A Power Reduction Scheme for Data Buses by Dynamic Detection of Active Bits

Muroyama, Masanori
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering at Kyushu University

Hyodo, Akihiko
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering at Kyushu University

Okuma, Takanori
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering at Kyushu University

Yasuura, Hiroto
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering at Kyushu University

出版情報：Proceedings of Euromicro Symposium on Digital System Design -Architectures, Methods and Tools-(DSD2003), pp.408-415, 2003-09. IEEE Computer Society
バージョン：
権利関係：

A Power Reduction Scheme for Data Buses by Dynamic Detection of Active Bits

Masanori Muroyama, Akihiko Hyodo, Takanori Okuma and Hiroto Yasuura
Department of Computer Science and Communication Engineering
Graduate School of Information Science and Electrical Engineering at Kyushu University
6-1 Kasuga-Koen, Kasuga, Fukuoka 816-8580, Japan
{muroyama,akihiko,okuma,yasuura}@c.csce.kyushu-u.ac.jp

Abstract

To transfer a small number, we inherently need the small number of bits. But all bit lines on a data bus change their status and redundant power is consumed. To reduce the redundant power consumption, we introduce a concept named active bits. In this paper, we propose a power reduction scheme for data buses using the active bits. Suppressing switching activity of inactive bits, we can reduce redundant power consumption.

We propose various power reduction techniques using active bits and the implementation methods. Experimental results illustrate 20% - 35% on average and up to 54.2% switching activity reduction.

1. Introduction

Recently, many portable devices have been made low power CMOS circuit design. In these devices low power is an important and necessary issue in order to achieve long battery life, low cost for package, high reliability and so on. A digital system such as processor almost has datapath and controller in many cases. The datapath is the largest power consuming component in a system, because buses, memories and arithmetic circuits consume a great deal of power.

Datapath width, the bit width of buses and operational units such as arithmetic circuits in a system, is an important design parameter for power optimization. The datapath width can be optimized for power minimization by means of analyzing required bit-width of variables [12]. The technique is a static approach, because the optimization is done in design phase. In this paper, we propose a difference approach that is dynamic approach applicable in run time.

To transfer a small number, we inherently need the small number of bits. But all bit lines on a data bus change their status and redundant power is consumed. We define *active bits* as necessary bits of the data for computation, and *inactive bits* as remaining bits excepting active bits in the data. For example, the unsigned decimal number 1000 needs 10 bits in binary representation. In a 32 bits system, lower 10 bits are active bits and upper 22 bits are inactive ones.

Some techniques using similar concepts are studied. Okuma et al. [8] presented a memory power reduction technique using active bits. They can achieve significant energy reduction compared to the monolithic memory, 52.2% and 84.2% for JPEG and MPEG-2, respectively. An arithmetic circuit power reduction technique is also introduced by Brooks and Martonosi [4]. They propose a method to reduce power in the integer execution unit of a processor with aggressive clock gating inactive bits. Consequently, the method reduces 45% - 60% power reduction for the integer unit.

The power consumption in a CMOS circuit can be classified as static power consumption and dynamic power consumption. In this paper, static power is ignored and only dynamic power is considered. The dynamic power consumed by a CMOS circuit can be defined as follows:

$$P_{chip} \propto V_{DD}^2 \cdot f \cdot \sum_{i=1}^{N} \cdot CL_i \cdot \alpha_i$$

where P_{chip}:total power consumption of a chip, N:total number of nodes of the circuit on the chip, f:the clock frequency, CL_i: the load capacitance at node i, V_{DD}:the power supply voltage and α_i:the activity factor at node i. It is note that capacitances for long buses are often significantly higher than the capacitances for gates.

