IEICE TRANS. ELECTRON., VOL.E87–C, NO.4 APRIL 2004

PAPER Special Section on Low-Power System LSI, IP and Related Technologies

Memory Data Organization for Low-Energy Address Buses

Hiroyuki TOMIYAMA[†a)], **Hiroaki TAKADA**[†], *Members*, *and* **Nikil D. DUTT**[††], *Nonmember*

SUMMARY Energy consumption has become one of the most critical constraints in the design of portable multimedia systems. For media applications, address buses between processor and data memory consume a considerable amount of energy due to their large capacitance and frequent accesses. This paper studies impacts of memory data organization on the address bus energy. Our experiments show that the address bus activity is significantly reduced by 50% through exploring memory data organization and encoding address buses.

key words: *compilers, embedded processors, memory data organization, low energy, bus encoding*

1. Introduction

Energy consumption is one of the most critical constraints in the design of portable embedded systems. System-level buses such as off-chip buses or long on-chip buses between IP blocks are often major sources of energy consumption due to their large load capacitance. An effective approach to bus energy minimization is to reduce their switching activities by encoding the buses, and a number of techniques for bus encoding have been developed so far.

Bus Invert Coding is one of well-known coding techniques [1]. The Bus Invert code is suited to data buses where consecutive two data values are less correlated with each other, but it is not so efficient for buses with a strong correlation. To solve this problem, several refinements were proposed, for example in [2]. In processor-based systems, accesses to instruction memory have a specific regularity, i.e., most accesses are sequential with a fixed stride value, and this regularity has been utilized by many encoding techniques. In [3], *Gray Coding* is adopted for address buses. With the Gray code, only one bit changes when memory accesses are in sequence. *T0* is a redundant coding technique with which no bit changes for in-sequence accesses [4], [5]. Afterward, several refinements based on T0 were proposed such as *Inc-Xor* [6] and *T0-C* [7], and their experimental results demonstrate these T0-based refinements outperform the Gray code and the original T0 code.

The Gray and T0-based codes are considered to be well suited to address buses of instruction memory. However, it is obvious that they are also effective for data memory address buses if many data accesses are sequential. In many media and DSP applications, a small number of kernel loops dealing with data arrays account for a significant portion of the total execution time, and such DSP kernels usually have a regularity in their data accesses. Of course, the regularity does not always mean sequential accesses. However, as we see later in this paper, the data accesses in many DSP kernels can be serialized to some extent by optimizing data organization in memory. Thus, the T0-based codes work efficiently.

This paper studies low-energy data organization in code generation of media and DSP applications. In this paper, we do not propose a new technique for data organization or bus encoding. The contribution of this paper is to demonstrate that data organization in memory has large impacts on energy consumption of buses between processor and data memory, and the bus energy can significantly be reduced by exploring memory data organization. This paper also demonstrates the effectiveness of T0-based encoding techniques, which were originally targeted towards (or considered to be effective for) instruction memory address buses, for data memory address buses.

This paper is organized as follows. Section 2 reviews some state-of-the-art address bus encoding techniques. In Sect. 3, data organization techniques for low-energy address buses are presented. In Sect. 4, our experimental results are reported. Finally, Section5 concludes this paper with a summary and future directions.

2. Address Bus Encoding Techniques

This section briefly reviews three state-of-the-art techniques which are effective for encoding address buses between processor and instruction memory, namely, *T0 Coding*, *TO-C Coding* and *Inc-Xor Coding*. All of these encoding techniques exploit the sequential access nature of instruction memory.

First, we introduce some notations used in the rest of this paper.

`b(t)`: Original address value to be sent at time `t`.
`B(t)`: Encoded value which is actually transfered on the bus at time `t`.
`S`: Stride value, i.e., the difference between consecutive addresses.

The *T0* code proposed by Benini et al. in [4], [5] has a redundant bit line, called `INC`. If accesses are sequential, the

Manuscript received August 29, 2003.
Manuscript revised December 24, 2003.

