Irredundant Address Bus Encoding Techniques based on Adaptive Codebooks for Low Power

Satoshi Komatsu

VLSI Design and Education Center
University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, JAPAN
Tel: +81-3-5841-6764
Fax: +81-3-5841-6764
e-mail: komatsu@cad.t.u-tokyo.ac.jp

Masahiro Fujita

Department of Electronic Engineering
University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, JAPAN
Tel: +81-3-5841-6673
Fax: +81-3-5841-6724
e-mail: fujita@ee.t.u-tokyo.ac.jp

Abstract— The power dissipation at the off-chip bus is a significant part of the overall power dissipation in digital systems. This paper presents irredundant address bus encoding methods which reduce signal transitions on the instruction address buses by using adaptive codebook methods. These methods are based on the temporal locality and spatial locality of instruction address. Since applications tend to JUMP / BRANCH to limited sets of addresses, proposed encoding methods assign the least signal transition codes to the addresses of JUMP / BRANCH operations in the past. Our encoding methods reduce the signal transitions on the instruction address buses by an average of 88%.

I. Introduction

Recently interest in VLSI design has shifted to minimizing the power dissipation from the conventional design criteria such as chip area and operation speed. In general microprocessor based systems, a large amount of power dissipation comes from the system buses such as highly capacitive memory buses. In these situations, power reduction of buses has become serious concern, for low power VLSI design, as the power dissipation related to the bus is becoming the major part of the total chip power dissipation. Generally, capacitance of such buses are several orders higher than the capacitance of on-chip internal logic because the buses between microprocessors and memories are off-chip wide long interconnects. It means that some additional encoding and decoding circuits can reduce signal transitions of buses (it corresponds to the power dissipation of buses) even though additional power dissipation of such codec circuits are necessary. Many power reduction methods for bus including data encoding schemes have been proposed[1, 2, 3, 8, 9, 10]. Figure 1 shows basic concept of low power bus encoding / decoding scheme. In this paper we introduce new address bus encoding techniques which are based on adaptive code-book methods. Since the address buses have their own characteristics (temporal and spatial locality), the signal transitions can be easily reduced by bus encoding schemes. In addition our methods use adaptive codebooks to enhance the performance of signal transition reduction, it gives more power reduction compared to the other previous works. Since our methods use codebooks more effectively than other work[7], the codebook hit rates are still high and higher reduction of signal transitions can be achieved.

The rest of this paper is structured as follows: first we describe the previous works related to our work. In section 3, we explain the idea of our encoding methods. In section 4, we present the experimental results by using an instruction set simulator and SPEC2000 benchmarks. Finally we conclude this work in section 5.

II. Previous Works

In this section we review previous related works. There are a lot of low power bus encoding methods and some of them are reviewed below. In many encoding methods for low power system buses proposed so far, they use one or more redundant bits to reduce signal transitions effectively on buses. In many systems, such redundant bus lines (interconnects) are not acceptable because of implementation problems on the PCBs or MCM. Since our methods are irredundant low power address bus encoding methods, most of the following are irredundant address bus encoding methods.

The following notation will be used in this section:

Data(t): Address value to be sent on the bus at time **t**
Bus(t): Encoded value on the bus at time **t**
S: Stride between successive two addresses

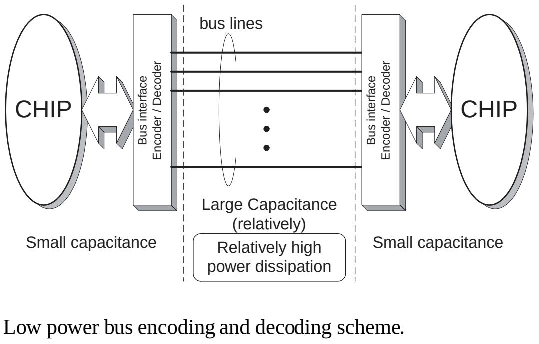

CHIP
Bus interface Encoder / Decoder
bus lines
Bus interface Encoder / Decoder
CHIP
Large Capacitance (relatively)
Small capacitance
Relatively high power dissipation
Small capacitance

