Reducing the Energy of Address and Data Buses with the Working-Zone Encoding Technique and its Effect on Multimedia Applications *

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

Tomás Lang
Dept. of Electrical and Computer Eng.
University of California at Irvine
Irvine, CA 92697
tomas@ece.uci.edu

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

Abstract

The energy consumption due to I/O pins is a substantial part of the overall chip consumption. This paper gives an overview of the Working Zone Encoding (WZE) method for encoding for low power the external address and data buses, 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 address (data) bus only the offset of this reference (data value) with respect to the previous reference (data value) to that zone, along with an identifier of the current working zone. This is combined with a one-hot encoding for the offset.

The paper then focuses on preliminary work on the following two topics:

- reduction of the effect of the WZE delay on the bus access time by overlapping this delay with the virtual to physical address translation. Although the modification to allow this overlapping might increase the bus energy, simulations of the SPEC benchmarks indicate that for a page size of 1 KB or larger the effect is negligible.
- extension of the technique for the data bus to some multimedia applications which are characterized by having packed bytes in a word. For two typical applications, the data-only data bus and data-only address bus I/O activity is reduced by 74% and 51% with respect to the unencoded case, and by 68% and 33% with respect the best of the rest of the encoding techniques.

1 Introduction

The I/O energy is a substantial fraction of the total energy consumption of a microprocessor [4], because the capacitance associated with an external pin is between one hundred and one thousand times larger than that corresponding to 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.

*This work has been partially funded by CICYT TIC 95-0419.

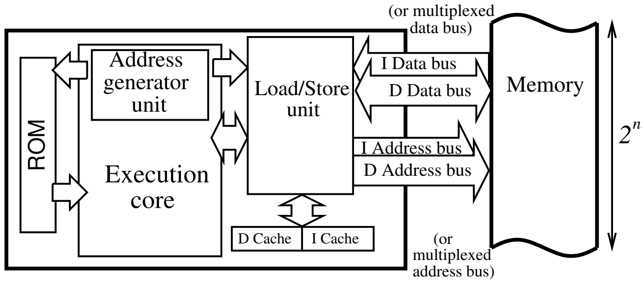

Figure 1: Bus types in a general-purpose microprocessor.

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 both of these buses. If the value carried by n bits has to be transmitted over a bus, a reduction in the switching activity of this bus may be obtained at the cost of extra hardware in the form of an encoder on the sender device, a decoder on the receiver device, and potentially a larger number of wires m.

In [10] and [11] we have presented the Working-Zone Encoding (WZE) method, which is based on the conjecture that applications favor a few working zones of their address space. Moreover, consecutive addresses to each of these working zones frequently differ by a small amount. In such cases, an offset with respect to the previous address for that zone is sent, together with a zone identifier. The offset is encoded so as to reduce the activity of the address bus. This scheme is extended to the data bus [8] by noticing that the data for successive accesses to a working zone frequently differ also by a small amount, so that it is effective also to send the offset.

To evaluate the effectiveness of the technique in general applications, several SPEC95 streams of references to memory along with the corresponding data values were used. Among the possible bus organizations (see Figure 1), we have considered a multiplexed address bus (for instruction and data addresses) and a multiplexed instruction/data bus, with and without a unified 8K-byte direct-mapped cache. We also have compared with previously proposed encodings: Gray, bus-invert, T0, combined T0/bus-invert, inc-xor and dbm-vbm. Table 1 summarizes the results obtained. The table shows the energy reduction ratios of the WZE tech-

	Address Bus Ratio		Ins/Data Bus Ratio		Both Buses Ratio	
	vs. non encoded	vs. best of rest	vs. non encoded	vs. best of rest	vs. non encoded	vs. best of rest
Avg. (no cache)	(0.39) 0.47	(0.54) 0.66	(0.67) 0.73	(0.77) 0.84	(0.56) 0.63	(0.69) 0.78
Avg. (with cache)	(0.71) 0.87	(0.87) 1.06	(0.53) 0.63	(0.61) 0.72	(0.60) 0.72	(0.70) 0.85

Table 1: Results summary for some SPEC95 streams. Ratios are calculated as *Energy WZE/Energy other*, being *other* the unencoded case and the best of the rest of the techniques evaluated. Energy overhead of the encoder/decoder logic is only included for the WZE technique. In parenthesis, without overhead.

nique with respect to the unencoded case and to the best of the rest of the techniques, for each of the buses and for both together. We conclude that the WZE encoding significantly reduces the activity in both buses. Moreover, for the case without cache, the technique presented here outperforms the other previous bus encoding proposals for low power. On the other hand, for the case with cache the best scheme for the address bus 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 outperforms the rest of the techniques when both buses are encoded, and requires fewer additional wires than the bus-invert with four groups.

