Extension of the Working-Zone-Encoding Method to Reduce the Energy on the Microprocessor Data Bus *

Tomás Lang
Dept. of ECE
Univ. of California aat Irvine
Irvine, CA 92697
tlang@uci.edu

Enric Musoll
Cyrix
(National Semiconductor)
Santa Clara, CA 95052
enric@cyrix.com

Jordi Cortadella
Dept. of Software
Univ. Politècnica de Catalunya
08071 Barcelona, Spain
jordic@lsi.upc.es

Abstract

The energy at the I/O pins is a significant part of the overall consumption of a chip. To reduce this energy, this work extends to the data bus the Working Zone Encoding method, originally applied to encoding an external address bus. This method is based on the conjecture that programs favor a few working zones of their address space at each instant and that addresses to consecutive accesses for each zone frequently differ by a small amount. When the difference is small, instead of sending the whole address, the method sends an offset with respect to the previous address to that zone, together with an identifier for the zone. In this paper the same idea is extended to the data bus.

The approach has been applied to several SPEC95 streams of references to memory along with the corresponding data values in a system with multiplexed address and multiplexed instruction/data buses. Moreover, the effect of instruction and data caches is evaluated. Comparisons are given with previous methods for bus encoding, showing significant improvement in all cases except for the multiplexed address bus with instruction cache, where the best scheme depends on the overhead of the implementation.

1. Introduction

The I/O energy is a substantial fraction of the total energy consumption of a microprocessor [1], because the capacitance associated with an external pin is between one hundred and one thousand times larger than that of an internal node. Consequently, the total energy consumption decreases by reducing the number of transitions on the high-capacitance, off-chip side, although this may come at the expense of some additional transitions on the low-capacitance, on-chip side.

For a microprocessor chip, the main I/O pins correspond to the address and data buses. We consider an encoding to reduce the activity in these buses based on the conjecture that applications favor a few working zones of their address space at each instant. Therefore, for an address to one of these zones, only the offset of this reference with respect to the previous reference to that zone is sent over the bus, along with an identifier of the current working zone. This is combined with a one-hot with transition-signaling encoding of the offset. The main contribution of the Working-Zone Encoding technique (WZE) is the tracking of the most recent working zones.

Previously, we proposed the WZE method for encoding the address bus [5, 6]. In this paper we also incorporate to this WZE method the data bus. We observe that the data for successive accesses to a working zone frequently differ by a small amount, so that it is effective to use also the one-hot with transition-signaling code for the data. Consequently, in this combined approach, when the address and the data are appropriate, we send:

- through dedicated wires, an identifier of the current working zone
- through the address bus, the offset of this reference with respect to the previous reference to that zone
- through the data bus, the offset of the current data value with respect to the value associated to the previous reference to that zone.

Several SPEC95 streams of references to memory along with the corresponding data values are used to evaluate the technique. Among the possible bus organizations (see Figure 1), in this paper we consider a multiplexed address bus (for instruction and data addresses) and a multiplexed instruction/data bus. We conclude that the encoding technique presented in this work significantly reduces the activity in both buses. Moreover, for the case without caches, the technique presented here outperforms other previous bus encoding proposals for low power, such as Gray, bus-invert, T0, combined

* This work was partially funded by CICYT grant TIC 95-0419

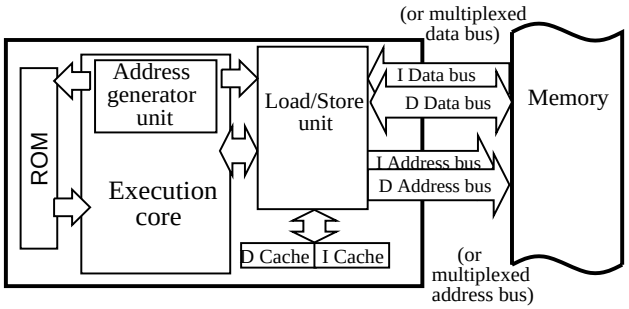

Figure 1. Types of buses in a general-purpose microprocessor.

