Working-Zone Encoding for Reducing the Energy in Microprocessor Address Buses

Enric Musoll, Tomás Lang, and Jordi Cortadella

***Abstract*— The energy consumption due to input–output pins is a substantial part of the overall chip consumption. To reduce this energy, this work presents the working-zone encoding (WZE) method for encoding an external address bus, based on the conjecture that programs favor a few working zones of their address space at each instant. In such cases, the method identifies these zones and sends through the bus only the offset of this reference with respect to the previous reference to that zone, along with an identifier of the current working zone. This is combined with a one-hot encoding for the offset. Several improvements to this basic strategy are also described. The approach has been applied to several address streams, broken down into instruction-only, data-only, and instruction-data traces, to evaluate the effect on separate and shared address buses. Moreover, the effect of instruction and data caches is evaluated. For the case without caches, the proposed scheme is specially beneficial for data-address and shared buses, which are the cases where other codings are less effective. On the other hand, for the case with caches the best scheme for the instruction-only and data-only traces is the WZE, whereas for the instruction-data traces it is either the WZE or the bus-invert with four groups (depending on the energy overhead of these techniques).**

***Index Terms*— Address bus, encoding for low power, low-power, microprocessor, input–output energy.**

I. Introduction

THE input–output energy is a substantial fraction of the total energy consumption of a microprocessor [1], [2], 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. In this work, we consider an encoding to reduce the activity in the address bus. This encoding is 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 encoding of the offset. The main contribution of the working zone encoding (WZE) technique is the tracking of the most recent working zones.

The encoding technique can be applied to any system in which there is a processing chip and an external memory referenced through a dedicated address bus. These systems can range from application-specific systems to general-purpose processors. This paper is a continuation of the work in [3], where the WZE technique was first described and evaluated for some application-specific benchmarks. Now, we extend this work as follows:

- by improving the technique, to reduce the interference of random addresses on the working zones held by the encoder/decoder hardware;
- by evaluating the technique for general-purpose benchmarks (where the memory references do not present such a regular pattern as in the application-specific benchmarks used in [3]); moreover, we have evaluated the effect on separate (instruction and data) and shared address buses;
- by incorporating the effect of instruction and data caches.

A. Previous Work

Several encoding techniques for reduced bus activity have been reported, such as one-hot [4], Gray [5], bus-invert [6], T0 and combined bus-invert/T0 [7], and inc-xor and dbm-vbm [8].

One-hot encoding results in a reduced activity because only two bits toggle when the value changes. However, it requires a number of wires equal to the number of values encoded, so that it is not practical for typical buses.

Gray and T0 encoding are targeted to situations in which consecutive accesses differ by one (or by a fixed stride). The Gray encoding is useful because the encoding of these values differs by one bit. In the T0 encoding an additional wire is used to indicate the consecutive access mode, and no activity is required in the bus.

The bus-invert method [6] 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. The bus-invert technique has been combined with T0 in [7], thus obtaining more activity reduction than each of the techniques by itself.

A source-coding framework is proposed in [8] 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

Manuscript received December 15, 1997; revised May 15, 1998. This work was supported in part by CICYT TIC 95-0419.

E. Musoll is with Cyrix (National Semiconductor Corp.), Santa Clara, CA 95054 USA.

T. Lang is with the Department of Electrical and Computer Engineering, University of California, Irvine, CA 92697 USA.

J. Cortadella is with the Department of Software, Universitat Politècnica de Catalunya, Barcelona 08071 Spain.

Publisher Item Identifier S 1063-8210(98)08515-1.