Fig. 1. Low power bus encoding and decoding scheme.

```
if (**Data(t)** == **Data(t–1)** + **S**)
    **Bus(t)** = **Bus(t–1)**
else if (**Data(t)** != **bus(t–1)** )
    **Bus(t)** = **Data(t)**
else
    **Bus(t)** = **Data(t–1)** + **S**
```

Fig. 2. Pseudo code of T0-C encoding method

Bus invert encoding[1] was proposed by M. R. Stan *et al.*. In this encoding method, one redundant bit is added to N-bit bus. In case that the hamming distance between **Data(t)** and **Bus(t–1)** is more than $N/2$, **Data(t)** is encoded to $\overline{\mathbf{Bus}(\mathbf{t}-\mathbf{1})}$. In the other cases, **Data(t)** is encoded to **Bus(t–1)**. The redundant bit is added to indicate if **Bus(t)** is inverted or not. By using this encoding method, the number of signal transitions is reduced by 10–20% for data buses (not for address buses). Adaptive code-book encoding method[4] is one of the low power bus encoding methods that have the same concept with bus invert encoding. In these encoding methods, the redundancy is added to reduce signal transitions on the buses by adding the redundant bits. Though these methods are effective for the data buses, they are not effective for address buses because they do not consider the characteristics of address buses.

T0 code[2] was proposed by L. Benini *et al.*. This encoding method is based on the sequentiality of the data stream of address buses to reduce signal transitions on address buses. In this encoding method, in case that the address increases by the uniform stride **S**, **Bus(t)** is encoded to **Bus(t–1)**. In the other cases, the **Bus(t)** is encoded to **Data(t)**. A redundant bit called *INC* is added to indicate if the encoded value is raw address or not. On average, 35% reduction in address bus switching activity is achieved by this encoding method. In addition, the encoding method based on the combination of bus invert encoding and T0 encoding was proposed as T0-BI encoding [3].

T0-C code[5] is an expansion of T0 encoding to omit the redundant bit. Figure 2 is the pseudo code of T0-C encoding method. In this encoding method, the concept is almost the same as T0 encoding. To omit the redundant bit, the exceptional operation is carried out to keep 2^N-to-2^N mapping between the original and encoded addresses. This expansion also increases the performance of signal transition reduction and it reduces the number of signal transition on address bus by 70%.

In [6], T0-XOR code and Offset-XOR code are proposed. The former works as follows:

$$\mathbf{Bus(t)} = \mathbf{Data(t{-}1)} \oplus \mathbf{S} \oplus \mathbf{Bus(t{-}1)}$$

It works the same as T0 encoding when an address increments uniformly. In other cases, the signal transitions are reduced compared to T0 encoding because the offset address between before and after JUMP operation is usually small. Its concept can be applied to the Offset-XOR encoding and it works as follows:

$$\mathbf{Bus(t)} = (\mathbf{Data(t{-}1)} - \mathbf{Data(t)}) \oplus \mathbf{Bus(t{-}1)}$$

Usually the offsets are small numbers and the XOR of the previous bus value and the offset has a small number of transitions.

In ALBORZ encoding[7], the authors introduced another encoding of the offset for Offset-XOR encoding. For example, the small minus offset (such as “–1”) has many 1s in binary representation. ALBORZ method changes it to a value with fewer number of 1s. This method also has codebooks and it contains the recently used offsets. In case the offset is hit to the contents of the codebook, the bus signal transitions are reduced by using LWC(Low Weight Code). This encoding method can reduce the number of signal transition on address buses by up to 89%.

III. Encoding Method

This section explains encoding techniques that we propose in this paper. First we describe T0-C based encoding method with an adaptive codebook. Then we describe Offset-XOR based encoding method with an adaptive codebook. Both encoding methods utilize adaptive codebooks to enhance the performance of signal transition reduction considering temporal and spatial locality of instruction address buses. Please note that both encoding methods are irredundant (no extra bit).

A. T0CAC Code

In this section, we introduce a new address bus encoding method based on T0-C code[5]. As described in the previous section, T0-C code is an expansion of T0 code by omitting the redundant bit. To improve the performance, we combined adaptive codebook with T0-C code, we named it T0CAC (T0-C with adaptive codebook) code.

