A System-level Energy Minimization Approach Using Datapath Width Optimization

Cao, Yun
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University

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

出版情報：電子情報通信学会技術研究報告. VLD, VLSI設計技術. 100 (646), pp.49-54, 2001-03-02. 電子情報通信学会
バージョン：
権利関係：

A System-level Energy Minimization Approach Using Datapath Width Optimization

Yun Cao Hiroto Yasuura
Department of Computer Science and Communication Engineering
Graduate School of Information Science and Electrical Engineering
Kyushu University
6–1 Kasuga-koen, Kasuga-shi, Fukuoka 816-8580 Japan
{*cao, yasuura*}*@c.csce.kyushu-u.ac.jp*

Abstract This paper presents a novel system-level approach that minimizes the energy consumption of embedded core-based systems through datapath width optimization. It is based on the idea of minimizing energy consumed by redundant bits, which are unused during execution of programs by means of optimizing the datapath width of processors. To minimize the redundant bits of variables in an application program, the effective size of variables is determined after variable size analysis, and Valen-C language is used to preserve the precision of computation. Analysis results of variables show that there are average 39% redundant bits in MPEG-2 decoder C source program. In our experiments for several embedded applications, energy savings without performance penalty are reported range from about 10.0% to 48.3%.

Key Words energy minimization design, datapath width optimization, soft-core processor, Valen-C

1 Introduction

Minimizing power consumption of embedded systems is a crucial task. Battery-operated portable systems demand tight constraints on energy consumption. Better low-power circuit design techniques and advances in battery technology have helped to increase battery lifetime. On the other hand, managing power dissipation at higher design levels can considerably reduce energy consumption, and thus increase battery lifetime. Energy consumption at all design levels should be considered to reduce the energy consumption of the whole embedded system.

We have developed the design platform, which consists of Valen-C retargetable compiler, soft-core processor and a cycle-based simulator. We also have done some research on reduction of area and cost for embedded core-based systems [6]. In this paper, we focus on minimizing energy dissipation and present a system-level approach for embedded core-based systems, which minimizes energy consumption of the whole system while providing adequate performance level. In the initial design phase of our approach, we design a system with a soft-core processor [1], data RAMs, instruction ROMs and logic circuits. Then we analyze the effective bit width of each variable of the application program. After that, using the analyzed results, we rewrite the application program in Valen-C [2], in which we specify the word length of each variable satisfying accurate computation. Thus, using Valen-C, we can reduce energy consumed by redundant bits of application program. After verifying the functionality of the initial design, we modify several design parameters of the soft-core processor, including the datapath width, the number of registers and the instruction set. Thus we can tune up the soft-core processor to minimize the energy consumption while satisfying the system performance constraints. To get first-cut estimates of energy consumption early in the design, a few component-based power estimation models was also developed, total energy is obtained by summing over all components of the system.

Our paper is structured as follows: the next Section 2 gives an overview of related work. Section 3 describes our energy minimization approach by datapath width optimization. Section 4 presents our energy estimation models. Experiments and results are shown in section 5. Finally, Section 6 concludes our work.

2 Related Work

Hardware and software techniques to reduce energy consumption have become an essential part of current system designs. Extensive researches on power optimization, from circuit level, gate level, RT level to system level, have been conducted in these recent years. Such techniques have particularly targeted the memory system due to the prevalent use of data-dominated signal and video applications in mobile devices, such as [10], which focuses on creatively exploiting cache to reduce power consumption. The work reported in [7] deals with an architectural-oriented power minimization approach. A power and performance simulation tool that can be used to conduct architecture-level optimizations has been introduced by Sato et al. [8]. The approach described in [9] uses a multiple-voltage power supply to minimize system-power consumption.

Different with these researches, we propose a system-level energy minimization approach in which designers can control the width of datapath freely. The energy consumption of the whole system is drastically reduced without decline of performance by optimizing the datapath width.

3 An Energy Minimization Approach

In the design of consumer electronic systems, it is very important to utilize soft-core processors in order to meet the complexity of embedded systems and stringent time-to-market constraints. Therefore, core-based solutions are present in energy-efficient system design. Our approach gives designers a freedom to determine the datapath width of soft-core processor, because the datapath width of a processor has great impacts not only on power consumption and performance of the processor but also on those of memories. Optimizing datapath width for each given application is an effective approach to minimize energy consumption of the whole embedded systems. The energy minimization problem is stated as:

$$\begin{array}{ll} minimize & Energy(w) \\ subject\ to & Cycle(w) \leq C_{cst} \\ & Area(w) \leq A_{cst} \end{array}$$

Where $Energy(w)$, $Cycle(w)$ and $Area(w)$ are functions of the datapath width w, C_{cst} and A_{cst} are the constraints on the execution cycle and area respectively. This is a nonlinear optimization problem. The overview of our energy minimization algorithm is described in Figure2.

