

Crosstalk-Aware Energy Efficient Encoding for Instruction Bus through Code Compression

Balaji Vaidyanathan, Yuan Xie
Department of Computer Science and Engineering
Pennsylvania State University, PA - 16801
{bvaidyan, yuanxie}@cse.psu.edu

ABSTRACT

Code compression techniques have been proposed to mitigate the problem of limited memory resources in embedded systems. As technology scales, reducing on-chip bus energy consumption is becoming important for embedded system designers. In this paper, we propose a crosstalk-aware energy-efficient code compression scheme, which can reduce inter-wire coupling transition induced instruction bus energy consumption, without sacrificing compression ratio. The experimental results show that the bus power consumption due to inter-wire coupling transition alone is reduced by 42-68% and the total bus power consumption is reduced by 55-71% for TMS320C6x benchmarks.

I. INTRODUCTION

Code size reduction is one of the most important design goals for embedded systems, because embedded systems are space and cost sensitive and memory is one of the most restricted resources. As technology scales, on-chip bus power consumption becomes an important part of the overall embedded system power consumption. Lahiri et al. [7] demonstrated that the on-chip bus can consume significant power that is comparable to other well-known primary sources of power consumption (such as embedded processor and caches). Another impact of technology scaling is the increasing importance of bus crosstalk. The inter-wire capacitance becomes dominant compared to the wire-to-substrate capacitance (self capacitance) [3], and the power consumption due to crosstalk induced coupling capacitance can dominate the power consumption due to the self capacitance.

Even though the primary goal of code compression techniques is to reduce the size of the instruction memory such that the system cost is reduced, it is essentially a re-encoding of the instruction bus and can be used as a scheme to reduce the instruction bus power consumption and crosstalk. In this paper, we demonstrate a scheme that can efficiently reduce bus power consumption and crosstalk without sacrificing the compression ratio, by using a Variable-to-Fixed (V2F) code compression technique.

The rest of the paper is organized as follows: Section II reviews related work; Section III gives a brief introduction to code compression algorithm we are using; Section IV describe the scheme to reduce the inter-wire coupling transition and hence the total bus power consumption via the codeword assignment for code compression algorithm; Section V shows the experimental result and Section VI concludes the paper.

II. RELATED WORK

There have been various approaches to code compression, including compiler techniques [8], ISA modification (which

modifies or customizes the original instruction set architecture), as well as using existing data compression algorithms [1][9].

Many techniques have been proposed to reduce bus energy by exploiting the data and address access patterns. For example, The Bus-Invert Code [10] toggles the polarity of the signals according to the Hamming distance (the number of differing bits) between two consecutive data values; The T0 code [11] uses an extra line to indicate whether the bus is in normal mode or increasing address. However, many techniques to reduce address bus toggling can not be applied to the instruction bus, since the bits transferred on the instruction bus are highly irregular. Therefore, very few research efforts have been applied to reduce the instruction bus energy consumption. Recently Petrov et al. [5] proposed a low power encoding framework for embedded processor instruction buses, using efficient instruction transformation so as to minimize the bit transitions on the instruction bus lines.

Various spatial and temporal redundant bus encoding schemes to reduce bus crosstalk incur extra performance or area penalties. For example, Victor et al. [4] have proposed self-shielding codes which eliminate the worst case crosstalk. Khan et al. [3] described a coupled crosstalk and power optimization methodology for eliminating worst crosstalk and reducing power using spatially redundant LWC code. However, the bus architecture is changed to accommodate redundant signals which certainly affect bus performance, area, and power.

The contribution of this paper is to propose an encoding methodology to code compression scheme, such that the crosstalk induced power consumption on the instruction bus is reduced. We demonstrate that, by clever encoding, instruction bus power reduction can be achieved as a by-product of code compression, incurring no extra delay, power, or area overhead compared to the *original* code compression mechanism. To the best of our knowledge, this is the first crosstalk aware energy efficient encoding scheme for instruction bus through code compression.