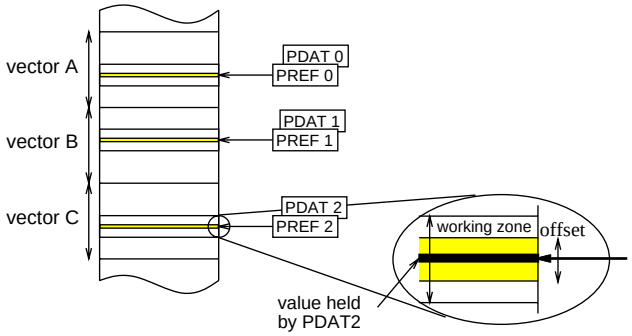

Figure 2. Address space with three vectors.

T0/bus-invert, inc-xor and dbm-vbm. On the other hand, for the multiplexed address bus with instruction cache the best scheme is either the WZE presented here or bus-invert with four groups, depending on the overhead of these two techniques. In any case, the WZE method requires fewer additional wires when coding both buses.

The paper is organized as follows: in Section 2 we review the previous work done in encoding a bus for reduced activity, with special insight in those techniques that target the data bus. In Section 3 we provide an overview of the WZE technique for the address bus as presented in [6] to give the reader a reasonable understanding of the method. The main contribution of this paper is presented in Section 4, where an extension of the WZE technique is presented that allows the data bus to be encoded by reusing a large portion of the hardware already used to encode the address bus. The results based on SPEC95 benchmarks are presented in Section 5. The conclusions of this paper are given in Section 6.

2. Previous work

The previously proposed encoding techniques for reduced bus activity may be classified as follows, depending on the degree of sequentiality of the values sent through the targeted bus:

- high degree: examples are the Gray [3], T0 [2], combined T0/bus-invert [2], and inc-xor [7] codes. These techniques are mainly applicable to the address bus without cache (and have been proposed for this case).
- low degree: examples are the bus-invert [8], and dbm-vbm [7] codes. These techniques have been proposed for the data bus; however they can also be applied to the address bus when caches are present.

Since we concentrate here on the data bus, we review the techniques of the second type [1]. The bus-invert method [8] consists on sending either the value itself or its bit-wise complement, depending on which would result in fewer transitions. An extra wire is used to carry this polarity information. For uniform and independent distributions, this encoding technique works better when the bit-width of the value to be sent is divided into smaller groups and each one encoded independently.

In [7] a source-coding framework is proposed as well as some specific codes. The scheme is based on obtaining a prediction function and a prediction error. This prediction error is XORed with the previous value sent to the bus so that the number of transitions is reduced in the likely case when the prediction error has a small number of ones. In the dbm-vbm technique, the prediction is the previous address and the prediction error is obtained by a function that increases with the absolute difference between the current input and the prediction. Then, code-words with fewer 1's are assigned to smaller error values.

3. Overview of the Working-Zone Encoding technique (WZE) for the address bus

This Section presents an overview of the proposed encoding technique for the address bus, as presented in [6]. In the next Section, we describe the extension of the WZE to encode the data bus.

The basis of the WZE technique for the address bus is as follows:

1. It takes into account the locality of the memory references: applications favor a few working zones of their address space at each instant. In such cases, a reference can be described by an identifier of the working zone and by an offset. This encoding is sent through the bus.
2. The offset can be specified with respect to the base address of the zone or to the previous reference to that zone. Since we want small offsets encoded in a one-hot code, the latter approach is the most convenient.

[1] We have reviewed the techniques of the first type in [6].

As a simple example consider an application that works with three vectors (A, B and C) as shown in Figure 2. Memory references are often interleaved among the three vectors and frequently close to the previous reference to the vector. Thus, if both the sender and the receiver had three registers (henceforth named *Prefs*) holding a pointer to each active working zone, the sender would only need to send:

- the offset of the current memory reference with respect to the Pref associated to the current working zone
- an identifier of the current Pref.