Figure1 shows our proposed approach, which consists of the following phases:

- Phase 1: The source program of the target application, which was originally written in C or other language, is rewritten in Valen-C language, after the bit width of each variable is analyzed. For instance, if the variable x requires at most 11 bits, the programmer can write $int11\ x$; in the variable declaration of Valen-C program.
- Phase 2: Bung-DLX is customized to different soft-core processors by choosing different design parameters, such as the datapath width and the address size of the data memory.
- Phase 3: The Valen-C source program of an application is compiled for the customized soft-core processors. Valen-C compiler generates the assembly code from the source program. As a result, different embedded systems are generated based on different customized processors and assembly codes. At this phase the size of both the data memory and the instruction memory of each system are estimated.
- Phase 4: The systems generated from phase 3 are evaluated. Execution cycles, memory size and power consumption are estimated. The impact of the design parameters on the energy consumption and on the system performance are evaluated. Thus the embedded system of the minimal energy, which satisfies the design constraints, is chosen among those systems.

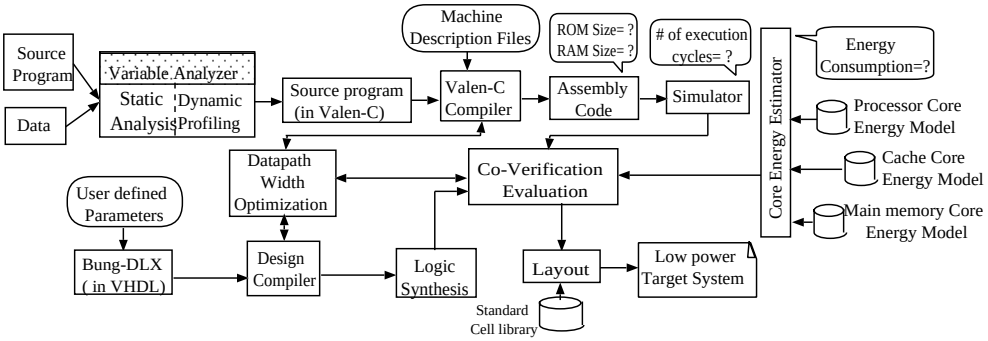

Figure 1: An energy minimization approach using datapath width optimization

3.1 Variable Size Analysis

In order to optimize datapath width, the effective size of each variable in programs needs to be analyzed. This section explains our methods to analyze effective sizes of variables in C programs. In this paper, we define *effective size* as the smallest size which can hold both maximum and minimum values of a variable. In many cases, some bits of a variable are never used during execution of a program. If a variable x of unsigned integer type whose value is in [0, 2000], i.e. between 0 and 2000, then the number of necessary bits of x is 11, because the 11-bit size is large enough to hold any value in [0, 2000].

We use two methods to analyze effective size of variables. One is dynamic analysis, which runs programs and monitors the value of each variable. Dynamic analysis is one kind of simulation-based method whose results depend on input data sets given to the programs. The other is static analysis.

For Static analysis, when the maximum value of an unsigned integer variable x is n_{max}, the effective size of x, $e(x)$, is given as follows:

$$e(x) \quad = \quad \log_2(n_{max} + 1) \tag{1}$$

For a signed integer x with a maximum value n_{max} and a minimum value n_{min}, $e(x)$ is defined as follows:

$$e(x) \quad = \quad \lceil \log_2 \mathcal{N} \rceil + 1 \tag{2}$$

where

$$\mathcal{N} \quad = \quad \max(|n_{max}| + 1, |n_{min}|) \tag{3}$$

Static analysis is an efficient method to analyze the effective size of variables. However, in many cases when we can not predict the assigned value of a variable unless we execute the program, such as the case of unbounded loops, static analysis becomes insufficient. As a solution to this problem, we adopted dynamic analysis in our approach.