In this paper we give an overview of the WZE technique and summarize previous work on the topic; this material is similar to that of [8] and should give the reader a reasonable understanding of the method. For more details consult [10, 11]. We then focus on preliminary work on the following two topics:

- Reduction of the effect of the WZE delay on the bus access time by overlapping this delay with the virtual to physical address translation.
- Use of the WZE technique in multimedia applications, which are characterized by having packed bytes in a word. Because of the particular features of these applications, we explore the possibility of special modifications to the encoding technique for the data bus.

1.1 Previous work

Several encoding techniques for reduced bus activity have been reported, such as one-hot [6], Gray [7], bus-invert [13], T0 and combined bus-invert/T0 [5], and inc-xor and dbm-vbm [12].

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 [13] 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 [5], thus obtaining more activity reduction than each of the techniques by itself.

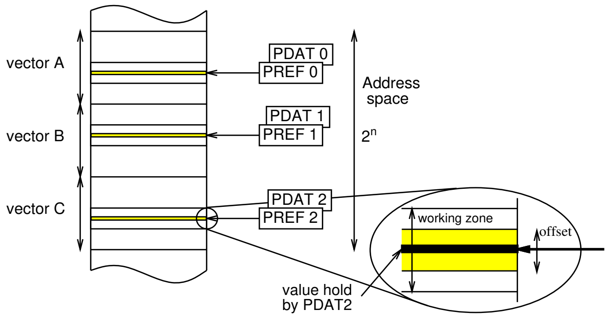

Figure 2: Address space with three vectors.

A source-coding framework is proposed in [12] as well as some specific codes. The scheme is based on obtaining a prediction function and a prediction error. This prediction error is `XOR`ed 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. For addresses, the only new code proposed is the inc-xor code, in which the prediction is the previous address plus one (or any fixed stride) 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 1-hot encoding used in this paper, such as the dbm-vbm, that are applied to the data bus. In the dbm-vbm technique, the prediction is the previous address and the prediction error is obtained by a function that increases as the absolute difference between the current input and the prediction increases. Afterwards, code-words with fewer 1's are assigned to smaller error values. Finally, the result is `XOR`ed with the previous value sent to the bus.

2 Overview of the WZE technique

In this Section an overview of the WZE technique for the address bus is given along with the implementation decisions made and the rationale behind them. Afterwards, an extension of the WZE technique [8] is reviewed which allows the data bus to be encoded by reusing a large portion of the hardware already used to encode the address bus.

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 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 [14]. 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 0-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.

2.1 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.

2.2 Extension to the data bus

The technique for the address bus can be 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 can also be sent as an offset, coded in the one-hot encoding. In this case, the zero-hot encoding is used when the offset is zero.

To implement this extension, as illustrated in Figure 2, we include an additional register, called *Pdat*, per working zone. On the other hand, if the access is not to an active working zone or if the offset is larger than possible for the

Value sent over the bus (either address or data)	Transition signaling and one-hot retrieval		Receiver action (address bus)	Receiver action (data bus)
	1. XORing	*2. One-hot retrieval*		
(-) 010011	-	-	-	-
(2) 011011	001000	3	offset 3 (Pref #2)	offset 3 (Pdat #2)
(1) 011011	000000	†	same offset (Pref #1)	same data value (Pdat #1)
(1) 011001	000010	1	offset 1 (Pref #1)	offset 1 (Pdat #1)

Table 2: Example of the decoding process. Assuming always hit; () in the first column indicates working zone number.

one-hot encoding, the whole value is sent through the bus. An additional wire is required to distinguish these cases.

In addition, 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 address bus we saw that the benefits of using the bus-invert in this case were very small.

In summary, for the address bus, to send the offset it is necessary to compare it with the previous offset to the same working zone. The following two situations occur:

- the offsets are the same: send again the previous value sent over the bus (zero transitions)
- they are different: send the one-hot encoded value of the offset using transition signaling (one transition).

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 situations occur:

- the values are the same: send again the previous value sent over the bus (zero transitions)
- they are different: send the one-hot encoded value of the offset (one transition).

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 (see Table 2):

- for the address bus, as a repetition of the previous offset to that same working zone,
- for the data bus, as a repetition of the previous data value when that same working zone was last accessed

2.3 Address and data bus fields

As shown in Table 3 (next page) the encoded address and data bus consists of five fields:

- the n_a wires of the original address bus (`word_address`)
- the n_d wires of the original data bus (`word_data`)
- $\lceil \log_2(H+M) \rceil$ wires to specify one of H working zones or M potential zones (`ident`)
- one wire to indicate whether there has been a hit or a miss in any of the zones (`WZ_miss`)
- one wire to indicate if the data bus has been able to be encoded using the offset (`dbus_WZ_encoded`)
- 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_encoded`).