3. To reduce the number of transitions, the offset is encoded in a one-hot code. Since the one-hot code produces two transitions if the previous reference was also in the one-hot code and an average of $n/2$ transitions when the previous reference is arbitrary, the number of transitions is reduced by using a transition-signaling code [8]. In this case, before sending the reference through the bus an XOR operation is performed with the previous value sent, always resulting in one transition.
4. One value can be sent using a zero-hot code, which with transition signaling produces zero transitions. This code should be used for the most-frequent event, which we have determined to be a repetition of the same offset for the current working zone.
5. When there is a reference that does not correspond to a working zone pointed by any Pref, it is not possible to send an offset; in such a case, the entire current memory reference is sent over the bus. Moreover, it is necessary to signal this situation.
6. In general, the total number of working zones of a program can be larger than the number supported by the hardware. Consequently, these have to be replaced dynamically. The most direct possibility is to replace an active working zone as soon as there is a miss. However, in this case any arbitrary reference would disturb an active working zone. To reduce this effect, we incorporate additional registers (henceforth named *potential working zones*) that store the references that cause a miss. Various heuristics are possible to determine when a potential working zone becomes an active one.

3.1. Implementation decisions

In the general scheme presented above, there are many aspects that have to be decided to obtain a suitable implementation. These decisions affect both the complexity of the implementation and the energy reduction achieved. Since there are many interdependent parameters, it is not practical to explore the whole space. Below we indicate the decisions made and the rationale for them.

- The number of active and potential working zones affects the number of registers and associated logic (and therefore the encoder/decoder energy consumption) and the number of values of the identifier. In the evaluation of the scheme, we have explored a range of values and determined the one that produces the largest reduction. It was determined that a small number of working zones is sufficient.
- When there is a hit to a working zone, an offset and an identifier are sent. There are choices for the set of values of the offset and the code of the identifier. Since the offset is sent in a one-hot code (with transition signaling) the set of values is directly related to the number of bits required. We have decided to use all bits of the original bus to send the offset. Moreover, we have seen that the number of hits is maximized if positive and negative offsets are used. Since all bits of the original bus are used for the offset, it is necessary to have additional wires for the identifier and, to minimize these additional wires, we use a binary code. We have considered using bits of the original bus for the identifier (thus reducing the offset bits) and have observed a significant increase in I/O activity with respect to the use of separate bits.
- When there is a miss, this situation has to be signaled to the receiver. Since in that case, all bits of the original bus are used to send the address, this hit/miss condition has to use some additional wire. As we already have decided to use additional wires for the identifier, one value on these wires might be used to signal the miss. However, this would produce a few transitions when changing from a hit to a miss. To assure only one transition, we have assigned an additional bit to signal a miss.
- The search for a hit in a working zone requires subtracting the previous address with the current one and detecting whether the offset is in the acceptable range. For the selection of which zones to check it is possible to use any of the schemes used for caches. Because of the small number of working zones, we have chosen a fully-associative search.
- There are two replacement procedures required: for the active working zones and for the potential working zones. As indicated before, when there is a miss the address is placed in a potential working zone. Since there are few of these, we use the LRU algorithm for this placement. Moreover, it is necessary to determine when a new active working zone appears and, in this case, which active working zone to replace. Among the possible alternatives, we have chosen to initiate a new active working zone when there is a hit in a potential working zone. Again, here we use the LRU replacement algorithm.

	m-wire encoded address and data bus					
	`WZ_miss` (1 wire)	`ident` ($\lceil log_2(H+M) \rceil$ wires)	`word_address` (n_a wires)	`dbus_WZ_coded` (1 wire)	`dbus_BI_coded` (1 wire)	`word_data` (n_d wires)
WZ **format**	0	WZ index	offset or last address value	1	don't care	offset or last data value
				0	1	**BI**(data)
					0	complete data
Non WZ **format**	1	don't care	complete address	don't care	1	**BI**(data)
					0	complete data

Table 1. Information assigned to each of the fields of the encoded address and data buses when there is a hit (WZ format) and a miss (Non WZ format) in the H working zones and in the M potential working zones. **BI** stands for bus-invert.