Lowering the activity factor is a very promising way of decreasing the power consumption on buses. So we are now focusing on reduction of α_i. Several signal coding schemes have been already proposed to minimize transition activity on buses. When statistical properties are unknown a priori, the bus-invert technique [11] and the on-line adaptive

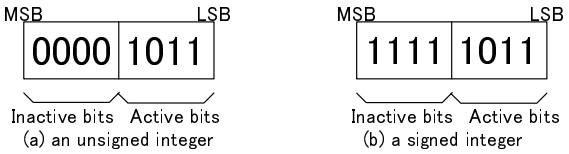

Figure 1. An example of active bits

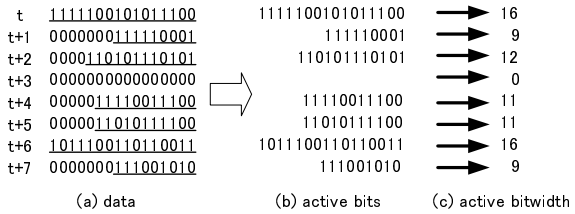

Figure 2. An example of active bits in data sequences(unsigned integers)

scheme [2] can be applied to coding randomly distributed signals. The concepts of these methods are not available in a whole system. In address buses, highly correlated data patterns exhibit a spatio-temporal locality, so Panda et al. [9] exploited the characteristics for energy reduction. Similar approaches are proposed using Gray code addressing [6], the T0 method [3], and the working-zone coding [7]. But these methods do not efficiently work for a bus on which distribution of the data change dynamically. In this paper, we propose a power reduction scheme for data buses, on which characteristics of the data distribution are often changed and hard to expect. In our scheme, the active bits are detected on the fly, and suppressing the wasteful switching activity of the inactive bits.

This paper is structured as follows: Section 2 introduces a concept of active bits. In section 3 we propose low power bus techniques using active bits, and show the power-optimized implementation of the encoder-decoder system. Experimental results are given in section 4. We conclude in section 5.

2. Active Bits

There are many representations for data; an unsigned integer form, a signed integer representation, a fixed point digit one and a floating point digit one. We can define active bits for each representation. In this paper we assume that all data are represented as the unsigned and signed integers.

Now we define inactive bits and active bits. For the unsigned integers, inactive bits are continuous string of 0's from the most significant bit. The remaining part is defined as active bits (Fig. 1 (a)). For the signed integers, sign extension is also inactive bits (Fig. 1 (b)).

We call the number of active bits as *active bitwidth*. In particular, we make active bitwidth zero when value of data is 0 (all bits are zero). Fig. 2 shows an example of active bits and active bitwidth in data sequences in the case of unsigned integer representation. The underlines (Fig. 2 (a)) indicate active bits. The remaining bits after deleting the continuous 0's (Fig. 2 (b)) and the length of active bits are active bits and active bitwidth (Fig. 2 (c)), respectively.

Fig. 3 shows an example of occurrence each active bitwidth in a 32-bit datapath width system. An mpeg2play program with a 245k byte picture is adopted as a sample

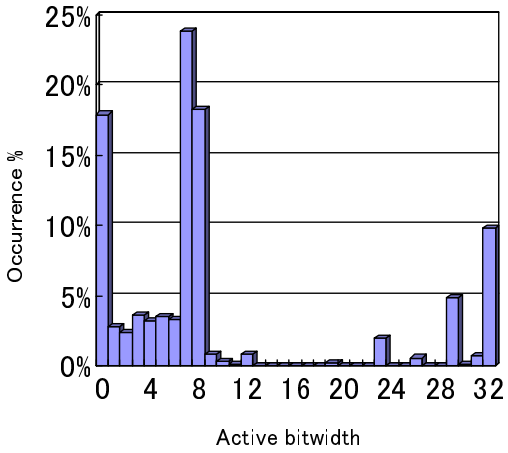

Figure 3. Active bitwidth on 32-bit data bus (app.: mpeg2play,data size: 245k byte)

application. In the figure 3, 80% of data on the data bus are 8 or less active bitwidth. When active bits only are activate on the bus, the bus activity is 30.0%.