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

3 Reducing the delay

The decoder introduces some delay in the bus access. Since this might be unacceptable, we now describe a method to overlap this delay with the virtual to physical address translation.

When an address translation is required, the most direct approach would be to perform the translation and then apply the encoding to the resulting physical address. However, this would produce an increased delay for the bus access. To reduce this delay we propose the following modification of the WZE technique:

- Use the virtual address to determine whether there is a hit in a working zone. To do this, each Pref contains the virtual address of the previous access to the corresponding zone.

Since, as it is well known, the translation modifies only the most-significant bits of the address (the page number) but keeps unaltered the least-significant bits (the page offset), this procedure is correct as long as the offset does not cross page boundaries. Consequently, it is necessary to detect when a change of page occurs and, in that case, the access is not treated as an offset.

Since the data value is not translated, this is encoded as in the original method.

Because of this modification in the WZE technique, the reduction in energy might be affected. This is because now we do not use offsets which cross a page boundary. On the other hand, the internal energy overhead might be reduced because now the detection of offset uses only the page-offset bits. The simulation of the SPEC benchmarks indicate that for a page size of 1 KB or larger the effect is negligible. We observe this in Table 4, where the I/O transitions per reference on the multiplexed address bus for the `gcc` benchmark (with unified cache) and for different page sizes is shown.

Page size	I/O transitions/reference (address bus)
(no pages)	3.26
4K bytes	3.26
1K bytes	3.29
256 bytes	3.36
64 bytes	3.72
1 byte	5.32

Table 4: Effect of the pages on the activity reduction if the WZE delay is overlapped with the virtual to physical address translation. For pages larger than 1K bytes the effect is negligible. Some unrealistic page sizes are also shown for comparison purposes. The data is for the `gcc` benchmark with unified cache.

	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_encoded` (1 wire)	`dbus_BI_encoded` (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	$\text{BI}_{G=1}$(data value)
					0	complete data
Non WZ **format**	1	don't care	complete address	don't care	1	$\text{BI}_{G=1}$(data value)
					0	complete data

Table 3: Information assigned to each of the several fields of the encoded address and data bus 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. The number of wires for each field is also shown.

4 Use in Multimedia Applications

In this Section we modify the WZE technique for the data bus for those multimedia applications that use data workloads composed of packets of data that may be brought from memory several in the same word. Examples of these are the image processing applications and we will evaluate the WZE technique for two particular examples. We show the results for the data-only data bus and the data-only address bus (see Figure 1). The code of the applications is assumed to be stored in an internal ROM (therefore no references to instructions are sent through the address bus and no instructions are fetched through the data bus).

In multimedia applications images are composed of pixels. These pixels consist of one or a few components, each with a relatively small number of values (for instance, in the examples we illustrate each pixel consists of three colors and each color can have 256 values). In such case, these pixels can be stored in one word which is subdivided into subwords for each component. In many applications, these components are accessed and processed simultaneously. We now describe how we modify the WZE technique for this situation.

The address bus encoding part of the technique is not modified, since the locality of reference is even more apparent in these applications with images. For the data part we do the following:

- Instead of using an offset for the whole word, we encode each of the bytes using the byte offset. In this way, the one-hot encoding allows eight possible offsets. We have determined that this number of offsets (which would correspond to offsets from -4 to +4) is insufficient to capture a significant portion of the data. Consequently, it is convenient to extend the encoding to include also k-hot encodings for $k > 1$.

- Moreover, if we allow two possibilities for each byte, namely that the data satisfies the offset range (a hit) or not (a miss), we would need individual wires per byte to indicate this hit/miss. Since this would be a significant wire overhead, we decided to code every data byte as an offset, no matter how large this offset is. Moreover, since a k-hot encoding with transition signaling generates k transitions in the bus, to reduce the average number of transitions we encode the offsets into a k-hot code, with smaller value of k for the more frequent offsets (that is, the value of k increases as the absolute value of the offset increases).

- When there is an address miss (that is, the address does not correspond to an active WZ), the most direct solution is to send the unencoded data value. However, we have found that the variation in the values of images is small enough so that it is better to send the offsets with respect to the last data value.

Note that in this modification of the WZE technique, the two extra wires (`dbus_WZ_encoded` and `dbus_BI_encoded` in Table 3) are not needed, since the data bus is always encoded and no bus-invert is done.

We have done some preliminary evaluations for two applications: *image alpha blending* and *motion estimation*. Both use color images where a pixel is composed of three bytes that specify the three main colors; a pixel is read or written per memory reference.

4.1 Image Alpha Blending

Alpha Blending [1] is used for imaging effects to merge two images together, weighting one image more than the other. Thus, alpha blending may be used for fading from one image to another and this is the case shown here: given two images of two different human faces, six images are generated so that the first image is transformed into the second one. This is done for three different sets of human faces [2].