In dynamic analysis, we execute the program with input data and monitor the values assigned to each variable. We insert the monitoring function to the assignment statement of variables. The arguments of the monitoring function are the variable name and its assigned value. The monitoring function checks the value assigned to the variable, verifies the bit width required and then memorize it. After that, it keeps the bit width temporarily in a table. When the monitoring function checks the same variable with a different assigned value, it compares the new bit width with the bit width already memorized in the table, and keeps the bigger one in the table and so on. Thus, the required bit width of the variable is got after executing the program.

3.2 Efficient Use of Data Memory

Since in many cases, high-level specifications are devoted to describe functionalities of target systems rather than implementation details, they often contain a lot of redundancies such as duplicated computations and never executed code. Therefore, the specifications must be optimized to remove the redundancies for energy-efficient design. Some redundancies are introduced in size of variables. For example, in C programs, a variable whose value is between 0 and 1000 is often declared as the *int* type, i.e., usually 16 or 32 bits depending on target processors, and then some upper bits make nonsense. This means that the memory has many unnecessary bits, which do not essentially contribute to the calculation of programs. Therefore redundant bits must be removed to reduce power consumption.

C provides for three integer sizes, declared using the keywords *short*, *int* and *long*. The compiler designer determines the sizes of these integer types. In many processors, the size of *short* is 16 bits, *int* is 16 or 32 bits, *long* is 32 bits. On the other hand, in Valen-C, programmers explicitly specify the required bit width of each integer data type. Thus it becomes possible to reduce the energy of the datapath and the data memory, which is dissipated by the redundant bits. For instance, if variables x, y, and z require 12, 20 and 24 bits respectively, the programmer can write "int12 x; int20 y; int24 z;" in the variable declaration of Valen-C program. If a processor with a datapath width of 20 bits is used in the system, the total memory size will be 80 bits. Moreover, the unused bits in the data memory will be 24 bits. On the other hand, if a processor of a datapath width of 12 bits is used, the total data memory size will become only 60 bits, and the unused memory size will decrease to 4 bits. As a result, specifying the word length required for each variable and changing the datapath width have a significant role in reducing the data memory size of a system. Therefore it also affects the power consumption of the system.

- Input:
  - source program : S
  - input data : D_{in}
  - constraint of cycle : C_{cst}
- Variable:
  - datapath width $w_i \in W$=[w_1,w_2,...w_n]
- Output:
  - execution cycles $c_i \in C$ = [c_1,c_2,...c_n]
  - Energy Consumption E_{min} when $c_k \leq C_{cst}$
  - datapath width w_k when $E_k = E_{min}$
- Phase 1 : Analysis
  - static analysis of variable size $x_i \in [x_1,x_2,...x_n]$
  - compile the source program
  - dynamic analysis of variable size $y_i \in [y_1,y_2,...y_n]$
- Phase 2 : Soft-core processor (Bung-DLX)
  - define the design parameters
    - datapath width w_i
    - the number of registers n_i
- Phase 3 : Valen-C program
  - variable declaration of bit width (x_{ie} and y_{ie})
  - compile the Valen-C source program for customized Bung-DLX at w_i
- Phase 4 : Estimation
  - for $w_i \in W$
    - calculate the execution cycles c_i
    - calculate the energy consumption E_i
    - get (E_{min},w_k) when $c_k \leq C_{cst}$
  - return (E_{min},w_k)

Figure 2: Pseudo code of the algorithm for energy minimization

3.3 Datapath Width Optimization

System designers can tune the value of the datapath width in accordance with the characteristics of target system to deliver most suited processor. Designers can reduce the datapath width until the single precision point (SPP) without performance loss. SPP is the processor datapath width, which is equal to the bit width of the largest variable in a program. It is the smallest datapath width at which all instructions can remain single-precision. Designers may obtain better solutions, more power savings by shrinking the datapath less than SPP, under performance constraints. Figure3 shows the overview of our datapath width optimization algorithm.

3.4 Power Versus Performance Tradeoff

Minimizing power consumption is not simply an altruistic activity. A device consuming less power will accrue several desirable advantages such as longer battery life for wireless devices, but somewhat less obvious advantages, such as reliability and performance. The datapath width of a processor strongly affects the power consumption of the whole system including the processor,data memories and instruction memories, It also affects the execution cycles of a given task, i.e., narrowing the datapath width less than SPP will cause the increase of execution cycles because of multiple-precision operations. For example, that an addition of 20 bit data is executed by only one instruction on a 20 bit processor is assumed, If the datapath width becomes to 10 bits, two instructions including additions of lower 10 bits and high 10 bits with carry are required. So trade-offs exist between datapath width and execution cycles. Although a processor with narrower datapath width dissipates lower power per clock cycle, the total energy for the task is not reduced always by narrowing the datapath width. Thus, for a given target system, trading off the performance and power consumption is an important work.