At the point of compilation, ways of access to memories are determined by instructions. There are three access types; word access (per 32 bits), half-word access (per 16 bits), byte access (per 8 bits). 28%, 27% and 45% of all access is word, half-word and byte access respectively. If the bus is activated by access unit, the bus activity is 52.8%. In this case, the part including many inactive bits is also activated. Therefore it is important to detect active bits and inactive bits dynamically.

3. Detection and Coding Mechanisms

3.1. Overview

All information is included in active bits. This means that values of inactive bits can be ignored. Since decreasing α_i, in other words increasing unchanged signal values,

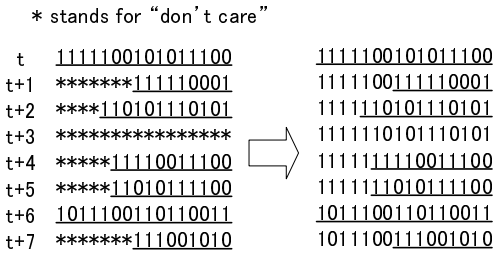

Figure 4. An example of encoding

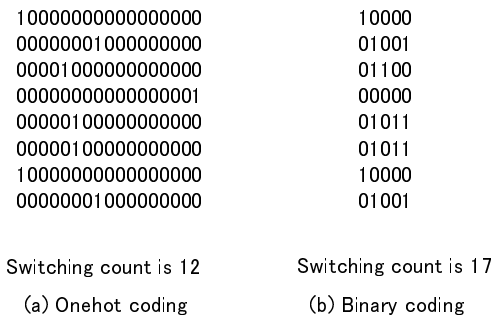

Figure 5. An example of extra bits

induce power reduction. Therefore it is possible to reduce power consumption by using a technique that signal values of inactive bits at time t hold previous values at time t-1. Figure 4 presents the result of applying the technique to the example of Figure 2. The underlines and don't cares indicate active bits and inactive bits, respectively. Total switching count is 51 in the original data sequences, while total switching count is 29 in the encoded data sequences. When active bitwidth of previous original data is large and active bitwidth of current original data is small (in other words, the difference of active bitwidth of two consecutive data is large), the switching reduction effect is more higher. In the case of the mpeg2play program with the data streaming, the difference of active bitwidth of two consecutive data is often large.

Extra bits are required when a receiver knows active bits in the data. We consider two techniques to indicate active bitwidth: *onehot coding* and *binary coding* (Figure 5). Onehot coding is one by which a bit is used to indicate a state (an active bitwidth). Using binary coding, whole states are encoded in binary form.

In an N-bit bus, there are N states to indicate active bitwidth. The number of extra bits with onehot coding is N+1. Binary coding requires $\lfloor \log_2 (N+1) \rfloor$ extra bits. In the both cases, a state of active bitwidth zero should be considered. Next, we introduce extra bits reduction techniques.

3.2. Granularity Control in Active Bitwidth

Increase of extra bits causes increase of area complexity and power consumption of extra bits. Hence it is important to reduce extra bits. It is effective to reduce the number of states to indicate active bits. Firstly, we inquire into the relation between the number of states and switching count of extra bits.

We encode γ states by using onehot and binary coding. We assume that γ is power of 2 ($= 1, 2, 4, 8, ...$) and the probability of occurrence of each state is equal.

Binary coding requires $log_2\gamma$ bits. The probability of occurrence of 1 (or 0) of each bit is 1/2, because the probability of occurrence of each state is equal. The probability of switching of a bit is 1/2. We define E_{bin} as expected value of switching count of extra bits by using binary coding. In the case, E_{bin} is $(log_2\gamma)/2$.

Onehot coding requires γ bits. The probability of occurrence of 1 of each bit is $1/\gamma$. The probability of of switching of a bit is $(2\gamma - 2)/\gamma^2$. We define E_{one} as expected value of switching count of extra bits by using onehot coding. In the case, E_{one} is $(2\gamma - 2)/\gamma$.