[†]The authors are with the Department of Information Engineering, the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464-8603 Japan.
[††]The author is with the Center of Embedded Computer Systems, University of California, Irvine.
a) E-mail: hiroyuki@acm.org

sender sets the INC line and keeps the other lines unchanged. Otherwise, the original address value $b(t)$ is sent on the bus and INC is de-asserted. In summary, the T0 encoder works as follows.

```
if (b(t) == b(t-1) + S) {
   B(t) = B(t-1);
   INC = 1;
} else {
   B(t) = b(t);
   INC = 0;
}
```

The *T0-Concise* code (or *T0-C* in short) developed by Aghaghiri et al. [7] is an irredundant code based on T0. The T0-C encoder works as follows.

```
if (b(t) == b(t-1) + S) {
   B(t) = B(t-1);
} else if (B(t-1) != b(t)) {
   B(t) = b(t);
} else {
   B(t) = b(t-1) + S;
}
```

The last case, i.e., `(b(t) != b(t-1) + 1) and (B(t-1) == b(t))`, can be considered as an exceptional case to the T0 code. In this case, if `b(t)` is sent on the bus, the receiver cannot judge whether the access is in-sequence or not, because `B(t) == B(t-1)` does hold and no bit line changes. In the T0-C code, `b(t-1) + S` is sent instead. Note that such a case rarely happens in practice. Therefore, T0-C is more efficient than T0 in terms of bus transitions due to its irredundancy.

In [6], Ramprasad et al. developed another T0-based irredundant code, called *Inc-Xor*[†]. The encoder works as follows.

```
B(t) = b(t) ⊕ (b(t-1) + S) ⊕ B(t-1)
```

Here, ⊕ denotes the Exclusive-Or (Xor) function. It is easily observed that no bit changes when the accesses are sequential.

The experimental results in [7] show that T0, T0-C and Inc-Xor achieve the average savings of bus activities by 62.0%, 73.1% and 75.0%, respectively. The experiments in [6] also demonstrate the effectiveness of Inc-Xor over T0 and Gray for instruction addresses. In [6], T0 and Inc-Xor were tested for data address streams, too, but the results were somewhat disappointing. In this paper, we demonstrate that T0-based codes are also effective for data address streams if memory data organization is optimized.

Another approach to low-energy address buses is designing an encoder and a decoder for a specific application. For example, the *Working-Zone* code [9] and the *Beach* solution [10] belong to this approach. These techniques are effective not only for instruction memory address buses but also for data memory address buses. However, designing application-specific encoders and decoders is often time-consuming, expensive, or even impossible.

In the rest of this paper, we assume T0-based coding techniques because of their efficiency and the simplicity of their encoder and decoder circuits.

3. Low-Energy Data Organization

In the last decade, memory data organization for low-energy embedded systems has been studied in the fields of high-level synthesis and code generation [12]–[14]. In [15], Panda et al. provided an extensive survey on this topic.

Many of the previous research efforts try to minimize accesses to energy-consuming memories (i.e., off-chip or large on-chip memories) by efficiently utilizing small on-chip memories such as caches or scratch-pad memories. These techniques are very effective for systems with memory hierarchy or with various types of memories. On the other hand, this paper does not assume any specific memory system architectures.

In the field of high-level synthesis, Panda and Dutt studied memory data organization strategies for minimizing switching activity on address buses [16]. Our work is very similar to theirs, but is different in several ways. The differences come from the fact that this paper addresses code generation while their work is for high-level synthesis. For example, in [16], they proposed a new data organization technique based on a *tile*, which requires a complicated address calculation. In hardware synthesis, it is possible to generate an address calculation unit[††] and place it before the bus encoder. On the other hand, that is impossible in code generation. Another difference is that this paper studies the effectiveness of T0-based coding techniques for data memory address buses. Moreover, we also study the impacts of data organization techniques for low-energy address buses on data cache performance[†††].