- Input:
  - datapath width : $W_i \in W$=[$W1$,$W2$,...,W_n]
  - execution cycles : $C_i \in C$=[$C1$,$C2$,...,C_n]
  - energy consumption : $E_i \in E$=[$E1$,$E2$,...,E_n]
- Output:
  - E_{min} when $C_k \leq C_{cst}$
  - W_k when $E_k = E_{min}$
  - C_k when $E_k = E_{min}$
- Algorithm:

```
for (W_i ≠ W_n)
    get (C_i, E_i)
    if C_i ≤ C_Cst then
      ( E_min := E_i
        W_k := W_i
        C_k := C_i )
    W_i := W_i+1
end loop
return(E_min, W_k, C_k)
```

Figure 3: Pseudo code of the algorithm for datapath width optimization

4 Energy Estimation Models

This section describes energy consumption models. The total energy consumption, E, is the summation of energy consumed by the processor (E_{proc}) and memories (E_{mem}).

$$E = E_{proc} + E_{mem} \tag{4}$$

We estimated E_{proc} and E_{mem} separately, and got the energy consumption model of our soft-core processor generated by HITACH 0.5um CMOS technology and the energy consumption models of memory generated by Alliance CAD System Ver.3.0 with 0.5um double metal CMOS technology.

E_{proc} is given by

$$E_{proc} = \sum_{i \in I} e_i \times Cycle_i \tag{5}$$

where

e_i : Average energy of instruction i

$Cycle_i$: The number of execution of instruction i

I : Instruction set of Bung-DLX

e_i is obtained by performing post-layout simulation of switch-level. After several simulations, we obtained the empirical energy model at several datapath widths in Table 1, where power savings are got by comparing to the power consumption of 32bits Bung-DLX. The power dissipation in static CMOS can be divided into static, dynamic and short-circuit power. Because static power and short-circuit power are far less than dynamic power, we just focus on dynamic power, which consists of Cell Internal Power(P_c) and Net Switching power(P_s).

e_i is shown as follows:

$$e_i = \frac{1}{2} \times V_{dd}^2 \sum_{net} [C(j) \times S(j) + E_{c(k)} \times S(k)] \tag{6}$$

Table 1: Power of Bung-DLX (V_{dd}=3.3V)

Datapath Width(bit)	P_c (mw)	P_s (mw)	P_{total} (mw)	Savings (%)
32	26.39	56.15	82.54	-
28	20.33	46.15	66.48	19.46
22	19.95	44.39	64.34	22.05
15	13.62	32.54	46.16	44.08
8	10.67	24.69	35.36	57.16

where
V_{dd}: Supply Voltage
$C(j)$: Load Capacitance of net j
$S(j)$: The average number of switching of net j per clock cycle
$E_{c(k)}$: Internal Power of Cell k
E_{mem} is estimated as follows:

$$E_{mem} = E_{ROM} + E_{SRAM} \quad (7)$$

$$E_{ROM} = e_{ROM} \times \sum_{i \in I} Cycle_i$$

$$E_{SRAM} = e_{Sr} \times Cycle_{load} + e_{Sw} \times Cycle_{store} \quad (8)$$

where
e_{ROM} : Energy per read access to ROM
$e_{Sr}(e_{Sw})$: Energy per read (write) access to SRAM
$Cycle_{load}(Cycle_{store})$: The number of read (write) accesses of SRAM

The access energy of memories (e_{ROM}, e_{Sr}, e_{Sw}) is obtained from the SPICE simulation of several memories with the different configurations. As the result, we have obtained the estimation models as follows:

$$e_{ROM} = 50.97 * b * \sqrt{N_{words}} + 1.4[pJ/cycle] \quad (9)$$

$$e_{Sr} = 24.9 * b * \sqrt{N_{words}} + 56[pJ/cycle] \quad (10)$$

$$e_{Sw} = 197 * b * \sqrt{N_{words}} + 369[pJ/cycle] \quad (11)$$

Where b is the word width of the memory and N_{words} is the word count.

5 Experiments and Results

In this section we present experiments and results based on several real applications to evaluate our proposed approach. We mainly illustrate how we use our approach to minimize energy consumption of MPEG-2 video decoder, a relatively large program.