Assume $\gamma = 16$, for binary coding the number of extra bits is 4 and $E_{bin} = 2.00$. For onehot coding, the number of extra bits is 16 and $E_{one} = 1.88$. In the case of γ = 8, for binary coding extra bits is 3 and $E_{bin} = 1.50$. For onehot coding the number of extra bits is 8 and $E_{one} = 1.75$. For low power, binary coding is superior to onehot coding when the number of states is 8 or less (remember γ is power of 2). Onehot coding is better than binary coding when the number of states is over 8. In all cases, the number of extra bits for onehot coding are larger than binary ones.

Secondly, we introduce a technique for reducing the number of states. A bus are divided into multiple parts. After division, each partition is called a segment. We define variables as follows:

- N_{seg}: the number of segments
- b_i: the bitwidth of segment i $(b_i \geq 1)$

The variables must satisfy the formula (1).

$$\sum_{i=1}^{N_{seg}} b_i = N \tag{1}$$

where N is the datapath width (equal to data bus width). In this subsection, we consider that all b_i are equal to N/N_{seg} (subsection 3.4 introduces a case that different bitwidth of each segment is permitted). We define the active segments as segments including one or more active bits. To simplify discuss, the representation of data is an unsigned one only. We call segments including active bits active segments. Remaining segments are inactive segments. The number of active segments is defined as active segmentwidth.

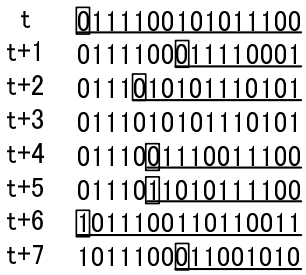

Figure 6. An example of embedding approach

The number of active segments plus one (one bit is used for active segmentwidth 0) is the number of states (γ) to be encoded by using onehot and binary coding. If N_{seg} is small (small states), the small number of extra bits and small switching count of extra bits is achieved generally. To reduce the number of states accomplishes switching count and the number of extra bits reduction. But it reduces the advantage of switching count reduction on data, because inactive bits in active segments increase.

3.3. Embedding Approach

This paragraph introduces a coding using active bits without extra bits. The reason why extra bits are not used is that information of active bitwidth is embedded in the data on the original bus. This encoding consists of three stages. At first stage, the current values in active bits on the bus are held. At second stage, the current value of the most significant active bit on the bus is inverted. Finally, the original data of the remaining active bits are added.

In a reciever, comparing previous and current bits from the most significant bit to the least significant bit between two data, a first bit which the two values of two bits are different indicates a start position of active bits. We call this technique an *embedding* approach. Using the number of the same bits from the most significant bit between two data (we define as *s*), active bitwidth can be obtained (N-s).

Figure 6 shows an example of the embedding approach. In the figure, underlines and squares indicate active bits and start positions of the most significant of active bits, respectively. Total switching count is 37 in the data sequences. It should be noted that its decoder cost to hold previous values on the bus is somewhat large (more detail in section 3.5).

3.4. Application-Specific Granularity Control in Active Bitwidth

This section presents a technique that the detection granularity of active bits is tailored to a particular application. It means that b_i can not be constant. Initially, we define the probability of appearances of active bitwidth i as $P_{act}(i)$. We assume that the probability of each active bitwidth is given. This application-specific information can be provided from simulation in advance. In the final, a bus is partitioned into M segments with optimized b_i.

In initial condition, we consider finest granularity detection that $\forall i, b_i = 1$. Next we combine two segments, which are continuous ones. Combining segments makes the number of segments and extra bits decrease. We don't combine three or more segments in the single operation because of avoiding computational complexity (this approach is heuristic approach). In this paper, an onehot coding technique is adopted to encode active segmentwidth. Application-specific detection granularity control by using a binary one is future work.