3.1 Data Organization for Low-Energy Address Buses

In this paper, we focus on how to place data arrays in memory. Scalar variables are assumed to be kept in registers. Our goal is to find an array organization in memory such that sequential data accesses are maximized.

Let us use a simple example code shown in Fig. 1(a) where two two-dimensional arrays exist. There are typically two forms for placing a multi-dimensional array in memory, i.e., *row-major* (row-by-row) and *column-major* (column-by-column) [11], and most of existing C compilers employ the row-major form, regardless of how the array is accessed. For multiple arrays, most compilers place them in the same order as their declaration. As a result, arrays `a` and `b` in Fig. 1(a) are organized in memory as shown in Fig. 1(b). In

[†]In [8], the *Inc-Xor* coding is referred as *T0-Xor*.

[††]The organization of the address calculation unit depends on the sizes of tiles.

[†††]It should be noted again that cache is not necessary in our work. However, we study cache performance in case it exists.

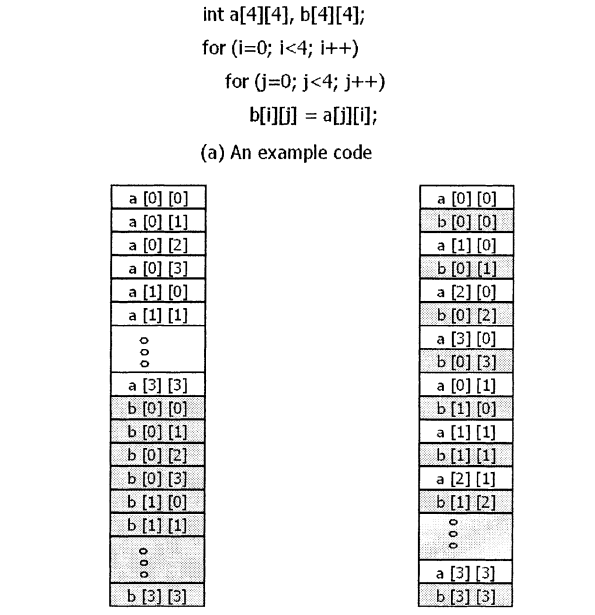

Fig. 1 A simple example code with two two-dimensional arrays.

this case, no memory access is sequential.

When we focus on array a alone, we can see that elements in a column are sequentially accessed at the inner loop. This brings the idea of placing array a in a column-major form. Still, no memory access is sequential because arrays a and b are alternately accessed. Several ways can be considered in order to serialize the accesses. One way is to patially unroll the loop and access more than one array element at each iteration. However, loop unrolling generally leads to an increase in code size, which is often unacceptable in embedded system design. Another way which is more efficient is to place the two arrays in an interleaved manner [16][†]. Figure 1(c) shows the optimal data organization where array a is of a column-major form, b is of a row-major form, and a and b are interleaved. With this data organization, all the memory accesses except the very first one become sequential.

3.2 Integrated Memory Access Scheduling

Next, we show that optimizing data organization alone has a very limited effect on the switching activity on address buses but combining it with memory access scheduling significantly reduces the switching activity.

Let us consider the SOR (Successive Over-Relaxation) algorithm shown in Fig. 2(a). The access pattern in the inner loop is illustrated in Fig. 2(b). Here, no access is sequential even if the arrays are placed in an interleaved manner. However, if we reorder the memory accesses as shown in Fig. 2(c) and organize arrays a, b, c, d, e, and f in an interleaved form, eight accesses out of twelve become sequential. It should be noted that applying scheduling alone is not effective enough. Thus, the example in Fig. 2 indicates the