4. Extension of the WZE to the data bus

The technique for the address bus is now extended to include also the data bus. This extension is based on the fact that in many instances the data values of consecutive accesses to a working zone differ by a small amount. If that is the case, the data is also sent as an offset, coded in the one-hot encoding with transition signaling. To implement this extension, as illustrated in Figure 2, we include an additional register, called *Pdat*, per working zone.

The special case of zero-hot coding (which produces zero transitions) is reserved to the most frequent offset, which is the zero offset; this is in contrast to the address case, in which the zero-hot encoding is used for the case in which the value of the offset is repeated.

In summary, for the data bus, to send the offset it is necessary to compare the current data value with the Pdat associated to the current working zone, and the following two situations occur:

- the offset is zero: send again the previous value sent over the bus (zero transitions)
- the offset is not zero: send the one-hot encoded value of the offset using transition signaling (one transition).

On the other hand, if the access is not to an active working zone or if the offset is larger than possible for the one-hot encoding, the whole value is sent through the bus. An additional wire is required to distinguish these cases. Moreover, to further reduce the bus transitions, when the value in the data bus is not encoded by the WZE method we use the bus-invert technique for the whole data value (thus requiring an extra wire); for the address bus we saw that the benefits of using the bus-invert in this case were very small.

The decoding of an offset in the receiver is done also in two steps: XORing the value that it receives with the previous one, and retrieving the one-hot of the result. When the XORing produces a 0 vector, the two values were the same and this is interpreted as a repetition of the previous data value when that same working zone was last accessed.

4.1. Address and data bus fields

As shown in Table 1 the encoded address and data bus consists of five fields:

- one wire to indicate whether there has been a hit or a miss in any of the zones (`WZ_miss`)
- $\lceil \log_2(H+M) \rceil$ wires to specify one of H working zones or M potential zones (`ident`)
- n_a wires of the original address bus (`word_address`)
- one wire to indicate if the data bus has been encoded using the offset (`dbus_WZ_coded`)
- one wire to indicate, in the case of a miss in the working zones, whether the data bus is coded with the bus-invert technique (`dbus_BI_coded`)
- n_d wires of the original data bus (`word_data`).

Therefore, $m = n_a + n_d + \lceil \log_2(H+M) \rceil + 3$ wires are required.

5. Simulations

In this Section, the WZE technique is evaluated for both the multiplexed instruction/data bus and the multiplexed address bus [2] and compared to the rest of the techniques. The results are first reported for each type of bus (with the activity associated to the `WZ_miss` and `ident` fields included in the address bus activity). Then, the best of the rest of the techniques for each type of bus is compared with the WZE technique.

Traces from several SPEC95 benchmarks (*gcc*, *li*, *m88ksim*, *hydro2d*, *su2cor*, *tomcatv* and *turb3d*) are used to perform the evaluations. These traces contain memory references along with the corresponding data values. A bit-width of 32 is assumed for the data and address buses. We

[2] A study for other address bus configurations is presented in [6].

Instruction and Data caches	Multiplexed Address Bus										
	non encoded	WZE	Rest of techniques								
			Gray	BI			T0	T0/BI		inc-xor	dbm-vbm$_{G=4}$
				G=1	*G=2*	*G=4*		*G=1*	*G=4*		
No	5.5	(2.1) 2.6	5.1	4.7	4.3	4.2	4.8	4.0	3.5	4.3	5.4
Yes	4.5	(3.2) 4.0	3.7	4.4	4.2	3.7	4.1	4.1	3.4	4.0	4.4

Instruction and Data caches	Multiplexed Instruction/Data Bus										
	non encoded	WZE	Rest of techniques								
			Gray	BI			T0	T0/BI		inc-xor	dbm-vbm$_{G=4}$
				G=1	*G=2*	*G=4*		*G=1*	*G=4*		
No	8.5	(5.7) 6.2	8.9	8.2	7.5	7.3	8.5	8.2	7.3	8.7	8.3
Yes	7.0	(3.7) 4.4	7.3	6.7	6.3	6.1	7.0	6.7	6.1	7.3	6.7