We deal with two continuous segments; a lower segment is segment i, a upper segment is segment i+1. To combine the segments results in changing switching count. The increasing and decreasing of switching in the original bus and extra bits are considered separately. We define the expected value of the increase and decrease of the switching count in the original and extra parts as $\Delta E_{bus}, \Delta E_{extra}$, respectively. In the original bus, switching count increase the only one case, when the segment i is active segment and the segment i+1 is inactive segment, respectively. ΔE_{bus} is defined as follows:

$$\Delta E_{bus} = (b_{i+1}/2) \cdot P_{act}(i) \cdot PT$$

where PT is the sum of data accesses on the bus.

The number of increase and decrease of switching in extra bits is defined as follows:

$$\Delta E_{extra} = -2P_{act}(i) \cdot P_{act}(i+1) \cdot PT.$$

Therefore the sum of increase of switching count ΔE_{total} is

$$\Delta E_{total} = P_{act}(i) \cdot (b_{i+1}/2 - 2P_{act}(i+1)) \cdot PT.$$

Computing the equation ΔE_{total}, a pair of two segments which ΔE_{total} is the minimum is combined. After that, the probability of occurrence of the segment is calculated by adding the probability of occurrence of each ones. This operation repeats until the sum of segments is M. We summarize this procedure in Figure 7. The information of $\forall i, P_{act}(i)$ is given in advance by tracing the data. The values of b each segment are outputed by this procedure.

3.5. Implementation

In this subsection we evaluate the encoder and decoder circuits to implement our techniques. The encoder circuit consists of two parts, a circuit to detect active bitwidth and a

```
Input: P_act(i), for all i
Output: optimized segment width, for all segments
Procedure Application-Specific Detection
   while until the number of segments is M
      forall pairs of continuous two segments
         compute delta E_total by using P_act(i)
      end forall
   (1) select a pair whose delta E_total is minimum
   (2) two segments of the pair are combined
   (3) the combined segments are dealt with a segment
   (4) compute the probability of occurrence of
      each active segments
   end while
end Procedure
```

Figure 7. Procedure for application-specific granularity control

circuit to generate encoded data on the basis of information of active bitwidth.

We designed the encoders and decoders with verilog-HDL, the circuits are synthesized by design compiler (Synopsys corp.) under constraint of power optimizing by using process technology of HITACHI 0.18μm. Table 1 and 2 report the evaluation of synthesized circuits.

We compared the Bus-Invert (BI) technique with ours. The BI technique is one of the most popular coding techniques. The BI and our techniques are well suited to a data bus. We evaluated the encoders and decoders of our techniques and BI with random input patterns. In the encoders, the BI consumes power about three times more than the others. The encoder circuit of BI uses plural FAs and XORs as basic cells. Power consumption of these cells is somewhat large. In the decoders, the circuit for the embedding approach consumes more power. The reason is that there are many latches to hold previous values in the decoder of the embedding approach.

If a whole system is designed by using active bit, it is possible to share an active bitwidth detector of each component of the system. Since an encoded data holds original information in active bits, it is easy to pick out original informaion by using an active bitwidth each data. Therefore overhead can be reduced when plural components in a whole system are designed by the concept based on active bit. For example, assume that in a data bus and an arithmetic circuit parts the techniques using active bits are used. When an encoded data using our codings is transfered from the data bus to inputs of the arithmetic circuit without decoding the data, an decoder in the bus and an active bit detector in the arithmetic circuit are not needed.

When only an encoder ($N_{seg} = 32(unsigned)$) is used without an active bitwidth detector and a decoder in the bus part, delay penalty is 3.18 $n\ sec$. Hence we can save on hardware cost by using techniques by Okuma et al. and Brooks et al. together. However for low power systems, we must design power efficient circuits of encoder-decoder to implement our scheme, especially embedding approach because of larger hardware penalty than others.