The basic concept is to assign the least signal transition codes to the most frequent patterns. For example, the following case occurs on the instruction address bus without JUMP or BRANCH operations.

$$\mathbf{Data(t)} = \mathbf{Data(t{-}1)} + \mathbf{S}$$

Our method assigns zero transition code to such cases just like T0 code (or T0-C code). As a result, **Data(t)** is encoded to **Bus(t–1)**. In addition, we use an adaptive codebook which contains the recently accessed destination address of JUMP / BRANCH operation (JUMP/BRANCH destination address history). In case that the address to be transmitted is hit to one of the addresses in the codebook, our method assigns MSB one-hot code to such cases. For example, in case of a loop operation, the same destination addresses of JUMP/BRANCH operations will frequently appear. By using T0CAC code, the number of signal transitions is effec-

```
1:  if ( Data(t) == Data(t–1) + S)
2:       Bus(t) = Bus(t–1)
3:  else if ( Bus(t–1) == Data(t) )
4:       Bus(t) = Data(t–1) + S
5:  else if (Data(t) == His[i])
6:             if ((Bus(t–1) $\oplus$OH[i] $\neq$ Data(t–1) + S)&&
7:            (Bus(t-1) $\oplus$OH[i] $\notin$ {His[j]}) &&
8:            (Data(t) $\notin$ {Bus(t–1) $\oplus$ OH[j]}))
9:                 Bus(t) = Bus(t–1) $\oplus$ OH[i]
10:                update His
11:           else
12:                Bus(t) = Data(t)
13: else if (Data(t) == Bus(t–1) $\oplus$OH[i])
14:           if ((His[i] $\neq$Bus(t–1)) &&
15:           (His[i] $\neq$Data(t0-1) + S) &&
16:           (His[i] $\notin_{j \neq i}$ Bus(t–1) $\oplus$OH[j]) &&
17:           (Data[i] $\notin_{j \neq i}$ {His[j]}))
18:                Bus(t) = His[i]
19:           else
20:                Bus(t) = Data(t)
21: else
22:      Bus(t) = Data(t)
```

Fig. 3. Pseudo code of T0CAC code

tively reduced in such cases. Then the encoder and the decoder update their own codebooks to keep consistency between them. Since this encoding method is irredundant, the 2^N-to-2^N mapping should be maintained in the N-bit buses. In other words, the encoded bus value must be decoded to the original value uniquely. To satisfy these assumptions, these assignments should be swapped in the 2^N-to-2^N mapping. The detailed operations are described in the following sections.

Encoding Method of T0CAC code

Figure 3 shows the pseudo code of T0CAC code. The following notations are used in this figure.

Data(t): Address value on the bus at time **t**

Bus(t): Encoded value on the bus at time **t**

S: Stride between successive two addresses

N: Address bus width

M: The maximum number of addresses to be stored in the codebook

His[i]: Addresses contained in the codebook ($0 \leq \mathbf{i} \leq \mathbf{M} \leq \mathbf{N}$)

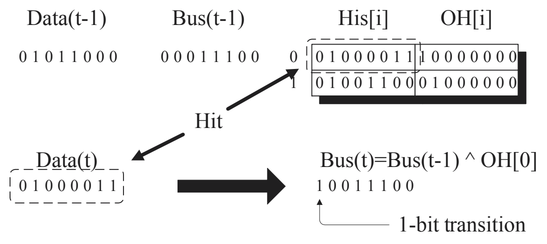
OH[i]: MSB One Hot code ($0 \leq \mathbf{i} \leq \mathbf{M} \leq \mathbf{N}$)

OH[0] = $10000..._{(2)}$

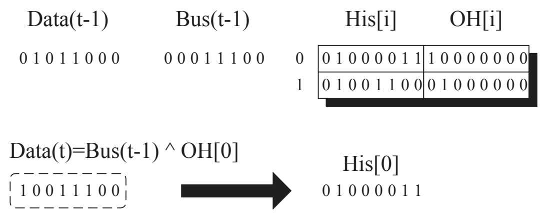
OH[1] = $01000..._{(2)}$

OH[2] = $00100..._{(2)}$

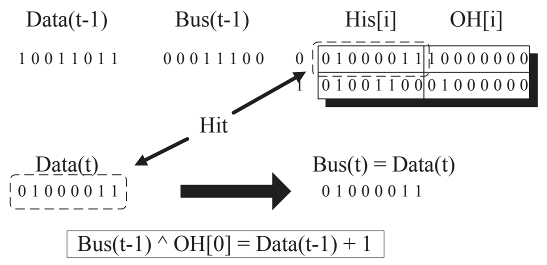
Figure 4 also shows the encoding examples of T0CAC code, where address bus width (**N**) is 8-bit, codebook size (**M**) is 2, respectively.