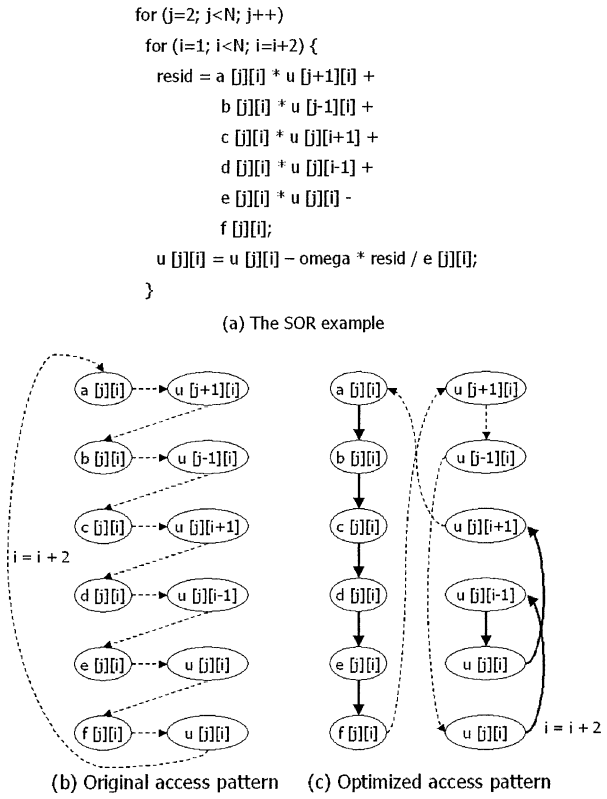

Fig. 2 An example of memory access scheduling. Broken edges indicate non-sequential accesses while solid edges are sequential ones.

importance of applying both data organization optimization and scheduling.

4. Experiments

To demonstrate the effectiveness of data organization optimization, we conducted a set of experiments. The SimpleScalar simulator and GNU C Compiler 2.7.2.3 were used as a platform of our experiments. Seven DSP kernels from [19], which are often used in image processing applications, were used as benchmark programs. Table 1 summarizes characteristics of the programs. For each program, the table shows the number and dimensions of data arrays, the type of the array elements, the depth of loop nests, the number of data memory accesses in the innermost loop, the total number of data memory accesses, and the code size in terms of the number of instructions[††]. Through the experiments, we also tested the effectiveness of three T0-based bus encoding techniques, i.e., the original T0, T0-C, and Inc-Xor.

Figure 3 shows the experimental results where the number of transitions on the address bus is presented for each combination of the benchmark programs and the bus encoding techniques. For each program, the results are normalized where the baseline is the combination of unopti-

[†]In [17], the interleaved placement is used for cache conflict minimization.

[††]Instructions for initialization and termination of the program are included.

Table 1 DSP kernels used as benchmark programs.

	Description	Arrays	Dimen-sions	Array element type	Loop nests	Data accesses in the inner-most loop	Total data accesses	Code size (inst.)
`compress`	Image compression	1	2	int	2	5	53,685	2,880
`gsr`	Gauss-seidal relaxation	2	2	double	3	6	118,938	2,918
`laplace`	Laplace edge enhancement algorithm	2	2	int	2	10	103,687	2,900
`linear`	Linear recurrence solver	3	1	double	2	4	1,215,690	2,892
`lowpass`	Lowpass filter	1	2	float	2	9	133,711	2,920
`sor`	Successive over relaxation	7	2	float	2	12	62,490	2,934
`wavelet`	Debaucles 4-coefficient wavelet filter	2	1	float	1	6	3,945	2,940