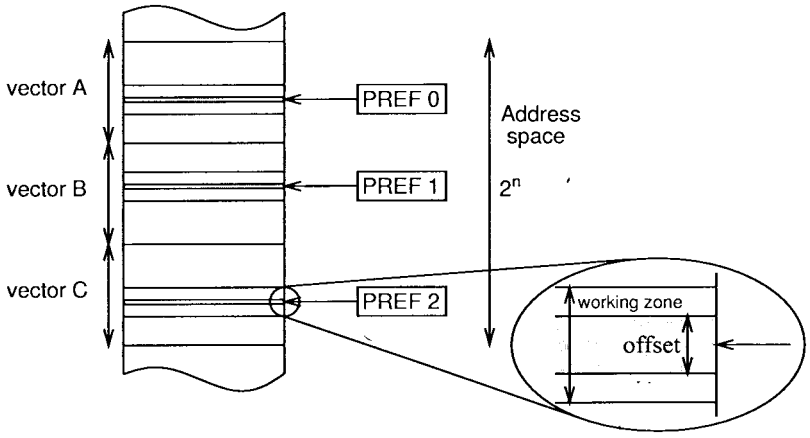

Fig. 1. Address space with three vectors.

number of transitions is reduced in the likely case when the prediction error has a small number of ones. For addresses, the only new code proposed is the inc-xor code, in which the prediction is the previous address plus one and the prediction error is obtained by the bit-wise XOR operation. This code is most beneficial for instruction address buses, where sequential addressing is prevalent. Also presented are codes which relate to the one-hot encoding used here, but they are applied only to the data bus.

The reduction of activity in the address bus has also been addressed by other techniques [8]–[11], which are not suitable for general-purpose microprocessors.

II. Working Zone Encoding Technique (WZE)

The basis of the WZE technique 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. As a simple example consider an application that works with three vectors (A, B, and C) as shown in Fig. 1. 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 P_{refs}) holding a pointer to each active working zone, the sender would only need to send:
   a) the offset of the current memory reference with respect to the P_{ref} associated to the current working zone;
   b) an identifier of the current P_{ref}.
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 [11]. 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 P_{ref}, 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.

A. 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

TABLE I
ADDRESS BUS FIELDS FOR A HIT (WZ FORMAT) AND A MISS (NON WZ FORMAT). IN PARENTHESIS, THE NUMBER OF BITS

	m-wire bus		
	Pref_miss (1)	ident ($\lceil \log_2(H+M) \rceil$)	word (n)
WZ format	0	WZ index	offset or last bus value
Non WZ format	1	don't care	complete address

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.
- With these decisions, as shown in Table I, the bus consists of the following three fields: 1) the n bits of the original address bus (named `word`), 2) $\lceil \log_2(H+M) \rceil$ bits to specify one of the H active or M potential working zones (named `ident`), and 3) one bit to indicate hit or miss (called `Pref_miss`). Consequently, the total number of wires is $m = n + \lceil \log_2(H+M) \rceil + 1$.
- 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.

B. Encoding and Decoding Algorithms

The WZE algorithm for the encoder/decoder is shown in Fig. 2. Values 1 to H ($H+1$ to $H+M$) in `ident` belong to active (potential) working zones. The registers used for the encoding are as follows:

- $\mathbf{Pref}_j$ contains the last address to working zone j;
- $\mathbf{prev\_off}_j$ is the offset of the last reference to zone j;
- **prev_sent** is the previous value sent over the bus:
- **prev_ident** is the value of `ident` of the previous hit.

Similarly, the registers involved for decoding are as follows:

- **prev_received** is the previous value received from `word`;
- $\mathbf{prev\_off}_j$ and $\mathbf{Pref}_j$ (as in the encoding algorithm).

The fraction of chip area required for the encoder and decoder hardware is small [3]. The delay introduced by the encoder logic may be considerable (roughly, a subtractor, a simple limit detector, a one-hot encoder, and two muxes). To reduce the bus access time, the decoder delay can be overlapped with the virtual-to-physical address translation. This can be done by using the virtual address to determine whether there is an offset in any of the working zones. This procedure is correct as long as the offset does not cross page boundaries, in which case the access is not treated as a hit. The simulations show that for a page size of 1 KB or larger the effect on the activity reduction is negligible.