(a) in case that address is hit to one of the contents of codebook



(b) in case that **Data**(t) = **Bus**(t–1) $\oplus$ **OH**[i]



(c) an example of exceptional operation

Fig. 4. Encoding examples of T0CAC code

The following are the detailed encoding operation of each step.

1. Operate T0-C encoding

First T0CAC encoder perform T0-C encoding to the raw address. It corresponds to the lines 1 to 4 of the Fig. 3.

2. Check whether the address is contained in {His[i]}

Then the raw address is checked whether it is contained in the destination address history list {**His[i]**} (it is the codebook). In case that the address is contained in the codebook, **Bus(t)** = **Bus (t–1)** $\oplus$ **OH[i]** is transmitted to the bus, where **OH[i]** is one of the MSB One-Hot codes. Therefore in such a "hit" case, the number of signal transition of bus is 1, which means that T0CAC code is more effective than T0-C code. This operation corresponds to the lines 5 to 12 of Fig. 3. Figure 4(a) shows this cases. On the contrary, in case that **Data(t)** = **Bus(t–1)** $\oplus$ **OH[i]**, the decoder can decode uniquely by transmitting **Bus(t)** = **His[i]**. It is the swapping of assignments. This operation corresponds to the lines 13 to 20 of Fig. 3. Figure 4(b) shows this cases.

3. Perform exceptional operations

After performing step 1 and 2, there may exist inconsistencies and it results in the breaking of 2^N-to-2^N mapping. For example, in the following case,

Bus(t–1) $\oplus$ **OH[i]** = **Data(t–1)** + **S**

the result of step 1 and the result of step 2 indicate same values. In such cases, our encoder transmitted original address value to the bus (**Bus(t)** = **Data(t)**). By this exceptional operations, decoder can decode uniquely because decoder has the same informations of (**Data(t), Data(t–1), Bus(t–1),** $\{$**His[i]**$\}$). Figure 4(c) shows this cases.

4. Update the codebook

Finally the encoder and the decoder update their codebook on the same manner. In the current implementation, the codebook is updated only when JUMP/BRANCH operation is performed and the address does not hit to the codebook. For simplifying the codebook circuits, we use the FIFO buffer for the codebook.

This encode method is similar to ALBORZ code[7]. In ALBORZ code, contents of codebooks are the offsets of successive instruction addresses to simplify the encoding algorithm. On the other hand, contents of codebooks are the original destination address values in our method. From these differences, we assume our method can increase the codebook hit rate because the hit rate depends not on the offset but the destination address essentially. As a result, our method can reduce signal transitions more effectively.

B. OXAC Code

In this section, we introduce another address bus encoding method based on Offset-XOR code[6]. T0CAC code can reduce signal transitions when the address value increments or the address value hits to the codebook. However in other situations, it cannot reduce signal transitions effectively because the raw address value is transmitted. To improve the performance, we introduce a new encoding method, OXAC (Offset-XOR with adaptive codebook) code.

Basic concept is very similar to T0CAC code in the sense that both encoding methods use an adaptive codebooks which contain history of JUMP/BRANCH destination addresses. In the cases that the address does not hit to the codebook, OXAC encoder transmits an offset between **Data(t)** and **Data(t–1)** whereas T0CAC encoder transmits a raw address value(**Data(t)**).

Figure 5 shows the block diagram of OXAC encoder. The following are encoding method of OXAC code:

1. Calculate an offset between Data(t) and Data(t–1) and encode it

First the **Offset** between the previous address and the expected address of incrementation is calculated as follows:

Offset = **Data(t)** – (**Data(t–1)** + **S**)

In case that an address increments, **Offset** is zero. To reduce the number of 1's in small negative offsets, LSB-Inv function[7] is given to the offset. LSB-Inv function inverts all bits of negative numbers except their MSBs, and positive numbers unchanged. Table I shows examples of LSB-Inv function and as a result the modified offsets of small negative value have less 1's in their binary representation.