III. CODE COMPRESSION ALGORITHM

The code compression algorithm we used belongs to Variable-to-Fixed (V2F) coding [1]. It is different from Huffman coding or other fixed-to-variable coding in that it maps variable-length bit sequences into fixed-length bit sequences. In other words, all codewords are of equal length. This characteristic will be exploited in Section 4 to help reducing bus crosstalk and power consumption.

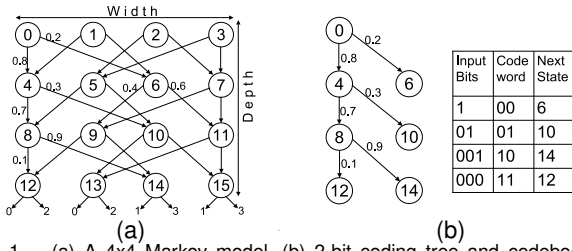


Fig. 1. (a) A 4x4 Markov model. (b) 2-bit coding tree and codebook for state 0 [1].

In V2F code compression, a Markov probability model is used to exploit the statistical dependence in the instruction bit stream. A Markov model consists of a number of states, where each state is connected to two other states: the left (right) edge with a probability P implies that the next input bit of the instruction stream has a probability of P to be zero (one). The steps to compress instructions using a Markov model and variable to fixed coding can be described as a two-pass scanning of the binary program code:

1. *The first pass is to collect statistics and construct codebook.* Starting from initial state 0, if there is a zero (one) in the binary instruction stream, a left (right) transition is taken, and the next state is reached. By going through the whole program code, the probability for each transition is gathered. For example, a 4X4 Markov model is shown in Figure 1a (in which part of the transition probability numbers are shown). After the Markov model is constructed, a N -bit coding tree and the codebook for each state are generated using the method described in [1]. For example, a 2-bit coding tree and the codebook for Markov state 0 is shown in Figure 1b. Each Markov state has its own codebook. Therefore, for an M -state Markov model using a N -bit codeword, there are totally M codebooks with each codebook has $2N$ entries (i.e., $2N$ codewords). Each codeword can be represented as $[C_i, W_i]$, in which C_i ($C_i = 1, 2, \dots, M$) is one of the codebooks and W_i ($W_i = 1, 2, \dots, 2N$) is the index for each entry in the codebook. For example, in Figure 1b, each entry in codebook 0 for state 0 can be represented as: $[0,1]=00$, $[0,2]=01$, $[0,3]=10$, $[0,4]=11$.

2. *The second pass of scanning the program code does the actual compression.* The instructions are compressed block by block. Starting from the initial state, the code book for each state is used to perform a table look-up to find a match for the input bit stream. Each code book entry contains a matching bit sequence and its corresponding codeword output, as well as the next state. For example, in Figure 1b, when the input bit sequence is 001, a codeword 10 will be produced, and the next state is 14, hence the compression procedure starts to use the code book for state 14 and scan the following bit streams.

IV. CROSSTALK AND POWER REDUCTION

After compression, the code transmitted on the instruction bus contains a series of N -bit codeword ($[C_i, W_i]$) as shown in Figure 2b. For the variable-to-fixed coding scheme described in Section III, the N -bit binary assignment to the codewords in a codebook can be arbitrary and it doesn't affect the compression efficiency, since the codeword length is fixed. The

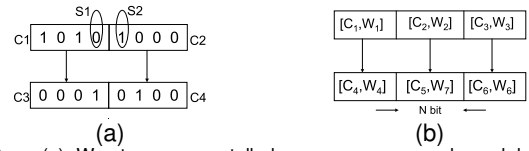


Fig. 2. (a) Worst-case crosstalk happens across codeword boundary, (b) Instruction bus transition

only restriction is that all codewords in the same codebook should be distinct, i.e., for $[C_i, W_i]$ and $[C_k, W_k]$, if $C_i = C_k$, and $W_i \neq W_k$, then $[C_i, W_i]$ should be different from $[C_k, W_k]$. In this section, we will describe how to take advantage of this property to reduce the crosstalk and power consumption on instruction bus. Based on the model given in [12], for an N -bit bus, the total capacitance on a victim wire at bit position k is given by equation 1.