III. EVALUATION AND COMPARISON

In this section, several traces (`espresso`, `gsview` and `gzip` from a MIPS processor; `CC1` and `SPICE` from the DLX processor) are used to evaluate the WZE encoding (for $H = 1, 2, 4, 8$, and $M = 0, 1, 2, 4$) and to compare it with previously proposed encodings, namely, Gray, T0, bus-invert (with one, two, and four groups), combined T0/bus invert (with one group, as done in [7]), inc-xor and dbm-vbm (the bus is divided in four groups).

To evaluate different types of buses, the traces have been broken down into instruction-only and data-only (for the case of two dedicated address buses) and instruction-data traces (for the most-common case of one shared address bus). Moreover, an instruction and data cache (4 KB I-Cache and 4 KB D-Cache or 8 KB Unified Cache; direct mapped; 32 byte line; write-back; no prefetching) has been considered. The address bus is of 32 bits; however, the two (five) least-significant bits do not need to be encoded in the cases without (with) caches.

We have estimated the energy overhead of the WZE method. For this, for each program, type of trace, and WZE configuration, we obtained the equivalent number of I/O transitions per reference by calculating the average number of transitions produced in the encoder and decoder hardware multiplied by the internal/external node capacitance ratio, which in this work is considered to be 10^{-3}.

```
for 1 ≤ i ≤ H + M do Δ_i = current - Pref_i                    (1)
if ∃Δ_r such that −n/2 ≤ Δ_r ≤ n/2 − 1 then
   offset = Δ_r
   Pref_miss = 0
   ident = r
   if offset = prev_off_r then
      word = prev_sent
   else
      word = transition-signaling(one-hot(offset))
   Pref_r = current
   prev_off_r = offset
   prev_ident = r
   if H + 1 ≤ r ≤ H + M then
      Pref_j = current   (1 ≤ j ≤ H)                           (2)
      prev_off_j = offset
else
   Pref_miss = 1
   ident = prev_ident                                          (3)
   word = current
   if M ≠ 0 then
      Pref_j = current   (H + 1 ≤ j ≤ H + M)                   (2)
   else
      Pref_j = current   (1 ≤ j ≤ H)                           (2)
   (leave prev_off_j as before)                                (4)
prev_sent = word
```

(1) active working zone search; in this work, fully associative

(3) ident is don't care; its previous value is sent

(a)

```
if Pref_miss = 0 then
   xor = prev_received XOR word
   if xor = 0 then
      current = Pref_ident + prev_off_ident
      (leave prev_off_ident as before)
   else
      current = Pref_ident + one-hot-retrieve(xor)
      prev_off_ident = one-hot-retrieve(xor)
   Pref_ident = current
   if ident > H then
      Pref_j = current,   (1 ≤ j ≤ H)                          (2)
      if xor = 0 then
         (leave prev_off_j as before)
      else
         prev_off_j = one-hot-retrieve(xor)
else
   current = word
   Pref_j = current
   (leave prev_off_j as before)                                (4)
prev_received = word
```

(2) replacement algorithm; in this work, LRU

(4) $\mathbf{prev\_off}_j$ is not modified since no previous offset is known

(b)

Fig. 2. (a) Encoder and (b) decoder algorithms.

TABLE II
RESULTS SUMMARY: TWO RATIOS OF THE WZE WITH RESPECT TO THE BEST OTHER ENCODING ARE SHOWN. IN PARENTHESIS RATIO WITH NO OVERHEAD, WITHOUT PARENTHESIS INCLUDES OVERHEAD FOR WZE BUT NOT FOR THE OTHER ENCODING. THE OVERHEAD IS THE EQUIVALENT NUMBER OF I/O TRANSITIONS PER REFERENCE DUE TO THE ENCODER AND DECODER HARDWARE