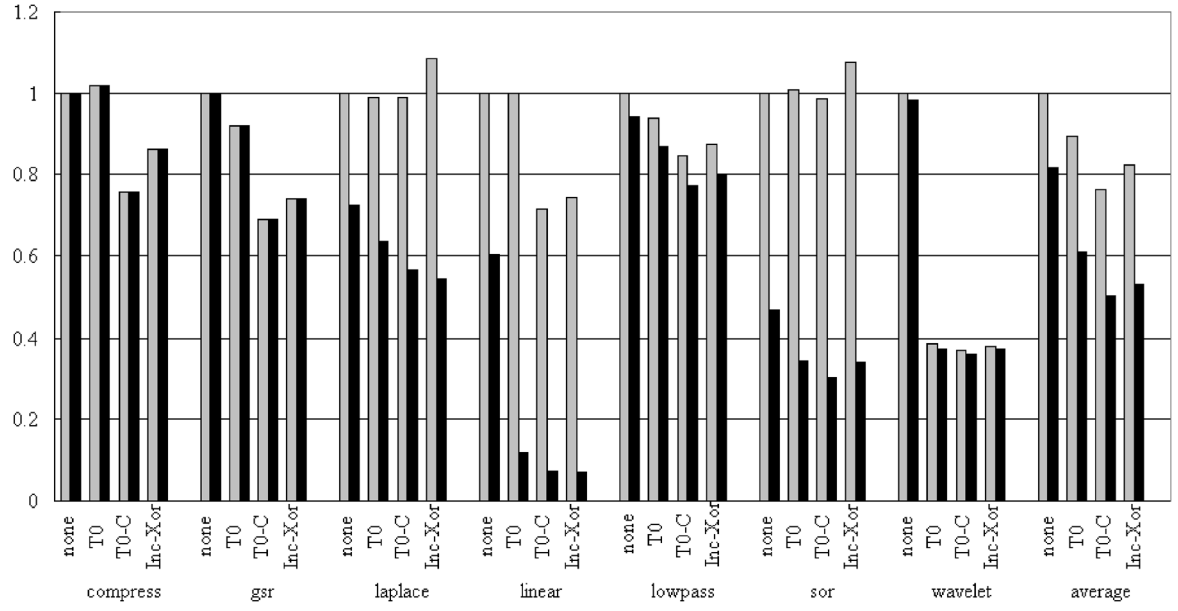

Fig. 3 The number of transitions on the address bus. Gray bars denote unoptimized data organization, while black bars denote optimized one.

mized data organization and no bus encoding. Gray bars show the cases data organization is not optimized. Even in these cases, conventional compiler optimizations were performed by giving the ''-O2'' option when running the compiler. This option sometimes reorders memory accesses but note that the objective is not energy optimization of course. Black bars in Fig. 3 show the cases both data organization optimization and memory access scheduling were performed. For each program, we explored all the possible data organizations, and then performed memory access scheduling for each data organization. An exhaustive search algorithm was used for memory access scheduling. Then, we selected the best combination of data organization and schedule in terms of the number of sequential accesses. Loop unrolling was not applied. The data organization exploration process was done by hand, and its efficient automation is remained as one of our future work.

In our experiments, T0-C was the best encoding technique. Without data organization exploration, the switching activity on the address bus was reduced by 24% on an average with the T0-C code. When data organization was optimized, the bus activity was further reduced. The average saving was 50%. Inc-Xor also achieves a high reduction in bus activity, but slightly worse than T0-C.

Table 2 Percentages of sequential accesses.

	unoptimized	optimized
`compress`	43.2%	43.2%
`gsr`	59.2%	59.2%
`laplace`	3.1%	41.6%
`linear`	50.1%	91.6%
`lowpass`	32.3%	39.8%
`sor`	12.9%	67.8%
`wavelet`	84.4%	84.7%
`average`	40.7%	61.1%

Table 2 shows the percentages of sequential accesses with and without data organization optimization. Since the size of array elements in `gsr` and `linear` was double words, about a half of memory accesses were sequential without optimization. Data organization optimization significantly increases sequenatial accesses for `laplace`, `linear` and `sor`, thus reduces bus transitions. For `compress`, `gsr` and `wavelet`, many memory accesses were sequenatial even without data organization optimization, and there was no opportunity for further serialization.