$$C_k(d_t, d_{t+1}) = \begin{cases} C_s((1+\lambda)\Delta_1 - \lambda\Delta_2)d_{t+1} & k = 1 \\ C_s((1+\lambda)\Delta_n - \lambda\Delta_{n-1})d_{t+1} & k = n \\ C_s((1+2\lambda)\Delta_k - \lambda(\Delta_{k-1} + \Delta_{k+1}))d_{t+1} & 1 < k < n \end{cases} \quad (1)$$

Where C_k is the total capacitance (self and inter-wire coupling) seen by the victim wire at bit position k of an n bit bus due to the data transition from d_t to d_{t+1} . C_s is self capacitance, $\lambda = C_c/C_s$ where C_c is the inter-wire coupling capacitance, d_{t_x} is the polarity of the data d_t at bit position x and $\Delta_x = d_{t+1_x} - d_{t_x}$. Note that λ increases as technology scales and it is estimated to be 1.742 for 250nm technology and 9.82 for 70nm using interconnect parameters specified in [13] with dielectric constant of 2.5 and calculated using PTM for interconnect [14].

N_c -Number of inter-wire coupling transitions on the middle wire
Transition symbols are defined as (\uparrow : $0 \rightarrow 1$), (\downarrow : $1 \rightarrow 0$), ($-$: $1 \rightarrow 1$), ($-$: $0 \rightarrow 0$), ($-$: -1 or -0), ($*$: any transition)

N_c	Transition pattern
-2	($\uparrow, -1, \uparrow$)
-1	($\uparrow, -1, -$), ($-$, $-1, \uparrow$)
0	($-$, $-$, $-$), ($\uparrow, -$, \downarrow), ($\downarrow, -1, \uparrow$), ($*$, $\uparrow, *$), ($\uparrow, -0, -$), ($-$, $-0, \downarrow$), ($\downarrow, -0, \downarrow$), ($\uparrow, -0, \uparrow$), ($-$, $-0, \uparrow$), ($\uparrow, -0, -$), ($\uparrow, \uparrow, \uparrow$)
1	($\downarrow, -1, -$), ($-$, $-1, \downarrow$), ($-$, \uparrow, \uparrow), ($\uparrow, \uparrow, -$)
2	($\downarrow, -1, \downarrow$), ($-$, $\uparrow, -$), ($\downarrow, \uparrow, \uparrow$), ($\uparrow, \uparrow, \downarrow$)
3	($\downarrow, \uparrow, -$), ($-$, \uparrow, \downarrow)
4	($\downarrow, \uparrow, \downarrow$)

TABLE I

INTER-WIRE COUPLING TRANSITIONS ON VICTIM LINE.

Equation 1 can be simplified as $C_k = C_s * (N_s + \lambda * N_c)$, in which $N_c(N_s)$ is the number of inter-wire coupling (self) transitions. Table I shows the corresponding number of inter-wire coupling transitions. It shows that there are seven different classes of inter-wire coupling transition values based on the transition patterns.

By taking advantage of the freedom to do codeword assignment in the code compression, we can develop heuristics to find the codeword assignment that reduces the overall power consumption. For example, for a 4-bit codeword assignment, if a pair of codewords transits to each other a lot, codeword assignment 0000 and 1111 is a better choice than the assignment of 0000 and 1010, since the first assignment corresponds to four transitions while the second one leads to eight transition as shown in table 2 (in which $\lambda = 3$).

The example above only takes into consideration the bit switching within a codeword boundary. Since codewords are

packed and transmitted on the instruction bus, the crosstalk effect across the codeword boundary has to be taken care of. For example, in Figure 2a maximum number of inter-wire coupling transitions of four happens at bit position S1 due to opposing transition in S2 and is not desirable. Therefore, it is necessary to take into account both the codeword transition on the bus and the codeword adjacency relationship when assigning appropriate codewords.