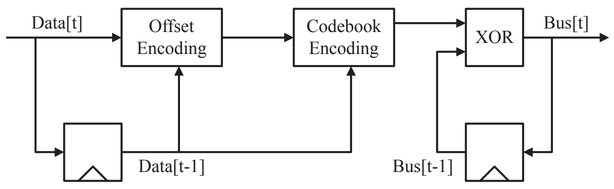


Fig. 5. Block diagram of OXAC encoder

TABLE I
EXAMPLE OF LSB-INV FUNCTION[7]

Original offset	Modified Offset
FFFFFFFFh, (-1)	80000000h
FFFFFFFEh, (-2)	80000001h
FFFFFFF5h, (-10)	80000009h
80000000h	FFFFFFFFh

2. Swapping using an adaptive codebook

Then the modified offsets are swapped as the same manner of T0CAC code. In case the address is hit to one of the contents of the codebook, the modified offset is replaced by the corresponding one-hot code (**OH[i]**). To keep 2^N-to-2^N mapping, the modified offset which is equal to **OH[i]** is also replaced by the corresponding modified offset (**Offset[i]**). These operations just swap mappings.

As a result of this step, the modified offsets have less 1's in their binary representation when the address increments or the address hits to the codebook.

3. XOR operation between previous bus value and modified offset To reduce signal transition, the encoder calculate XOR operation between **Bus(t–1)** and modified offset and transmit it to the bus. It results in the signal transition reduction of the bus since the modified offsets are assumed that they have less 1s in terms of binary representation.

4. Update the codebook

Finally the encoder and the decoder update their codebook on the same manner. It is done just same as T0CAC code which is described in the previous section.

C. Decoding Methods of T0CAC Code and OXAC Code

As described above, both T0CAC code and OXAC code use an input address value (**Data(t)**, **Data(t–1)**) and the previous bus value(**Bus(t–1)**) and the codebook ($\{$**His[i]**$\}$). The decoder can also use them and decode to the original value if the codebook of the encoder and the codebook of the decoder are always identical.

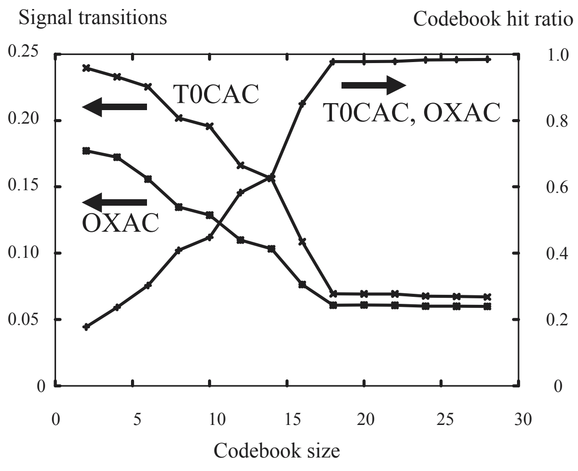


Fig. 6. Signal transition reduction ratio and hit ratio to the codebook for each codebook size

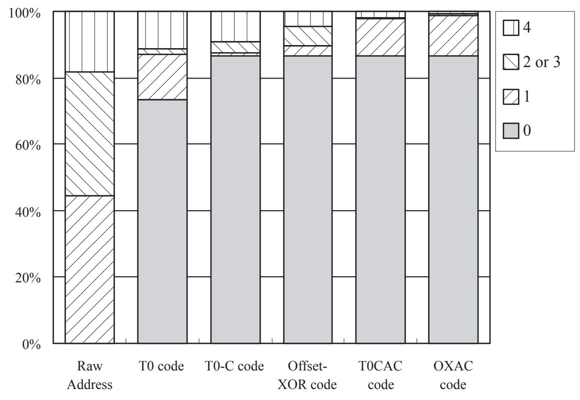


Fig. 7. Distribution of signal transitions in the encoded stream for each encoding method (M=16 for T0CAC code and OXAC code)

IV. Experimental Results

To evaluate the effectiveness of the proposed encoding methods, we implemented these algorithm in C-language. We used the Simplescalar[12] architecture simulator to trace the instruction addresses. We generated instruction address traces for 8 of the SPEC 2000 benchmark programs and each trace consists of 100,000,000 instructions and all the simulations have been done using the SPEC 2000 test input parameters. The used benchmark programs are `gzip`, `gcc`, `mcf`, `ammp`, `art`, `bzip2`, `equake` and `parser`.

