Low Overhead Encodings For Reduced Activity In Data And Address Buses

Paulo J. Ramos
IST-INESC
Rua Alves Redol, 9
1000 Lisboa, Portugal
pjjr@algos.inesc.pt

Arlindo L. Oliveira
IST-INESC/CEL
Rua Alves Redol, 9
1000 Lisboa, Portugal
aml@inesc.pt

This work describes and evaluates two bus encoding methods that aim at reducing the number of transitions observed in data and address buses. Unlike existing proposals, these methods have the characteristic that they require no extra bus lines and only a small amount of encoding logic.

The experimental results, obtained in traces from the SPEC95 benchmark, show that transition coding is effective in reducing the number of transitions in data buses, while bit prediction methods can be applied with success to address buses.

I. Introduction

In complex VLSI systems, the power dissipated in buses represents a large fraction of the total power. This is due, on one hand, to the high capacitance of bus lines, and, on the other hand, to the increasing larger bandwidth required in high performance systems.

Due to the large difference in intrinsic capacitance exhibited by external buses lines and internal nodes, the dynamic power dissipated on bus lines can represent more than half of the total power dissipated in the system.

For these reasons, there have been several proposals for bus encoding techniques that aim at reducing activity on external, high capacitance lines, at the expense of some added circuit complexity.

Regrettably, the majority of the proposals made so far have not been extensively applied to actual designs, since they require the addition of extra bus lines, very complex additional circuitry and, in some cases, the execution of complex off line algorithms.

In this work, we examine two low overhead approaches to the problem. One approach is based on the application of transition coding, a method that has already been proposed by other researchers in conjunction with a transformation on the data words [SB94]. The proposal made here uses transition coding without applying any recoding of the transmitted data, a method that, intuitively, seems to have little advantage over the transmission of the plain codewords, but that, experimentally, reveals significant advantages.

The second approach is based on the use of bit prediction techniques, that try to predict the value of a data bit given its previous history. Although this approach implies a larger overhead in terms of circuit complexity, the extra circuitry added is very regular and does not involve the use of extra bus lines. This solution can be applied with good results to address buses.

We evaluate the applicability and power savings of the methods by analyzing the bus traffic that results from two different system configurations: the first a processor communicating with a unified cache and the second a processor communicating with two separate caches for data and instructions.

The results show that the performance of the methods varies very significantly with the system configuration and the bus to which the method is applied. In fact, the experiments performed lead us to believe that, under very general conditions, the use of transition coding leads to significant power savings in data buses, at the expense of very little extra circuit area, but is not applicable to address buses. On the other hand, bit prediction techniques are applicable to address buses, with significant power savings, but this is achieved with some area overhead.

II. Related work

A variety of bus encoding methods that aim at reducing bus line activities have been proposed in the literature. They are based either on the addition of redundancy to the bus codes, on the exploration of regularities in consecutive bus words or both.

The idea underlying the use of redundant codes is that, when more than one code is possible for a given bus word, the system may chose the code that leads to the most reduced activity in the bus lines.

In the bus invert method [SB95], an extra bus line is used to signal if the code on the bus is or is not complemented. The idea is that, if the previous pattern differs from the current one in more than $n/2$ bits, the complement of the data is sent, with the extra bit set at 1 to signal code inversion.

Using this encoding method in a system where 2^n different patterns are equally probable, a maximum of $n/2$ transitions per cycle is obtained, which compares well against the maximum of n transitions possible in an unencoded system.

Although the technique itself is quite simple, the computation of the Hamming distance between successive bus words requires the use of a non-trivial amount of hardware, that will, in practice, make the approach hard to apply in wide buses.

Another approach [SB94] is based on the idea of changing the way bus words are encoded as to reduce the probability of occurrence of a 1 in the bus line (P_1) to a value smaller than 0.5. This is obtained by using wider buses and choosing codes that have much less ones than zeros. Once the words

to be transmitted exhibit a smaller number of ones than zeros, transition coding can be used, resulting in reduced bus activity.

Encoding methods specific to the address bus make use of the locality of memory references. One obvious approach is to encode the address space using a code that exhibits a low number of transitions for consecutive addresses [STD94], but this code rapidly loses efficiency when the percentage of sequential accesses decreases.

In another method [EMC97], the address space is split into working zones. A reference to one address in one of these zones can be transmitted as a relative reference to the address of that working zone.

Although this technique may lead to large power savings, the actual implementation of this scheme in hardware is sufficiently complex to offset, in large measure, the gains obtained.

Other approaches proposed achieve significant power reduction by dividing the bus in groups and encoding each group independently of the others. One can even use different encoding schemes for different groups [BMM+98]. In this coding technique address bus traces are statistically analyzed, bits with high correlation are grouped and an encoding function is generated for each group. This method is trace dependent and requires the execution of a complex off line synthesis procedure. These facts, coupled with the considerable overhead imposed by the encoding circuitry makes it unlikely that it will be used in a large number of actual designs.