Table 2. Average number of I/O transitions/reference for all the encoding techniques for the multiplexed address and multiplexed instruction/data buses. Energy overhead included only for WZE (in parenthesis, without overhead). **BI** stands for bus-invert.

also evaluate the effect of a 8K byte, direct-mapped, 32-byte line, write-back instruction and data cache. For the scenario without caches, the first 10M references to memory (after the first 10M instructions executed) are used in the simulations. For the scenario with caches, the first (10/8)M misses to either cache (after the first 10M instructions executed) are used, which correspond to 10M data references since the cache line is 32 bytes and the data bus width is 32 bits.

The data bus is logically divided into bytes, and the processor reads or writes one, two or four aligned bytes. For the unencoded data bus, whenever one or two bytes are accessed, the rest of the bus has undefined values (this is the behavior, for example, of the i486 data bus [4]), which are modeled as random values in the simulations.

The configuration for the WZE technique, i.e. the values for H and M, is derived from [6], where we saw that the best configuration to reduce the energy of the multiplexed address bus is ($H=2$,$M=2$) when no caches are present, and ($H=4$,$M=2$) in the presence of caches. Thus, the heuristic we follow in this paper is to fix the configuration of the WZE technique to minimize as much as possible the energy on the multiplexed address bus, and then reduce the energy of the multiplexed instruction/data bus. The rationale behind this is that we expect more energy reductions by encoding the address bus rather than the data bus since the locality property is stronger in the address bus.

The WZE technique is compared to the Gray, T0, bus-invert (with one, two, and four groups), combined T0/bus-invert (with one and four groups for the bus-invert part), inc-xor and dbm-vbm (with four groups) encoding techniques. Although some of these techniques have been proposed specifically for the address bus, for completeness we have evaluated them also for the data bus.

5.1. Energy overhead evaluation

Since the WZE technique is more complex and needs more area than the other techniques, we evaluate the energy overhead for the WZE technique and incorporate it in the overall energy, whereas we do not include this overhead for the others. This difference is due to the fact that we have not implemented the hardware for the other techniques; the hardware for the WZE (for the address bus encoding only) is described in [6].

In [6] we estimate the energy overhead for each of the WZE configurations and for each type of address bus. This energy is calculated as the average number of transitions generated in the encoder and decoder hardware. This number is then multiplied by the capacitance ratio between an on-chip and off-chip nodes, which in this work is considered to be 10^{-3}, to obtain the equivalent number of I/O transitions per reference. According to [6], for the multiplexed address bus and configurations ($H=2$,$M=2$) and ($H=4$,$M=2$) the energy overheads are 0.49 and 0.71 I/O transitions per reference, respectively. We will use these values also to penalize the overhead of the logic due to the Pdat registers. Note that this penalty is pessimistic since it implies that a duplication of the whole encoder/decoder is done for the data bus, and this is not the case since a substantial portion of logic is shared for encoding/decoding the address and data buses.

5.2. Results

Table 2 shows the average number of I/O transitions per reference for the multiplexed address and multiplexed instruction/data buses. Table 3 summarizes the results. In the first part we show the average number of I/O transitions per reference for both the address and data buses, for the WZE technique and for the best of the other encoding techniques.