Table II shows the signal transition reduction ratios for each encoding method. In this table, the signal transition reduction ratio means the ratio between the number of signal transitions of the raw instruction address and the encoded address. In other words, smaller signal transition reduction ratio indicates higher performance of signal transition reduction. We evaluated T0 code, T0-C code, Offset-XOR code for comparison in addition to T0CAC code and OXAC code. The last row of this table shows the average signal transition reduction ratio over the all benchmarks for each encoding method. As shown in the table II, the proposed encoding methods with 8 or 16 codebooks are superior to other encoding methods in terms of signal transition reduction. And the signal transitions are reduced to 15.8% and 11.8% by using T0CAC code and OXAC code, respectively. These results indicate that adaptive codebook based encoding methods are very effective and applicable for low power encoding of instruction address buses.

Figure 6 shows signal transition reduction ratio and the hit ratio to the codebook for each codebook size for proposed encoding methods. We used `gzip` benchmark for this experiment. As described in the previous section, T0CAC code and OXAC code use the same algorithm for updating codebook. Therefore hit ratio of them are identical. From these results, the signal transition reduction ratio highly depends on the codebook hit ratio. And we can assume 16-entry codebook is sufficient for proposed encoding methods since the codebook hit ratio is saturated around there. As compared with ALBORZ code[7], the hit rate is higher because our methods use codebooks more effectively as described in the previous section. To obtain the same hit rate, our methods require codebooks of half sizes compared to ALBORZ. It indicates that the additional hardware costs of our methods are smaller since the codebook size is one of the parameters which affect the codec circuits.

Figure 7 shows the distribution of signal transitions in the encoded stream for each encoding method. In this experiment, the `gzip` benchmark is also used for evaluation. T0CAC code and OXAC code used the codebooks which contain 16 words for retaining JUMP/BRANCH destination address. Though there is almost no zero-transition pattern in raw address, around 80% of patterns are zero transition in other encoding methods. It is because over 80% of the instruction addresses increment in `gzip` benchmark. Moreover almost 98% of encoded patterns are zero bit or one bit transition in our adaptive codebook encoding methods. It also certifies the performance of our methods.

Finally, we analyzed the power dissipation overhead of the codec circuit of the proposed encoding method. To estimate the power dissipation of OXAC encoder and decoder, we designed them by using Verilog-HDL and we got the physical transistor net-list by using a commercial logic synthesis tool and a commercial P&R tool. The target technology was 1.8-volt 0.18-μm CMOS process. We also used a switch level circuit simulator to estimate the power dissipation with traced instruction address of `gzip` benchmark. Figure 8 shows the total power reduction including power dissipations of encoder / decoder circuits for each value of the external address bus capacitance. In this figure, the total power reduction ratio indicates the ratio between the non-encoded raw address stream and the encoded address stream including the power dissipation of codec circuits, where the signal swings of I/O interfaces are 3.3-volt. Since this experimental result is a first trial of implementation of the proposed methods, the circuit is not well optimized. Still it is clear that our methods are effective for large value of address bus capacitances.

TABLE II
SIGNAL TRANSITION REDUCTION RATIO WITH RESPECT TO BINARY ENCODING FOR EACH BENCHMARK

Benchmark	Raw Address	T0	T0-C	Offset-XOR	T0CAC				OXAC			
					M=2	M=4	M=8	M=16	M=2	M=4	M=8	M=16
gzip	1	0.373	0.270	0.195	0.239	0.233	0.202	0.109	0.177	0.172	0.135	0.076
gcc	1	0.248	0.258	0.211	0.245	0.232	0.215	0.196	0.192	0.181	0.167	0.152
mcf	1	0.280	0.205	0.208	0.198	0.167	0.095	0.076	0.150	0.135	0.077	0.067
ammp	1	0.333	0.279	0.190	0.266	0.240	0.204	0.161	0.174	0.166	0.141	0.117
art	1	0.167	0.158	0.138	0.147	0.106	0.094	0.090	0.083	0.056	0.049	0.047
bzip2	1	0.437	0.309	0.247	0.309	0.309	0.285	0.285	0.247	0.247	0.219	0.219
equake	1	0.371	0.262	0.188	0.261	0.235	0.232	0.224	0.186	0.170	0.168	0.163
parser	1	0.399	0.302	0.199	0.282	0.257	0.127	0.126	0.176	0.166	0.102	0.101
Average	1	0.339	0.255	0.197	0.243	0.222	0.182	0.158	0.173	0.162	0.132	0.118