By going through the whole compressed program, we can build two graphs, namely the vertical graph and horizontal graph (V-graph and H-graph). The V-graph contains vertically adjacent codewords. Vertical adjacency means that the two compressed codeword occur one after the other in subsequent cycle of bus transmission. Each edge between two nodes has a weight E_i associated with it, indicating how many times the bus transition happens between these two codewords. For example, in Figure 2b, a transition between codeword [C1, W1] and [C4, W4] happens and the weight on the edge between these two codewords increases by 1. The H-graph contains the horizontally adjacent codewords. Horizontal adjacency means that the two compressed codeword occur adjacent to each other in the same cycle of bus transmission. For example, in Figure 2b, codeword [C1, W1] is to the left of codeword [C2, W2], which increases the weight for the edge from [C1, W1] to [C2, W2] in H-graph. Figure 3a and 3b gives a snapshot of V-graph and H-graph respectively.

For an M-state N-bit code compression algorithm, the possible number of codeword assignment is $(2^N!)^M$ (because there are M codebooks with each codebook has 2^N entries, and the possible codeword assignment for each codebook is $2^N!$), which is a huge search space and it is infeasible to exhaustively try all possibility and find the optimal one for minimal crosstalk and power. Our goal is to efficiently find a good codeword assignment such that the overall crosstalk and power consumption on the bus is reduced. The greedy heuristic algorithm we developed is described as follows:

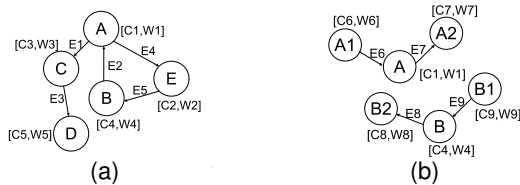


Fig. 3. (a) Vertical graph showing the vertical adjacency. (b) Horizontal graph showing horizontal adjacency.

Greedy algorithm

- 1) Sort all the edges in V-graph in decreasing order of their weights.
- 2) Pick a pair of nodes (A and B) which are connected by a vertical edge in the sorted order. Assign A and B (if not already assigned) with codewords that minimizes bus energy.
- 3) Then pick the most adjacent node to A on both sides of them. Here, most adjacent indicates the node that is connected to A through a horizontal edge of highest weight.
- 4) Assume that A1 and A2 are the most adjacent ones to the left and right of A respectively. Then assign last(first) bit of A1(A2) to be equal to first(last) bit of A.
- 5) Repeat the steps 2-4 till all the nodes are assigned.

In step 2 of the above algorithm, we choose node A and node B for assignment. We assign node A to some

codeword, say $(0000)_2$. Now the node A codeword becomes the head. The codeword assignment for Node B is done using a codeword selection methodology. This methodology is explained using Table II for the head value of node A which is equal to $(0000)_2$. Table II shows the total transition (N_t) incurred in transmitting any of the 16 possible code word after transmitting the Head $(0000)_2$. The total transition is calculated based on the expression given in [3] as follows.

$$N_t = \sum_{k=1}^n N_k(d_t, d_{t+1}) \quad (2)$$

Where N_t is total transition seen by the n-bit bus, N_k is total transition at bit position k and is equal to $(N_s + \lambda * N_c)$, d_t and d_{t+1} are current data and next data transmitted on a n-bit bus. Note that N_s and N_c are calculated using equation 1 and are used in calculating N_k here.

(Note: $\lambda = 3$)

Head=0000			
Code Word	Total Transition(N_t)	Code Word	Total Transition(N_t)
0000	$0 + 0\lambda = 0$	0010	$1 + 2\lambda = 7$
0001	$1 + 1\lambda = 4$	0100	$1 + 2\lambda = 7$
1000	$1 + 1\lambda = 4$	0110	$2 + 2\lambda = 8$
1111	$4 + 0\lambda = 4$	1001	$2 + 2\lambda = 8$
0011	$2 + 1\lambda = 5$	1011	$3 + 2\lambda = 9$
1100	$2 + 1\lambda = 5$	1101	$3 + 2\lambda = 9$
0111	$3 + 1\lambda = 6$	0101	$2 + 3\lambda = 11$
1110	$3 + 1\lambda = 6$	1010	$2 + 3\lambda = 11$

TABLE II

CODWORD SELECTION TABLE FOR 0000

The dynamic bus power calculation is based on the equation 3 from [2].

$$P_{dyn} = 0.5 \times C_s \times N_t \times V_{dd}^2 \times f \quad (3)$$

Where N_t is calculated from equation 2, V_{dd} is the supply voltage and f is the operating clock frequency. The codeword column is sorted based on the total capacitance N_t seen by a 4-bit bus by transmitting one of the 16 possible code words after transmitting the head. The algorithm tries to pick the codeword for B with the topmost value on the codeword column of the Table II. Obviously, if A and B belong to the same codebook, the first codeword in the codebook is not available for assignment, since codewords in the same codebook must be distinct. If the codeword has been assigned to other entries in the same codebook, the next available codeword in the sorted table will be picked.