An interesting observation is that even if the bus was

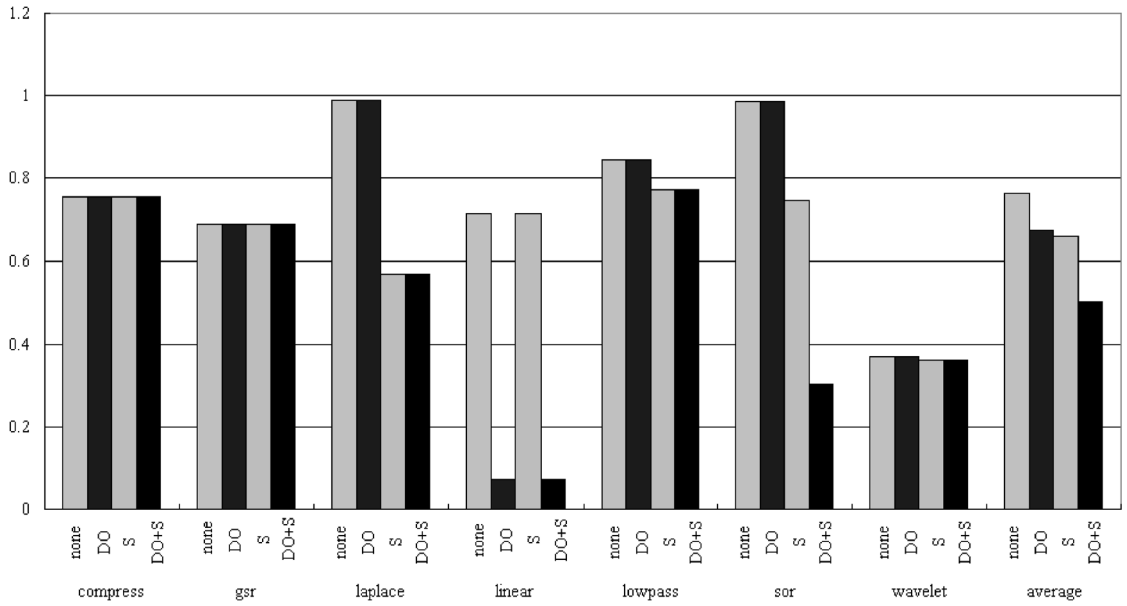

Fig. 4 Effects of individual techniques on address bus transitions. T0-C is used for bus encoding.

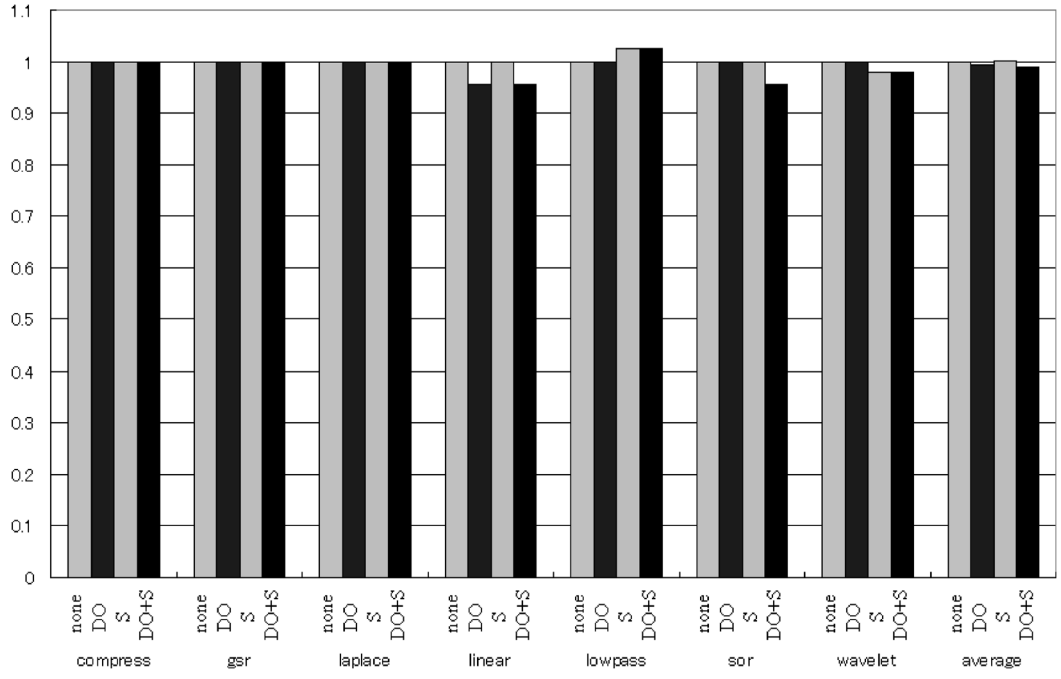

Fig. 5 The number of data cache misses. 1K byte, direct mapped cache is used.