Techniques that aim at reducing the power dissipated in buses by dynamically reordering and complementing the words to be transmitted [MFO98] have also been proposed but their actual implementation in hardware is complex and the techniques are applicable only in very special circumstances.

III. Low overhead bus encodings

With the possible exception of the bus-invert method, all the approaches presented above are somewhat unlikely to be widely used in practice, since they require extensive changes in the system design and a large amount of intervention from the circuit and system designers.

Furthermore, they also impose a significant overhead, either in circuit area or on the number of bus lines, two factors that offset somewhat the possible gains and limit the applicability of the techniques to very specific cases.

The two techniques we propose and evaluate in this paper require very little additional hardware and no extra bus lines, achieving, nonetheless, a significant reduction of the activity level on the bus.

A. Transition coding

Transition coding is a well known encoding technique used in many communication systems. When using transition coding, there is a transition on the bus every time the data to be transmitted is a 1, and there is no transition on the bus if the data to be transmitted is a 0.

Transition coding can be implemented by the simple circuitry shown in figure 1, and requires no additional bus lines.

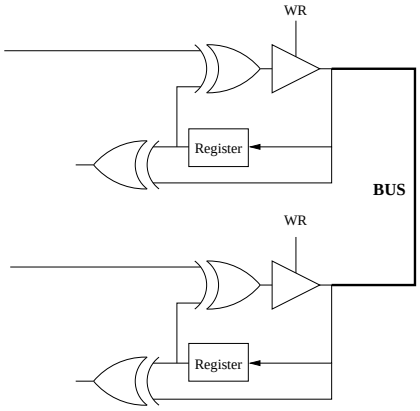

Fig. 1. Hardware needed to implement transition coding

Note that, apart from the bus drivers, that need to be present in any case, the implementation of transition coding requires only two extra gates and one register in each device attacking or reading the bus. In many cases, the register is already used, and only two extra gates are required to implement this type of coding.

It is somewhat counter-intuitive that transition coding leads to a reduction in the dissipated power when applied directly to buses, with no re-encoding of the data words.

To analyze why transition coding may be interesting, one has to perform a more detailed analysis of typical traffic in data buses. Let the probability of values 0 and 1 in the bus be given by P_0 and P_1, respectively. Assuming that the probabilities of consecutive bits in the same line are independent or each other, the probability of a transition under the usual level coding is given by equation 1.

$$P_{\text{level}} = P_0 P_1 + P_1 P_0 = 2P_1(1 - P_1) \tag{1}$$

On the other hand, using transition coding, the probability of a transition in a bus line is simply given by P_1:

$$P_{\text{trans}} = P_1 \tag{2}$$

Clearly, if $P_0 = P_1$, $P_{\text{level}} = P_{\text{trans}}$ and nothing is gained by using transition coding. This can be overcome if the words on the bus are re-encoded, with the objective of reducing the probability of the presence of a 1 [SB94].

We must note, however, that the above analysis is not applicable to real traffic on buses. On the first place, ones and zeros are usually not equally likely to appear. On second place, consecutive words are not independent, a factor that has a significant impact on the average number of bus transitions.

In the data buses, analysis of typical patterns in traces shows that the probabilities P_0 and P_1 are very different. In fact, a large amount of data includes small integer values used for iterations and array indexes, a factor that makes P_0 considerably larger than P_1.

However, this factor by itself is still not enough to make transition coding useful. In fact, consider the extreme condition where all the data represent small integers that can be encoded with $l << n$ bits, where n is the width of the bus. Although P_0 is much smaller than P_1, transition coding will still not save power, because, in the higher $n - l$ bits, no

transitions will take place. On the lower l bits, P_0 will be approximately equal to P_1, and therefore $P_{\text{level}} = P_{\text{trans}}$

The justification for the gains observed experimentally when transition coding is used resides in the fact that, in many cases, data words with a small number of ones alternate with data words that have a more uniform distribution of zeros and ones. This particular mix leads to considerable savings if transition coding is used, since this condition is equivalent to a low value of P_1 in the bus lines that correspond to the more significant bits.

B. Bit prediction

The basic idea underlying bit prediction is that some buses exhibit very regular bit patterns. For instance, address buses exhibit a very high percentage of addresses that are sequential, a factor that can be used to predict the value of the next data word with reasonably high accuracy.