In the experiments, we assumed the target system, a SOC chip, which consists a Bung-DLX processor, a ROM and a SRAM. Bung-DLX is a non-pipelined, simple RISC processor, which has several design parameters including the datapath width and the number of registers. All instructions are executed within a single machine cycle. The ROM and the SRAM are used as instruction memory and data memory respectively. These memories are generated by Alliance CAD System Ver. 2.0 with $0.5\mu m$ double metal CMOS technology. For simplicity, we assumed that no other core is integrated in the SOC chip.

5.1 Variable Size Analysis of MPEG-2

Our program is based on Mpeg2decode program from the MPEG Software Simulation Group. It is a player for MPEG-1 and MPEG-2 video bitstreams. Mpeg2decode is an implementation of an ISO/IEC DIS 13818-2 decoder,whose emphasis is on correct implementation of the MPEG standard and comprehensive code structure. We rewrote it in Valen-C with about 6650 lines. The MPEG-2 core consists of several function blocks such as a soft-core processor, IDCT blocks, a couple of motion estimation blocks, a motion compensation block, variable length encoding, decoding blocks and so on. The supply voltage that we used is 3.3V.

Table 2: Static analysis results(MPEG-2 decoder)

E.Size	N.of Variables	E.Size	N. of Variables
1bit	50	12bits	14
2bits	17	14bits	46
3bits	10	15bits	2
4bits	11	16bits	39
5bits	8	17bits	2
6bits	11	18bits	2
7bits	12	26bits	2
8bits	9	27bits	4
9bits	7	28bits	3
10bits	3	29bits	3
11bits	6	30bits	7
Total	5656bits	−34%	(8576bits)

Table 3: Dynamic Analysis Results(MPEG-2 decoder)

V.name	E.size	V.name	E.size
fn	5bits	g2nc	7bits
fl	12bits	rbx	12bits
sn	6bits	rby	12bits
nl	20bits	rec4s1	24bits
gb32l	20bits	rec4s4	24bits
gbl	20bits	rec4cs1	24bits
gbn	5bits	rec4cs4	24bits
g2ai	20bits	rechs1	24bits
g2asign	20bits	rechcs1	24bits
g2aincn	18bits	rec4as1	24bits
g2anc	6bits	rec4as4	24bits
gi	7bits	rechas1	24bits
gsign	20bits	rechas2	24bits
gincnt	6bits	rec4acs1	24bits
g2i	7bits	rec4acs2	24bits
g2sign	3bits	rechacs1	24bits
g2incnt	7bits	rechacs2	24bits
Toyal	1056bits	521bits	−52%

We analyzed the C source program of MPEG-2 video decoder and got the results of static variable analysis depicted in Table 2, and that of dynamic analysis shown in Table 3. From Table 2 and Table 3, we can see that there are many redundant bits in the variables of MPEG-2 video decoder. We got 34% reduction of bits from the static analysis and 52% from the dynamic analysis.

To verify our analysis results of variable size, we used the following model.

$$PSNR = 10 \times \log_{10}[\frac{1}{E} \times 255^2][dB] \quad (12)$$

where, $PSNR$: Ratio of pick signal to noise
E : Mean-suqare error

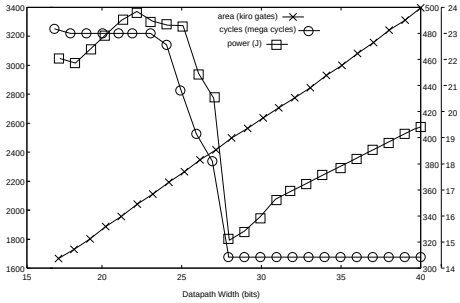

Figure 4: Energy consumption, execution cycles and area(gates) for MPEG-2 decoder

Table 4: Energy savings (V_{dd}=3.3V)

Applications	Energy Consumption(J)		
	No Opt.	Opt.	Savings
Lempel-Ziv	0.9517	0.4919	48.3%
ADPCM	1.181	0.9187	22.8%
MPEG2	156.42	139.66	10.7%

Our experimental results of $PSNR$ are $infinite$, so it shows that the variables, which are assumed according to the analysis results can work exactly as that of the source program of MPEG-2 video decoder.Therefore, our analysis results are verified.

5.2 Power and Performance Estimation

We report some experimental data concerning the use of our approach to reduce energy consumption. We obtain the cycle count using our instruction-level simulator. The input of the simulator is the assembly code, which is generated by the retargetable Valen-C compiler. Results of energy consumption include energy of a core processor, a data RAM and an instruction ROM. We use the energy consumption models in section 4.