	Mux. Address Bus			Mux. Ins/Data Bus			Extra wires†	
	WZE	**Best of rest**		**WZE**	**Best of rest**		**WZE**	**Best of rest**
Avg. (no caches)	(2.1) 2.6	T0/BI$_{G=4}$	3.5	(5.7) 6.2	BI$_{G=4}$	7.3	5	9
Avg. (with caches)	(3.2) 4.0	T0/BI$_{G=4}$	3.4	(3.7) 4.4	BI$_{G=4}$	6.1	6	9

	Mux. Address Bus Ratio		Mux. Ins/Data Bus Ratio		Overall Ratio†	
	vs. non encoded	**vs. best of rest**	**vs. non encoded**	**vs. best of rest**	**vs. non encoded**	**vs. best of rest**
Avg. (no caches)	(0.39) 0.47	(0.60) 0.74	(0.67) 0.73	(0.77) 0.84	(0.56) 0.63	(0.72) 0.81
Avg. (with caches)	(0.71) 0.86	(0.94) 1.18	(0.53) 0.63	(0.61) 0.72	(0.60) 0.72	(0.73) 0.88

†To encode both buses

Table 3. Results summary. Energy overhead is only shown for the WZE technique (in parenthesis without overhead).

Moreover, the number of extra I/O wires used to encode both the address and data buses is also presented.

The second part of Table 3 provides the energy reduction ratios of the WZE technique versus the non-encoded case and the best of the rest of the techniques, for each of the buses. Finally, the overall ratio (comprising both the instruction/data and address buses) is given.

The results show that for the multiplexed bus

- **without caches**, the WZE technique is well suited for the reduction of energy. Moreover, for the address bus the best of the other techniques is the T0/BI, since it combines the effect of the T0 technique which is suitable for a group of instruction addresses with the bus-invert which is useful for branches and for the transition between instruction and data addresses. On the other hand for the data bus, the best of the other techniques is bus-invert. In both cases, the advantage of the WZE technique stems from the use of several working zones and the use of offsets.

- **with caches**, the WZE technique is quite effective for the data bus. This might be due to the fact that the memory access is to a whole cache line, with words belonging to the same working zone that may have similar values.

6. Conclusions

This work extends the Working Zone Encoding method, originally applied to encoding an external address bus, to the data bus. Among the possible bus organizations, we have considered a multiplexed address bus (for instruction and data addresses) and a multiplexed instruction/data bus.

Several SPEC95 streams of references to memory along with the corresponding data values are used to evaluate the technique. We conclude that the Working-Zone Encoding technique significantly reduces the activity in both buses. Moreover, for the case without caches, the technique presented here outperforms other previous bus encoding proposals for low power, such as Gray, bus-invert, T0, combined T0/bus-invert, inc-xor and dbm-vbm. On the other hand, for the multiplexed address bus with instruction cache the best scheme is either the WZE presented here or bus-invert with four groups, depending on the overhead of these two techniques. In any case, the WZE method requires fewer additional wires when coding both buses.

References

[1] H. Bakoglu. *Circuits, Interconnections and Packaging for VLSI*. Menlo Park, CA, 1990.

[2] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano. Address bus encoding techniques for system-level power optimization. In *Design, Automation and Test in Europe*, pages 861–866, Feb. 1998.

[3] R. O. H. Mehta and M. Irwin. Some issues in gray code addressing. In *Great Lakes Symposium on VLSI*, pages 178–180, Mar. 1996.

[4] Intel Corp. *Microprocessors Vol. I*, 1993.

[5] E. Musoll, T. Lang, and J. Cortadella. Exploiting the locality of memory references to reduce the address bus energy. In *Int. Symp. on Low Power Design and Electronics*, pages 202–207, Aug. 1997.

[6] E. Musoll, T. Lang, and J. Cortadella. Working-Zone Encoding for reducing the energy in microprocessor address buses. Technical Report http://www.eng.uci.edu/numlab/archive/pub/nl98b/01.ps.Z, University of California, Irvine, Mar. 1998. To be published in the next special issue of Trans. on VLSI on low-power electronics and design.

[7] S. Ramprasad, N. Shanbhag, and I. Hajj. Coding for low-power address and data busses: a source-coding framework and applications. In *Proc. of the Int. Conf. on VLSI Design*, pages 18–23, Jan. 1998.

[8] M. Stan and W. Burleson. Low-power encodings for global communications in CMOS VLSI. *IEEE Trans. on VLSI Syst.*, pages 444–455, 1997.