4. Experimental Results

We demonstrate experimental results by using our proposed techniques and BI one. The SimpleScalar [1], which is the processor simulator, is utilized as a tool to obtain data on the bus. This model has a 32-bits data bus. We select the mpeg2play program and several SPEC2000 benchmarks as experimental application programs. For data dependency analysis, three images are used for the mpeg2play program: 116k byte, 245k byte and 649k byte images. We first trace data sequences on the data bus when memory accesses are occurred. After that, we apply our techniques and the BI technique to the data sequences. In the BI coding, if the Hamming distance between the present data and the last data of the bus is lager than N/2, the present data is transmitted with each bit inverted. An extra bit, called invert line, is required to signal the receiver side whether the bus is inverted or not.

Table 3 shows the results of switching count reduction by various techniques. Five techniques are used to encode data sequences of each benchmark. The five techniques are as follows:

- Finest granularity detection (N_{seg} = 32) for unsigned integers (u)
- Finest granularity detection (N_{seg} = 32) for signed integers (s)
- Embedding approach
- operand: word, half-word, byte access
- BI (Bus-Invert) technique

In the operand case, the number of active bits of each memory access type(word,half-word,byte) is 32, 16 and 8, respectively.

In the table 3, all results contain the switching count of extra bits. Power saving from 20% to 34% can be achieved by using our techniques. In the best case, 54.2% of total switching count is reduced. Most cases using our techniques are better than the operand and BI cases. In the case of the operand, since the processor almost accesses the memory by the word access type, few switching count is reduced. The BI technique has no effect in the case that Hamming distance do not change drastically. Therefore switching count of some applications are not reduced. We cannot observe data dependency for the mpeg2play program.

Table 1. Evaluation of Encoders

	our approach			
	$N_{seg} = 32(unsigned)$	$N_{seg} = 32(signed)$	embedding	BI
extra bits	32	32	0	1
Cells	196	261	279	288
Area (μm^2)	10775	14100	12157	17625
Power(μW)	377	593	452	1560
Delay($n\ sec$)	6.63	9.89	8.78	5.91

Table 2. Evaluation of Decoders

	our approach			
	$N_{seg} = 32(unsigned)$	$N_{seg} = 32(signed)$	embedding	BI
extra bits	32	32	0	1
Cells	80	124	255	32
Area (μm^2)	2865	4369	12995	1966
Power(μW)	177	281	665	215
Delay($n\ sec$)	4.25	4.79	7.72	0.19

The experimental results of granularity control are introduced in table 4. In the case of onehot coding for active bitwidth, high swiching count saving is achieved by fine granularity segmetation. Contrary, using rough granularity segmentation in the case of binary coding for active bitwidth, high switching count reduction is achieved. Bold types in the table 4 indicate the most efficient granularity efficient cases each application. In the rough granularity cases, large switching count of extra bits is reduced. In consequent, total switching count is more reduced than the fine granularity cases.

Table 5 and Figure 8 show the results using the application-specific granularity control technique. We adopted the mpeg2play program with a 245k byte image. In the figure 8, we can see that each segmentwidth is differenct after applying the application-specific granularity control. High switching count saving is achieved in the case of $N_{seg} = 2$. If you can know an application apriori, the technique introduced in 3.4 is efficient.

We discuss the total power consumption of the bus, which include the extra circuits. The coding system consists of three parts; encoder, decoder, bus itself. Here the mpeg2play with a 245k byte image is used for this experiment. Figure 9 shows the comparison of total power consumption by using the all coding techniques. The X axis is a capacitance scale, which is proportion to wire length on a bus. In short bus, the N_{seg} = 32 (an unsigned representation) is the most low power coding, The embedding approach is the most effective coding in long bus. In all cases, our proposed techniques are more power-efficiency than the BI coding. But this coding's delay penalty is the most largest. When delay penalty in a system is severe, the finest granularity coding $N_{seg} = 32$ is more better. However this coding needs a lots extra bits. If the margin for extra bits (for example, I/O pins and so on) is small, the granularity control approach is reasonable. If other system components are made by the concept of active bits, the power consumption in the whole system will be more lower because of the decrease of overhead.