Trace	Avg. I/O trans./ref. (non encoded)	Best WZE technique		Best of rest of techniques†		Ratio vs. non encoded	Ratio vs. best of rest†
		Config.	Extra wires	Config.	Extra wires		
instruction-only/no ICache	1.99	$H=1, M=0$	1	inc-xor	0	0.44	(0.93) 1.04
data-only/no DCache	7.15	$H=4, M=2$	4	Gray	0	0.57	(0.65) 0.78
instr.-data/no caches	7.00	$H=2, M=2$	3	$\mathrm{BI}_{G=4}$	4	0.43	(0.45) 0.54
instruction-only/ICache	4.00	$H=2, M=0$	2	$\mathrm{BI}_{G=4}$	4	0.78	(0.74) 0.82
data-only/DCache	6.90	$H=4, M=2$	4	Gray	0	0.54	(0.76) 0.94
instr.-data/Unified cache	5.94	$H=4, M=2$	4	$\mathrm{BI}_{G=4}$	4	0.80	(0.85) 1.00

Table II is a summary of the results. Since the benchmark traces are of different lengths, the average results reported have been obtained by weighting the results for each trace according to its length. The second column shows the average number of address bus transitions per reference when no encoding is performed. Then, the best WZE configuration and the best of the rest of techniques are indicated, together with the corresponding wire overhead. The first of the two ratios compares the WZE with the case of no encoding, showing significant improvements for all types of traces. The second ratio compares with the best of the other encodings showing also improvements, although smaller. Since we have not evaluated the overhead of the other techniques, the actual ratio should be between the value in parenthesis and that without parenthesis.

IV. CONCLUSIONS AND FUTURE WORK

This paper presents the WZE method for encoding an external address bus. For the case without instruction and data caches, the proposed scheme is specially beneficial for data-address and shared buses, which are the cases that have the largest number of bus transitions and for which other codings are less effective; for instruction-address buses, the proposed scheme is similar to the best previously proposed, as can be deduced by the fact that the best WZE encoding is for one working zone. For the case with caches the WZE method is the best scheme for the instruction-only and data-only traces, whereas for the instruction-data traces, the best is either the WZE or the bus-invert with four groups (depending on the energy overhead of these techniques).

The major limitation of the WZE is its larger encoder and decoder logic overhead, which limits the benefits of the I/O activity reduction. Also it is necessary to adopt techniques to minimize the additional delay, a possibility for this is to overlap the encoding with the virtual-to-physical address translation.

The WZE technique can be extended to the data bus since data for successive accesses to a working zone frequently differ by a small amount, and, therefore, it is also effective to use an encoding based on offsets for the data bus. Moreover, the proposed method can be combined with other methods, such as bus-invert, to obtain additional reductions. The authors are currently exploring these extensions of the technique.

REFERENCES

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

[2] T. Burd and B. Peters, "A power analysis of a microprocessor: A study of an implementation of the MIPS R3000 architecture," Univ. California at Berkeley, Tech. Rep., May 1994.

[3] 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 Design Electron.*, Aug. 1997, pp. 202–207.

[4] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. New Yi]ork: Kluwer Academic, 1995.

[5] R. M. Owens H. Mehta, and M. J. Irwin, "Some issues in gray code addressing," in *Proc. Great Lakes Symp. VLSI*, Mar. 1996, pp. 178–180.

[6] M. R. Stan and W. P. Burleson, "Bus-invert coding for low power I/O," *IEEE Trans. VLSI Syst.*, 1995, pp. 49–58.

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

[8] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "Coding for low-power address and data busses: A source-coding framework and applications," in *Proc. Int. Conf. VLSI Design*, Jan. 1998, pp. 18–23.

[9] P. R. Panda and N. D. Dutt, "Reducing address bus transitions for low power memory mapping," in *Proc. EDAC*, Mar. 1996.

[10] 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 Design and Electron.*, Aug. 1997, pp. 24–29.

[11] M. R. Stan and W. P. Burleson, "Low-power encodings for global communications in CMOS VLSI," *IEEE Trans. VLSI Syst.*, 1997, pp. 444–455.