Step 2 in the algorithm takes care of the reduction of inter-wire capacitive coupling between two vertical transitions within a codeword. Steps 3 and 4 take care of reducing the worst case inter-wire capacitive coupling that occurs across codeword boundary as shown in Figure 2a. The transition shown in Figure will lead to worst case inter-wire coupling capacitance in signals S1 and S2. As explained in step 4 we assign the boundary bits (S1 and S2 in Figure 2a) to be the same, which prevents the worst case inter-wire coupling capacitance in the boundary bits.

V. EXPERIMENTS AND RESULTS

In this section, we demonstrate the experimental results by using TMS320C6x (Texas Instruments DSP VLIW processor) applications. Benchmark applications are provided by Texas Instrument and compiled using the Code Composer Studio

IDE. It has been proved in [1] that by using a 4-bit codeword, the best compression ratio can be achieved, so we only show experiments using a 4-bit codeword. To construct the V-graph and H-graph, we use a cycle accurate simulator for TMS320C6x and profile the benchmark applications to capture accurate bus transition activities.

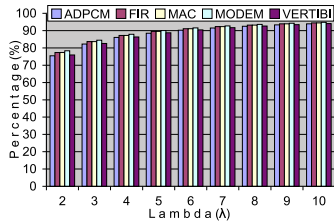


Fig. 4. Percentage contribution of inter-wire coupling transition to total bus power

The contribution of the inter-wire coupling transitions ($\lambda * N_c$) to the total power for un-compressed code transmitted on the bus is predominant compared to the self capacitance as shown in Figure 4 and this contribution increases as λ increases in future technologies. This justifies the importance of bus power optimization which takes inter-wire coupling transitions into account. We have used λ values ranging from two to ten.

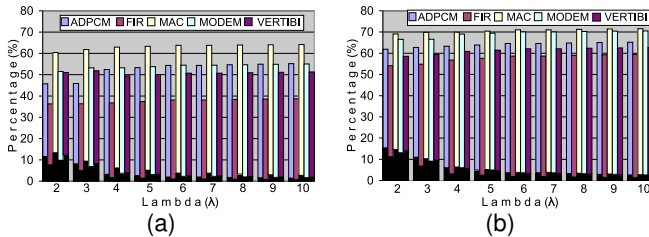


Fig. 5. Total bus power reduction (percentage) using (a)8x32, (b)32x32 Markov model

Figure 5a and 5b shows the decrease in total transitions for our optimized method applied to 8x32 and 32x32 Markov models respectively, compared to the uncompressed code. We observe a power reduction of 35-63% for 8x32 Markov model compared to 55-71% reduction for 32x32 Markov model. In general larger Markov model extracts more information about the code word transition patterns and hence code assignment is more effective in reducing the total power. The black portion shows the power reduction due to decrease in self capacitance and the rest due to decrease in inter-wire coupling transitions. The power reduction due to inter-wire capacitive coupling is 42-68% for 32x32 Markov model and 30-60% for 8x32 Markov model.

The importance of the proposed bus power optimization is further explained by comparing the optimized bus power with random and worst case code assignment in Figure 6a and 6b normalized over the optimized case. The experimental result shows that inappropriate codeword assignment results in enormously high bus power consumption compared to optimized code. The random case increases the bus power to 225% and the worst case to 670%.

Our experimental results show that our encoding scheme is efficient for a wide range of λ values and its efficiency increases as λ increases as visible from improvement in power reduction in Figure 5a and 5b. This is due to the fact

that we optimize inter-wire coupling transitions induced power, which becomes a major portion of total bus power for larger λ in future technologies.