not encoded, its activity was reduced by up to 53% (18% on an average). This demonstrates that the bus energy can be save without introducing encoder and decoder circuits, thus no hardware overhead is necessary.

Next, we investigated the impacts of data organization optimization and memory access scheduling individually. The results are presented in Fig. 4. In the graph, `DO`, `S`, and `DO+S` mean data organization optimization, scheduling, and combination of both, respectively. These results demonstrate the importance of applying both of data organization optimization and scheduling.

If the memory system has caches, changing memory data organization or reordering memory accesses may change the performance of the cache. The objective of data organization and memory access scheduling is to serialize data accesses, and the serialization can be considered as a specific type of spatial localization. Therefore, it is expected that by applying data organization optimization and scheduling the cache performance is not degraded (hopefully improved). In order to confirm this, we ran cache sim-

ulation and counted the number of cache misses. The results are shown in Fig. 5. The number of cache misses was slightly decreased for three programs out of seven, but was slightly increased for a program. In conclusion, no remarkable change was seen in the cache performance.

5. Conclusion

This paper addressed low-energy code generation for media and DSP applications. We demonstrated that data organization in memory has large impacts on the energy consumption of address buses. The experimental results showed that the bus activity was reduced by 50% through data organization exploration and T0-based bus encoding.

In future, we will develop an efficient algorithm for data organization optimization. Then, we will apply it to real-world media applications.

Acknowledgement

This work was partially supported by NSF grants CCR-0203813 and CCR-0205712.

References

[1] M.R. Stan and W.P. Burleson, "Bus-invert coding for low power I/O," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.3, no.1, pp.49–58, 1995.
[2] Y. Shin, S.-I. Chae, and K. Choi, "Partial bus-invert coding for power optimization of application-specific systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.9, no.2, pp.377–383, 2001.
[3] C.L. Su, C.Y. Tsui, and A.M. Despain, "Saving power in the control path of embedded processors," IEEE Des. Test Comput., vol.11, no.4, pp.24–31, 1994.
[4] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic zero-transition activity encoding for address buses in low-power microprocessor-based systems," Proc. GLS-VLSI, pp.77–82, 1997.
[5] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Address bus encoding techniques for system-level power optimization," Proc. DATE, pp.861–866, 1998.
[6] S. Ramprasad, N.R. Shanbhag, and I.N. Hajj, "A coding framework for low-power address and data busses," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.7, no.2, pp.212–221, 1999.
[7] Y. Aghaghiri, F. Fallah, and M. Pedram, "Irredundant address bus encoding for low power," Proc. ISLPED, pp.182–187, 2001.
[8] W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano, "Power optimization of system level buses based on software profiling," Proc. CODES, pp.29–33, 2000.
[9] E. Musoll, T. Lang, and J. Cortadella, "Working-zone encoding for reducing the energy in microprocessor address buses," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.6, no.4, pp.568–572, 1998.
[10] L. Benini, G. De Micheli, E. Macii, M. Poncino, and S. Quer, "Power optimization of core-based systems by address bus encoding," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.6, no.4, pp.554–562, 1998.
[11] A.V. Aho, R. Sethi, and J.D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986.
[12] F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele, and A. Vandecappelle, Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design, Kluwer Academic Publishers, 1998.
[13] F. Catthoor, K. Danckaert, C. Kulkarni, E. Brockmeyer, P.G. Kjeldsberg, T. Van Achteren, and T. Omnes, Data Access and Storage Management for Embedded Programmable Processors, Kluwer Academic Publishers, 2002.
[14] P.R. Panda, N.D. Dutt, and A. Nicolau, Memory Issues in Embedded Systems-On-Chip Optimizations and Exploration, Kluwer Academic Publishers, 1998.
[15] P.R. Panda, F. Catthoor, N.D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A. Vandercappelle, and P.G. Kjeldsberg, "Data and memory optimization techniques for embedded systems," ACM TODAES, vol.6, no.2, pp.149–206, 2001.
[16] P.R. Panda and N.D. Dutt, "Low-power memory mapping through reducing address bus activity," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.7, no.3, pp.309–320, 1999.
[17] C. Kulkarni, F. Catthoor, and H. De Man, "Advanced data layout organization for multi-media applications," Proc. PDIVM, pp.186–193, 2000.
[18] SimpleScalar Tools, http://www.simplescalar.com/
[19] P.R. Panda and N.D. Dutt, "1995 High level synthesis design repository," Proc. ISSS, pp.170–174, 1995.