Prediction of the value of a data bit can be performed by using a table with 2^k entries, indexed by the last k values of that bit in previous words. Clearly, this method is feasible only for small values of k. If the prediction is correct, no transition takes place on the bus. If the prediction is not correct, the transition on the bus signals that the value in the table is wrong, should be complemented and the table should be corrected. Schematically, the hardware required to implement this approach is shown in figure 2, where the table shown has 2^k entries, and the shift register has k one bit latches. Bit

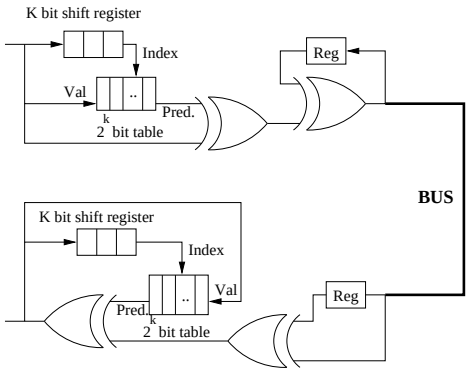

Fig. 2. Hardware needed to implement bit prediction coding

prediction has the significant advantage of not requiring extra bus lines, but it requires some additional hardware. The advantage is that this hardware does not impose any significant delays on the bus signals, and therefore has little impact on the bus speed.

Over other methods, bit prediction also exhibits the advantage of not requiring off line analysis and adaptation to a particular trace, an advantage that is significant in systems that are to be operated in a number of different conditions.

IV. Experimental results

A. Experimental setup

We used the SimpleScalar toolset [BA97] to evaluate the efficiency of the proposed approaches in a set of traces obtained from the execution of a number of programs in the SPEC95 benchmarks. The model used by the simulator is a modified version of the MIPS R3000 RISC processor. We considered two different configurations for the system.

On the first configuration, we evaluated traffic on the address and the data buses, assuming a unified cache for data and instructions. The second configuration uses separate data and instruction caches, as show in figure 3. For each exam-

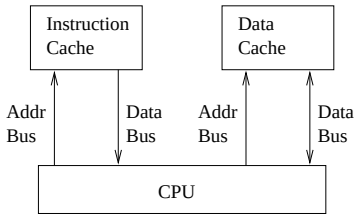

Fig. 3. Separate data and instruction caches configuration.

ple in the benchmark, distributed in binary format with the SimpleScalar simulator, we analyzed a trace of a typical execution. Statistics for the traces used are shown in table I. The second and third column show the number of data accesses and the number of instruction fetches executed, while the last column gives the total number of accesses, and, therefore, of bus transactions. For each configuration, we considered 3

TABLE I
STATISTICS FOR THE SPEC95 TRACES

Program	Memory Accesses		
	Data	Instr.	Total
applu	1088691	2864737	3953428
compress	804245	3006971	3811216
hydro2d	39760	105096	144856
m88ksim	142647	503602	646249
su2cor	703717	2410880	3114597
turbo3d	1232604	2792781	4025385
Wave5	1045241	2886462	3931703

possible encodings: transition coding, as described in section III-A, and bit prediction coding, with $k = 3$ and $k = 4$, as described in section III-B. For comparison, we also present the results for the bus invert method, considering the buses divided in sections of 16 bits, one of the most favorable configurations for this approach.

B. Results

In the first experiment, we considered all the bus transactions that correspond to traffic between the unified cache and the processor. Table II shows the results obtained for the data and address buses. The entries in the table report the total number of transitions observed, as a fraction of the reference value obtained using standard level coding. The second experiment used the dual cache configuration, shown in figure 3. In this configuration, there are two address buses and two data buses. Table III shows the results for the data cache, while table IV shows the results for the instruction cache.

C. Analysis of results

The results presented in the previous sections show that transition coding exhibits surprisingly good power savings compared with level coding, when applied to the data bus, in either the unified or dual cache configuration. The savings

TABLE II

NORMALIZED ACTIVITY IN THE UNIFIED CACHE BUSES

	Data Bus			
Program	Transition	Bit K=3	Bit K=4	Bus Inv.
applu	**.89**	.93	.97	.95
compress	**.71**	.88	.84	.83
hydro2d	**.71**	.87	.84	.88
m88ksim	**.74**	.89	.85	.86
su2cor	**.71**	.88	.82	.86
turbo3d	**.70**	1.00	.90	.95
Wave5	**.77**	1.02	1.03	.89
Average	**.75**	.92	.89	.89
	Address Bus			
Program	Transition	Bit K=3	Bit K=4	Bus Inv.
applu	1.93	.65	**.60**	.82
compress	1.36	.85	**.83**	.94
hydro2d	1.13	.74	**.73**	.76
m88ksim	1.44	.77	**.75**	.81
su2cor	1.38	.91	**.80**	.94
turbo3d	1.76	.96	**.94**	.95
Wave5	1.52	.96	1.03	**.70**
Average	1.50	.83	**.81**	.85