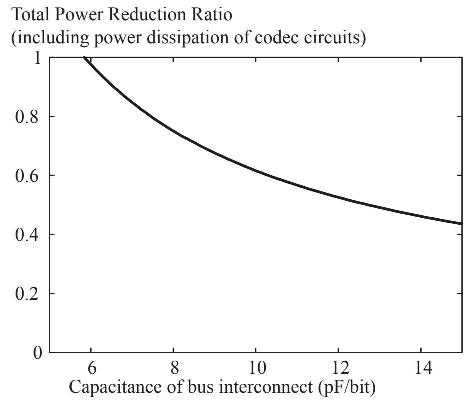


Fig. 8. Total power reduction including power dissipations of codec circuits by using OXAC encoding method

V. CONCLUSION AND FUTURE WORKS

In this paper, we proposed new low power encoding methods which were applicable for instruction address buses. Since the proposed methods use codebooks which contain JUMP / BRANCH destination addresses, they can reduce signal transitions on the address buses more effectively than other methods. These methods can be easily applicable for conventional systems, because they are irredundant encoding methods. Experimental results showed that these methods can reduce signal transitions by 80–95% (average 88%). Moreover circuit simulation results indicate these methods are very attractive when the address bus capacitance is large.

As future works, we are refining the encoding algorithm to decrease hardware overheads of codec circuits. We believe that small modifications of encoding algorithms can reduce the overheads considerably. Moreover we are investigating low power address bus encoding methods based on other profiles of address traces though we only used only JUMP / BRANCH destination address in this paper. We assume that these low power bus encoding scheme will contribute to future low power systems.

ACKNOWLEDGEMENTS

The VLSI circuit in this study is designed with Synopsys and Cadence CAD tools through the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo.

REFERENCES

[1] M. R. Stan *et al.*, “Bus-Invert Coding for Low Power I/O,” *IEEE Transactions on very large scale integration systems,* pp. 49–58, Mar. 1995.

[2] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, “Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems,” *IEEE 7th Great Lakes Symposium on VLSI,* pp. 77–82, Mar. 1997.

[3] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, “Address Bus Encoding Techniques for System-Level Power Optimization,” *Design Automation and Test in Europe,* pp. 861–866, 1998.

[4] S. Komatsu, M. Ikeda, K. Asada, ”Bus Data Encoding with Coupling-driven Adaptive Code-book Method for Low Power Data Transmission,” *27th European Solid-State Circuits Conference,* pp. 312-315, Sep. 2001.

[5] Y. Aghaghiri, F. Fallah, M. Pedram, “Irredundant Address Bus Encoding for Low Power,” *International Symposium on Low Power Electronics and Design ’01,* pp. 182–187, 2001.

[6] W. Fornaciari, M. Polentarutti, D. Sciuto and C. Silvano, “Power Optimization of System-Level Address Buses Based on Software Profiling,” *CODES 2000,* pp. 29-33, 2000.

[7] Y. Aghaghiri, F. Fallah, M. Pedram, “ALBORZ: address level bus power optimization,” *International Symposium of Quality Electronic Design 2002,* pp. 470–475, 2002.

[8] L. Benini *et al.*, “Synthesis of low-overhead interfaces for power-efficient communication over wide buses,” *Design Automation Conference ’99,* pp. 128-133, Jun. 1999.

[9] P. Sotiriadis *et al.*, “Transition Pattern Coding: An approach to reduce Energy in Interconnect,” *ESSCIRC 2000,* Sep. 2000.

[10] S. Ramprasad *et al.*, “A coding framework for low-power address and data busses ,” *IEEE Trans. on VLSI Systems,* pp. 212–221, Vol.7, No. 2, Jun. 1999.

[11] Semiconductor Industry Association, “International Technology Roadmap for Semiconductors 1999 Edition,” 1999.

[12] http://www.simplescalar.org/