VI. CONCLUSION

This paper proposes PBI coding scheme, which is quite efficient for data address buses of application-specific systems though the scheme is general enough to be used in other types of buses such as data buses. In the proposed scheme, we minimize the number of bus lines involved in bus encoding as well as the number of total bus transitions. We present a heuristic algorithm of selecting a subgroup of bus lines such that bus transitions are minimized by encoding only those bus lines. MPBI coding scheme is also proposed to better exploit correlation among bus lines. We present a heuristic to partition a bus into multiple subbuses to be used in MPBI coding. Experimental results show that reductions in the number of bus transitions with both PBI and MPBI coding are substantial for benchmark examples and a large example such as an audio decoder. The performance of the proposed subbus selection algorithm for PBI coding is almost as good as that of simulated annealing in bus transition reduction.

TABLE III
COMPARISON OF THE TOTAL BUS TRANSITIONS FOR PATTERNS AT DATA BUSES

Applications		Unenc.	BI	PBI		MPBI
Name	n	T_{bus}	Red.	Red.	$\|\mathcal{S}\|$	Red.
speech	32	107989	28.9	43.6	18	56.2
parser	8	6506	12.1	12.8	6	22.1
fft	40	13554	27.1	27.8	34	54.9

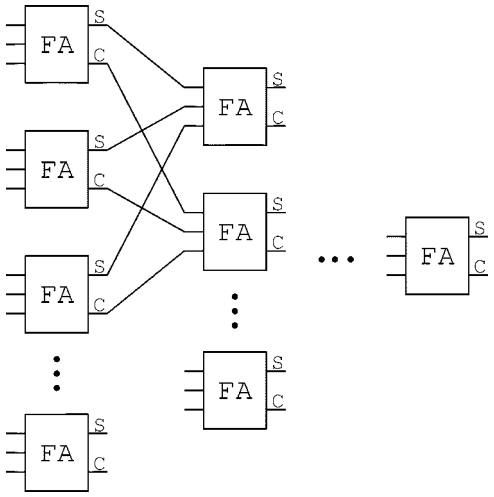

Fig. 7. Block diagram of a majority voter circuit implemented with a tree of full adders.

APPENDIX A

Here we derive the approximate number $N(x)$ of full adders (FAs) used in a majority voter circuit with x inputs. The majority voter can be implemented as a tree of FAs as shown in Fig. 7. The first level (the leftmost column in the figure) consists of $x/3$ FAs[3] which deliver $(x/3)2$ inputs to the second level. Then the second level consists of $(x/3)(2/3)$ FAs. It follows then that

$$N(x) = \frac{x}{3} + \frac{x}{3}\left(\frac{2}{3}\right) + \frac{x}{3}\left(\frac{2}{3}\right)^2 + \cdots + \frac{x}{3}\left(\frac{2}{3}\right)^{k-1} = x\left[1 - \left(\frac{2}{3}\right)^k\right] \tag{8}$$

where k is the height of the tree. Because the last level consists of a single FA, we require

$$\frac{x}{3}\left(\frac{2}{3}\right)^{k-1} = 1 \tag{9}$$

[3]When x is not divisible by 3, we can use a simplified logic for $(x/3) - \lfloor x/3 \rfloor$ inputs rather than using a FA. Hence, we maintain a fractional value for the number of FAs.

Authorized licensed use limited to: Seoul National University. Downloaded on August 24, 2009 at 01:55 from IEEE Xplore. Restrictions apply.

provided that we maintain a fractional value for k. Solving for k and substitute it in (8)

$$N(x) = x\left[1 - \left(\frac{2}{3}\right)^{1+\log_{3/2} x/3}\right] = x - 2. \quad (10)$$

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] 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.

[3] 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.

[4] 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.

[5] 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 Symp. VLSI*, Mar. 1997, pp. 77–82.