The results are reported in Table 5, where the WZE technique is compared to the unencoded case and to the best of the rest of the techniques (bus-invert with three groups, one per byte). The WZE technique uses four active and two potential working zones.

4.2 Motion Estimation

The Motion-estimation algorithm [9] is used in video transmission to lower the bandwidth of the network where the video is being transmitted. The frame to be transmitted is divided into blocks which are compared to several blocks in the previous frame and the best match is selected.

The basic motion-estimation algorithm is applied to two different sets of images in motion [3, 2] (weather satellite, human face and football images). Results are shown in Table 6; the best of the rest of the techniques is the dbm-vbm with three groups. The WZE technique uses three active working zones and no potential working zones.

Images	**Data-only Data Bus**		
	non encoded	**WZE**	$\mathbf{BI}_{G=3}$
`claire` $\rightarrow$ `missa`	10.07	3.00	8.68
`claire` $\rightarrow$ `susie`	10.68	3.23	8.58
`missa` $\rightarrow$ `susie`	10.43	3.83	8.97

Table 5: Data-only data bus I/O transitions per reference for the alpha-blending application. The best of the rest of the techniques is the bus-invert with three groups.

Application	Data-only Data Bus Ratio		Data-only Address Bus Ratio		Both Buses Ratio	
	vs. non encoded	vs. best of rest	vs. non encoded	vs. best of rest	vs. non encoded	vs. best of rest
Image Blending	0.32	0.38	0.17	0.23	0.26	0.32
Motion Estimation	0.58	0.81	0.36	0.41	0.49	0.67

Table 8: Results summary for the two multimedia applications. Smaller ratio means fewer I/O transitions.

Image Sequences	Data-only Data Bus		
	non encoded	WZE	dbm-vbm$_{G=3}$
`weather satellite`	10.05	5.93	7.58
`human face`	9.13	5.14	5.89
`football`	10.79	6.22	7.79

Table 6: Data-only data bus I/O transitions per reference for the motion-estimation application. The best of the rest of the techniques is the dbm-vbm with three groups.

Application	Data-only Address Bus		
	non encoded	WZE	best of rest
Image Blending	7.78	1.33	5.78
Motion Estimation	6.71	2.40	7.21

Table 7: Data-only address bus I/O transitions per reference for the two example multimedia applications.

Table 7 shows the data-only address bus results for both applications. For the image blending the best of the rest of the techniques is the dbm-vbm whereas for the motion estimation is the bus-invert. In any case, the WZE technique clearly outperforms any previously proposed technique.

The averaged results for both applications are shown in Table 8. We show the ratio of the I/O transitions with respect the unencoded case and to the best of the rest of the techniques for the data-only data bus, data-only address bus and both buses. The I/O activity when coding both buses is reduced by 74% and 51% with respect the unencoded case, and by 68% and 33% with respect the best of the rest of the encoding techniques.

5 Conclusions

This paper gives an overview the Working Zone Encoding (WZE) method for encoding an external address and data bus, based on the conjecture that programs favor a few working zones of their address space at each instant.

For the general-purpose microprocessor multiplexed address bus and multiplexed instruction/data bus (with unified cache), the WZE significantly reduces the I/O activity, around 30% with respect to the unencoded case and 15% with respect the best of the rest of the encoding techniques. When no caches are present, the reductions are even larger.

For two multimedia applications using images as the workload, the activity in the data-only data bus and the data-only address bus is reduced by 74% and 51% with respect the unencoded case, and by 68% and 33% with respect the best of the rest of the encoding techniques. These results are promising and make it worthwhile to pursue the development of this approach.

References

[1] http://developer.intel.com/drg/mmx/appnotes/ap554.htm.

[2] http://ipl.rpi.edu/sequences/sequences.html.

[3] http://meteosat.e-technik.uni-ulm.de/meteosat.

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

[5] L. Benini, G. De 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, February 1998.

[6] A.P. Chandrakasan and R.W. Brodersen. *Low Power Digital CMOS Design.* Kluwer Academic Publishers, 1995.

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

[8] T. Lang, J. Cortadella, and E. Musoll. Extension of the Working-Zone Encoding method to reduce the energy on the microprocessor data bus. Technical Report http://www.eng.uci.edu/numlab/archive/pub/nl98b/02.ps.Z, University of California, Irvine, May 1998. To be published in the next Int. Conference on Computer Design (ICCD'98).

[9] C. Lin and S. Kwatra. An adaptive algorithm for motion compensated colour image coding. *IEEE Globecom*, 1984.

[10] 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, August 1997.

[11] 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, March 1998. To be published in the next special issue of Transactions on VLSI on low-power electronics and design.

[12] 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. of the Int. Conf. on VLSI Design*, pages 18–23, January 1998.

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

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