Table 5. Comparison of switching count reduction ratio using application-specific detection approach (%)

	N_{seg}				
	32	16	8	4	2
fixed $b_i(= N/N_{seg})$	45.6	44.1	42.7	40.3	27.0
variable b_i	45.6	44.1	44.1	43.6	38.8

5. Conclusions

This paper presented a novel low power bus scheme based on reducing switching activity using dynamic active bits detection. Suppressing switching activity of inactive bits, we can reduce redundant power consumption. Our scheme is more effective when active bitwidth in an application is very changeful.

Table 3. Comparison of switching count reduction

App.	unencoded	reduction(%)				
	#switching	$N_{seg} = 32$(u)	$N_{seg} = 32$(s)	embedding	operand	BI
mpeg2play						
(116k)	116570540	37.1	40.1	38.7	21.1	41.2
(245k)	196764229	45.6	49.0	49.6	32.5	48.0
(649k)	578222755	34.4	37.0	34.4	17.9	38.0
go	812346509	33.8	33.8	54.2	0.00	0.00
gzip	1054170245	10.5	10.2	16.6	-3.08	10.8
bzip2	8858426	17.8	17.8	36.2	26.6	0.00
perl	1828596	14.5	20.4	24.1	0.00	11.2
mesa	1744339	19.0	19.0	31.5	-0.13	1.11
wupwise	22486	18.0	19.0	27.9	4.18	14.9
average	–	22.7	24.4	34.3	8.58	12.6

Table 4. Switching count reduction ratio by using granularity control (%)

App.	onehot coding						binary coding				
	N_{seg}										
	32	16	8	4	2	1	32	16	8	4	2
mpeg2play											
(116k)	37.1	35.5	34.2	31.8	22.9	6.48	30.0	35.4	37.9	**38.6**	26.9
(245k)	45.6	44.1	42.7	40.3	27.0	5.19	40.7	44.8	**46.5**	45.3	30.9
(649k)	34.4	32.1	31.0	28.0	21.9	6.86	26.7	31.3	34.4	**35.8**	26.0
go	33.8	28.8	28.8	38.9	30.5	20.3	23.7	28.8	39.0	**49.1**	20.3
gzip	10.5	12.0	9.19	7.63	1.94	1.58	8.83	11.6	**12.7**	12.2	8.13
bzip2	17.8	18.5	19.1	18.5	19.2	29.6	7.19	13.1	19.1	25.6	**31.1**
perl	14.5	13.8	11.8	11.5	3.67	5.25	13.3	10.8	14.2	**16.7**	12.0
mesa	19.0	17.1	16.1	13.6	10.8	15.9	6.75	9.30	14.5	20.0	**22.1**
wupwise	18.0	16.6	15.6	16.2	9.71	5.02	12.3	13.9	17.7	**18.7**	15.2
average	22.7	21.6	20.5	20.9	14.7	11.8	16.1	18.9	23.1	**26.8**	20.0

We presented overhead reduction methods and introduced the implementation of our proposed techniques. We can also obtain optimized granularity for switching activity saving when an application is specified in a system. Since it is available to use active bits in a whole datapath for low overhead, we will study how to integrate the techniques using active bits.

In ultra deep submicron VLSI designs coupling effects between on-chip interconnects and leakage current must be addressed. The upper bits, the more effective it is to reduce switching activity by using our scheme. Since lower bits are low switching activity, we can utilize lower bits for shielding against coupling effects like bus shuffling coding [10, 5]. Extra bits cause additional leakage energy. Therefore we must suppress increasing extra bits. We will study low power bus techniques considering these effects on power consumption in deep submicron era.

6. Acknowledgments

This work has been supported by the Grant-in-Aid for Creative Scientific Research No. 14GS0218 and the Silicon Sea-Belt Project “Establishing Project of a Cluster for System-LSI Design and Development”. We are grateful for their support.