[6] E. Musoll, T. Lang, and J. Cortadella, "Exploiting the locality of memory references to reduce the address bus energy," in *Proc. Int. Symp. Low Power Electronics Design*, Aug. 1997, pp. 202–207.

[7] 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. Symp. Low Power Electronics Design*, Aug. 1997, pp. 24–29.

[8] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd ed. New York: Cambridge Univ. Press, 1992.

[9] P. Panda and N. Dutt, "1995 high level synthesis design repository," in *Proc. Int. Symp. System Synthesis*, 1995.

[10] S. Kirkpatrick Jr., C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

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

[12] P. Landman and J. Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 173–187, June 1995.

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

Architecture Driven Circuit Partitioning

Chau-Shen Chen, Ting Ting Hwang, and C. L. Liu

***Abstract*—In this paper, we propose an architecture driven partitioning algorithm for netlists with multiterminal nets. Our target architecture is a multifield-programmable gate array (FPGA) emulation system with folded-Clos network for board routing. Our goal is to minimize the number of FPGA chips used and maximize routability. To that end, we introduce a new cost function: the average number of pseudoterminals per net in a multiway cut. Experimental result shows that our algorithm is very effective in terms of the number of chips used and routability as compared to other methods.**

***Index Terms*—Field-programmable gate array (FPGA), folded-Clos network interconnection architecture, multi-FPGA emulation system, partitioning.**

I. INTRODUCTION

There is an ever-increasing interest in utilizing field-programmable gate array (FPGA)-based computing engines as high-speed, reconfigurable prototyping and emulation systems. An FPGA-based computing engine consists of multiple FPGAs which are interconnected through a certain interconnection architecture. Studies in interconnection architecture [2], [4], [1] are abundant in the literature in which various interconnect architecture were proposed.

Interconnection architecture can be classified as two distinct types. In the first type, FPGAs are connected through a certain fixed routing wire, e.g., the interconnection architecture in Splash I [1], and so on. In the second type, FPGAs are connected through interconnection chips, e.g., the interconnection architecture in Realizer [2], and so on. The latter interconnection architecture has the advantage, vis-a-vis the former, of higher FPGA logic utilization and delay uniformity [2]. Interconnection architectures using routing chips include one-full-crossbar interconnection architecture as was proposed in [3] and folded-Clos network interconnection architecture as was proposed in BORG [5] and Realizer [2]. The one-full-crossbar interconnection architecture is superior to the folded-Clos network interconnection architecture in terms of routability. However, its size grows as the square of the total pin-count which is not practical for a large number of FPGAs. The folded-Clos network interconnection architecture [2], [4] has bounded interconnect delay, scales linearly with pin-count and allows hierarchical expansion. However, it requires: 1) the design of an effective partitioning algorith and 2) the design of an effective board routing algorithm. In this paper, we will study a new partitioning algorithm.

Most of the previous partitioning algorithms did not take the routing architecture into consideration. Consequently, although there are partitioning algorithms that minimize the number of chips used [7], [8] or minimize the number of cut-nets [6], [9], the partitioning result produced may not be routable in the folded-Clos interconnect architecture. To take the routability into consideration, we propose a new cost function: the average number of pseudoterminals per net in a multiway cut which turns out to be a good indicator of the routability of a partitioning result. We then design an iterative improvement partitioning algorithm that will reduce the average number of pseudoterminals per net. In our

Manuscript received July 7, 1998; revised July 13, 2000. This work was supported in part by a grant from the National Science Council of R.O.C. under Contract NSC-89-2218-E-007-062.

The authors are with the Department of Computer Science, National Tsing Hua University, HsinChu, 30043, Taiwan (e-mail: tingting@cs.nthu.edu.tw).

Publisher Item Identifier S 1063-8210(01)00701-6.

1063–8210/01$10.00 © 2001 IEEE

Authorized licensed use limited to: Seoul National University. Downloaded on August 24, 2009 at 01:55 from IEEE Xplore. Restrictions apply.