TABLE III

NORMALIZED ACTIVITY IN THE DATA CACHE BUSES

	Data Bus			
Program	Transition	Bit K=3	Bit K=4	Bus Inv.
applu	.98	1.12	1.07	**.91**
compress	.88	.92	.91	**.79**
hydro2d	**.83**	1.03	.96	**.83**
m88ksim	.85	.96	.92	**.74**
su2cor	.90	.95	.94	**.85**
turbo3d	**.79**	1.03	.99	**.79**
Wave5	**.88**	1.08	1.05	.89
Average	.87	1.01	.98	**.83**
	Address Bus			
Program	Transition	Bit K=3	Bit K=4	Bus Inv.
applu	3.40	1.03	**.75**	.91
compress	2.33	.97	**.84**	.90
hydro2d	2.17	1.18	.94	**.71**
m88ksim	2.56	1.12	1.05	**.71**
su2cor	1.55	.42	**.38**	.96
turbo3d	4.20	.87	**.50**	.98
Wave5	2.49	1.30	1.21	**.67**
Average	2.67	.98	**.81**	.83

are larger in the unified cache configuration, due, probably, to the mix of small integers and instruction codes that occurs in the single data bus. The results also show that transition coding is never effective when applied to the address bus, regardless of the configuration used.

On the other hand, significant savings can be obtained in the address bus if bit prediction codes are used. In fact, the bit prediction method with $k = 4$ always exhibits the best performance on the average, although not by a large margin in all the configurations. The savings are remarkable when the method is applied to the address bus of the instruction cache, achieving a power reduction of almost 50%.

The partial bus invert method yields significant savings in all configurations, although it exhibits the best performance in only one situation, the data bus in the dual cache configuration.

TABLE IV

NORMALIZED ACTIVITY IN THE INSTRUCTION CACHE BUSES

	Data Bus			
Program	Transition	Bit K=3	Bit K=4	Bus Inv.
applu	1.04	**.93**	.98	.96
compress	**.81**	.88	.85	.85
hydro2d	.94	1.02	.97	**.89**
m88ksim	.91	.98	.91	**.87**
su2cor	.89	.86	**.71**	.86
turbo3d	**.80**	.96	.87	.95
Wave5	**.87**	1.08	1.07	**.87**
Average	**.89**	.96	.91	**.89**
	Address Bus			
Program	Transition	Bit K=3	Bit K=4	Bus Inv.
applu	4.14	.53	**.29**	1.00
compress	2.52	.76	**.62**	.99
hydro2d	3.61	.84	**.70**	.99
m88ksim	4.15	.75	**.59**	.99
su2cor	3.51	.78	**.56**	1.00
turbo3d	3.65	.53	**.38**	1.00
Wave5	3.61	.74	**.68**	1.00
Average	3.60	.70	**.55**	1.00

V. CONCLUSIONS

We proposed two approaches for the problem of power reduction in system buses and evaluated their performance in a set of problems from the SPEC95 benchmark using two possible cache configurations. These approaches have the advantage of imposing very little circuit overhead and not requiring extra bus lines.

The results have shown that transition coding is effective when applied to the data bus, in both the unified and dual cache configurations, but is not effective when applied to the address bus. On the other hand, the bit prediction method is effective when applied to address buses, exhibiting a remarkable performance (from the point of view of power savings) when applied to the address bus of instruction caches. This method is, however, comparatively ineffective when applied to data buses.

REFERENCES

[BA97] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. Technical Report 1342, University of Wisconsin-Madison Computer Sciences Department, 1997.

[BMM+98] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano. System-level power optimization of special purpose applications: The Beach solution. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 24–29, 1998.

[EMC97] T. Lang E. Mussol and J. Cortadella. Exploiting the locality of memory references to reduce the address bus energy. In *Proceedings of the 1997 International Symposium on Low Power Electronics and Design*, pages 202–207, Montrey, CA, 1997.

[MFO98] R. Murgai, M. Fujita, and A. L. Oliveira. Using complementation and resequencing to minimize transitions. In *Proc. of the ACM/IEEE Design Automation Conference*, pages 694–697, Los Angeles, CA, June 1998. ACM Press.

[SB94] M. R. Stan and W. P. Burleson. Limited-weight codes for low-power i/o. In *Proceedings of the 1994 International Workshop on Low Power Design*, pages 209–214, 1994.

[SB95] M. R. Stan and W. P. Burleson. Bus-invert coding for low power i/o. *IEEE Transactions on VLSI Systems*, 1(1):49–58, 1995.

[STD94] C. L. Su, C. Y. Tsui, and A. M. Despain. Saving power in the control path of embedded processors. *IEEE Design and Test of Computers*, pages 24–30, 1994.