Hiroyuki Tomiyama received his Ph.D. degree in computer science from Kyushu University in 1999. From 1999 to 2001, he was a visiting postdoctoral researcher with the Center of Embedded Computer Systems, University of California, Irvine, where he was financially supported by JSPS Postdoctoral Fellowship for Research Abroad. From 2001 to 2003, he was a researcher at the Institute of Systems & Information Technologies/KYUSHU. He is currently an assistant professor with the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. His research interests include system-level design methodologies for embedded systems, computer-aided design of VLSI systems, and compilers for embedded systems. He has served on the organizing and program committees of several premier conferences including ASP-DAC and ICCAD. He is a member of ACM, IEEE, and IPSJ.

Hiroaki Takada is a Professor at the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. He received his Ph.D. degree in Information Sciece from University of Tokyo in 1996. He was a Research Associate at University of Tokyo from 1989 to 1997, and was an Assistant Professor and then an Associate Professor at Toyohashi Univesrity of Technology from 1997 to 2003. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of ACM, IEEE, IPSJ, and JSSST.

Nikil D. Dutt received a B.E. (Hons) in Mechanical Engineering from the Birla Institute of Technology and Science, Pilani, India in 1980, an M.S. in Computer Science from the Pennsylvania State University in 1983, and a Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 1989. He is currently Professor of CS and EECS at the University of California, Irvine and is also affiliated with the following Centers at UCI: Center for Embedded Computer Systems (CECS), California Institute for Telecommunications and Information Technology (Cal-(IT)2), and the Center for Pervasive Communications and Computing (CPCC). His research interests are in embedded systems design automation, computer architecture, optimizing compilers, system specification techniques, and distributed embedded systems. He is a coauthor of three books: "High-Level Synthesis: Introduction to Chip and System Design," Kluwer Academic Publishers, 1992, "Memory Issues in Embedded Systems-on-Chip: Optimizations and Exploration," Kluwer Academic Publishers, 1999, and "Memory Architecture Exploration for Programmable Embedded Systems," Kluwer Academic Publishers, 2003. He received best paper awards at CHDL89, CHDL91, VLSIDesign2003 and CODES+ISSS 2003. Professor Dutt currently serves as Associate Editor of ACM Transactions on Design Automation of Electronic Systems (TODAES) and as Associate Editor of ACM Transactions on Embedded Computer Systems (TECS). He was an ACM SIGDA Distinguished Lecturer during 2001–2002, and is currently an IEEE Computer Society Distinguished Visitor for 2003–2005. He has served on the steering, organizing, and program committees of several premier CAD and Embedded System Design conferences, including ASPDAC, DATE, ICCAD, CODES/ISSS, CASES, ISLPED and LCTES. He is a senior member of the IEEE, serves on the advisory boards of ACM SIGBED and ACM SIGDA, and is Vice-Chair of IFIP WG 10.5.