Figure 4 shows the energy consumption, execution cycles and area(gates) of MPEG-2 video decoder, and Figure 5 describes the energy savings of our benchmarks, such as Lempel-Ziv algorithm, ADPCM encoder, and MPEG-2 AAC decoder and so on. Table 4 shows their energy savings. No Opt. means the original datapath width of Bung-DLX (32bits). Opt. is the datapath width where the whole system has the minimization energy consumption without performance loss. For Lempel-Ziv algorithm, we got energy savings of 48.3% at datapath width of 15bits, for ADPCM encoder, energy savings is 22.8% at datapath width of 19bits and for MPEG-2 video decoder, the energy savings is 10.7% at datapath width of 28bits. For different application, the number of variables is different and the effective size of variables is also different, therefore the optimal datapath width of minimal energy is different. For a given application, our approach just tries to take advantage of the characteristics of the application to reduce the energy consumption.

6 Conclusions

In this paper, we have proposed a system-level energy minimization approach through datapath width optimization, which can suit the complexity of embedded systems and stringent time-to-market constraints.

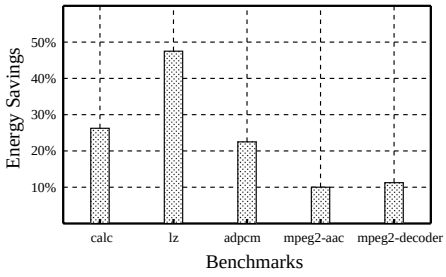

Figure 5: Energy savings of benchmarks

We also presented a set of algorithms that minimize energy consumption in system-level. We illustrated issues and tradeoffs involved in the design. Our experimental results show that for a given application we can reduce significantly the energy consumption by datapath width optimization. We have demonstrated energy savings without performance penalty range from about 10.7% to 48.3%, which based on a number of embedded applications. Extending parameter-tuning for low power to DSPs is our future work.

References

[1] *H.Yasuura,H.Tomiyama,A.Inoue,F.N.Eko, "Embedded System Design Using Soft-Core Processor and Valen-C", Journal of Information Science and Engineering, No.14,pp.587-603,August 1998.*

[2] *A.Inoue,H.Tomiyama,T.Okuma,H.Kanbara,and H.Yasuura, "Language and Compiler for Optimizing Datapath Width of Embedded Systems", IEICE Trans. Fundamentals, Vol. E81-A, No.12, pp. 2595-2604, Dec. 1998.*

[3] *F. N.Eko,A.Inoue, H.Tomiyama, H.Yasuura, "Soft-Core Processor Architecture for Embedded System Design",IEICE Trans.on Electronics, Vol.E81-C No.9,pp1416-1423, Sep.1998.*

[4] *H.Yamashita,H.Tomiyama,A.Inoue,F.N.Eko, T.Okuma,and H.Yasuura,"Variable Size Analysis for Datapath Width Optimization,"Proc.of Asia Pacific Conference on Hardware Description Languages(APCHDL'98), pp.69-74,July 1998.*

[5] *E.N.Eko and H.Yasuura, "A Cycle-Accurate Simulator Toolkit for Soft-Core Processors", Proc.of Asia Pacific Conference on cHip Design Languages (APCHDL'99), pp.11-16, October 1999.*

[6] *A.Inoue, T.Ishihara and H.Yasuura, "Flexible system lsi for embedded systems and its optimization techniques",Journal of Design Automation for Embedded System,5(2),2000.*

[7] *P.Landman and J.Rabaey,"Architectural Power Analysis:The Dual Bit Type Method", IEEE Transactions on VLSI Systems, Vol.3, No.2, June 1995.*

[8] *T.Sato, M.Nagamatsu, H.Tago, "Power and Performance Simulator:ESP and its Application for 100 MIPS/W Class RISC Design", IEEE Proc.of Symposium on Low Power Electronics, pp.46-47, 1994.*

[9] *I.Hong,D.Kirovski et al.,"Power Optimization of Variable voltage Core-Based Systems", IEEE Proc.of 35th. Design Automation Conference(DAC'98), pp.176-181,1998.*

[10] *M.Kamble, K.Ghost, "Analytical Energy Dissipation Models for Low Power Caches", International Symposium on Low Power Electronics and Design, pp.143-148, Aug.1997.*