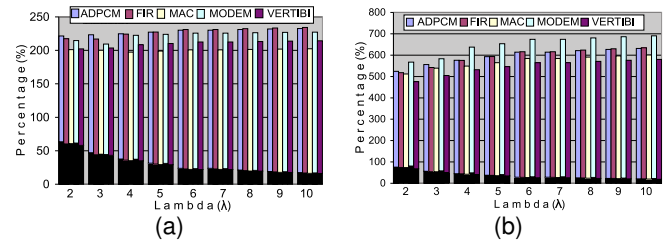


Fig. 6. (a) Random case. (b) Worst case bus dynamic power increase for TMS using 32x32 Markov model

The average compression ratio using a 4-bit codeword are 72% and 68% for 8x32 and 32x32 Markov model, respectively. With intelligent code word assignment we have achieved bus power reduction (considering the power due to inter-wire coupling transitions).

VI. CONCLUSION

In this paper, we propose an encoding methodology to code compression scheme considering inter-wire coupling transitions, such that power consumption on the instruction bus is reduced. Compared to the original code compression scheme, we show that a clever encoding scheme can achieve bus power reduction (including both self and inter-wire coupling transitions) as a by-product of code compression and incurs no extra delay, power, or area overhead compared to the original code compression mechanism [1]. Experimental results show that, after applying our encoding technique using a variable to fixed code compression scheme, up to a 71% reduction in bus power is achieved.

REFERENCES

- [1] Y. Xie, W. Wolf, H. Lekatsas, "Code compression for VLIW processors using variable-to-fixed coding," In *Proc. of ISSS*, pp.138-143, 2002
- [2] M. Cha, C. Lyuh, T. Kim, "Resource-constrained low-power bus encoding with crosstalk delay elimination," In *Proc. ASPDAC*, pp.834-837, 2004
- [3] Z. Khan, A. T. Erdogan, T. Arslan, "Dual low-power and crosstalk immune encoding scheme for on-chip data buses," In *IEEE Electronic Letters*, vol 39, no.20, pp.1436-1437, Oct 2003
- [4] B. Victor, K. Keutzer, "Bus Encoding to Prevent Crosstalk Delay," In *Proc. of ICCAD*, pp.57-63, 2001
- [5] P. Petrov, A. Orailoglu, "Low-Power Instruction Bus Encoding for Embedded Processors," In *IEEE Trans. on VLSI Systems*, pp.812-826, Aug 2004
- [6] H. Lekatsas, J. Henkel, W. Wolf, "Code Compression for Low Power Embedded System Design," In *Proc. of DAC*, pp.294-299, 2000
- [7] K. Lahiri, A. Raghunathan, "Power analysis of system-level on-chip communication architectures," In *Proc. of CODES+ISSS*, pp.236-241, 2004
- [8] S. K. Debray, W. Evans, R. Muth, B. D. Sutter, "Compiler techniques for code compaction," In *Proc. of ACM TOPLAS*, 22(2):378-415,2000
- [9] C. Lefurgy, E. Piccininni, T. Mudge, "Analysis of a High Performance Code Compression Method," In *Proc. of Micro*, pp.93-102, 1999.
- [10] M. R. Stan, W. P. Burtleson, "Bus-Invert Coding for Low Power I/O," In *IEEE Trans. on VLSI Systems*, vol.3, issue.1, pp.49-58, March 1995
- [11] L. Benini, G. Micheli, E. Macii, D. Sciuto, C. Silvano, "Address bus encoding techniques for system-level power optimization," In *Proc. of EuroDAC*, pp.861-867,1997
- [12] P. P. Sotiriadis, A. Chandrakasan, "Low Power Bus Encoding Techniques Considering Inter-Wire Capacitances," In *Proc. of IEEE CICC*, pp.507-510, 2000
- [13] D. Sylvester, C. Hu, O. S. Nakagawa, S.-Y. Oh, "Interconnect Scaling: Signal Integrity and Performance in Future High-Speed CMOS Designs," In *IEEE symposium on VLSI Technology Digest of Technical Papers*, pp.42-43, 1998
- [14] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, C. Hu "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," In *Proc. of IEEE CICC*, pp.201-204, 2000