The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo in collaboration with Hitachi Ltd. and Dai Nippon Printing Corp. and Cadence Design Systems, Inc. and Synopsys, Inc.

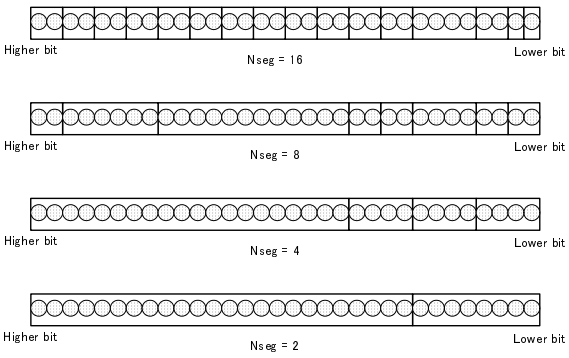

Figure 8. Application-specific division

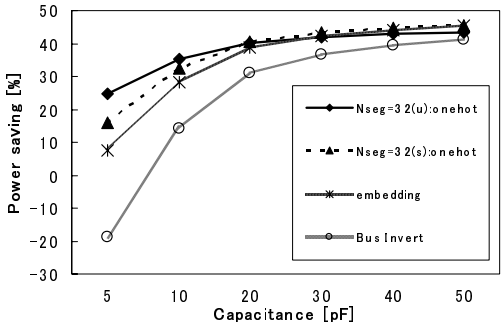

Figure 9. Comparison of total power consumption

References

[1] http://www.simplescalar.com/.

[2] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi. Synthesis of low-overhead interfaces for power-efficient communication over wide buses. In *Proceedings of 36th Design Automation Conf.*, pages 128–133. IEEE/ACM, June 1999.

[3] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano. Asymptotic zero-transition activity encoding for address buses in low-power microprocessor-based systems. In *Proceedings of the 7th Great Lakes Symposium on VLSI*, pages 77–82, March 1997.

[4] D. Brooks and M. Martonosi. Value-based clock gating and operation packing: dynamic strategies for improving processor power and performance. *ACM Transactions on Computer Systems (TOCS)*, 18(2):89–126, 2000.

[5] J. Henkel and H. Lekatsas. a^2bc:adaptive address bus coding for low power deep sub-micron designs. In *Proceedings of 38th Design Automation Conf.*, pages 744–749. IEEE/ACM, June 2001.

[6] H. Mehta, R. M. Owens, and M. J. Irwin. Some issues in glay code addressing. In *Proceedings of the 6th Great Lakes Symposium on VLSI*, pages 178–180, March 1996.

[7] E. Musoll, T. Lang, and J. Cortadella. Workng-zone encoding for reducing the energy in microprocessor address buses. *IEEE Tansaction on VLSI Systems*, 6(4):568–572, December 1998.

[8] T. Okuma, Y. Cao, M. Muroyama, and H. Yasuura. Reducing access energy of on-chip data memory considering active data bitwidth. In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 88–91. ACM Press, 2002.

[9] P. R. Panda and N. D. Dutt. Reducing address bus transitions for low power memory mapping. In *Proceedings of 1996 European Design and Test Conference*, pages 63–67, March 1996.

[10] Y. Shin and T. Sakurai. Coupling-driven bus design for low-power application-specific systems. In *Proceedings of 38th Design Automation Conf.*, pages 750–753. IEEE/ACM, June 2001.

[11] M. R. Stan and W. P. Burleson. Bus-invert coding for low-power i/o. *IEEE Transaction on VLSI Systems*, 3(1):49–58, March 1995.

[12] H. Yamashita, H. Tomiyama, A. Inoue, F. N. Eko, T. Okuma, and H. Yasuura. Variable size analysis for datapath width optimization. In *Proceedings of Fifth Asian Pacific Conference on Hardware Description Language*, pages 69–74, July 1998.