Reducing Dynamic Power and Leakage Power for Embedded Systems

Cao, Yun
Department of Computer Science and Communication Engineering, Kyushu University

Yasuura, Hiroto
Department of Computer Science and Communication Engineering, Kyushu University

Reducing Dynamic Power and Leakage Power for Embedded Systems

Yun Cao Hiroto Yasuura
Department of Computer Science and Communication 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 system-level technique for embedded processor-based systems targeting both dynamic power and leakage power reduction using datapath width optimization. By means of tuning the design parameter, datapath width tailored to a given application requirements, the processors and memories are optimized resulting in significant power reduction, not only for dynamic power but also for leakage power. In our experiments for several real embedded applications, power reduction without performance penalty is reported range from about 14.5% to 59.2% of dynamic power, and 21.5% to 66.2% of leakage power.

1 Introduction

The increasing use of battery-operated portable computing and wireless communication systems makes power consumption a major concern in modern designs [1] [2]. Reducing power consumption hence becomes a crucial challenge for today's embedded system designers. Maximization of battery life is an obvious goal for these applications. Extensive researches for low power designs show that optimization of power consumption can be considered at all design levels from circuit level to system level [3] [4] [1] [5].

In CMOS digital circuits, power consumption consists of dynamic and static components. In circuits with a high supply voltage, a relatively high transistor threshold voltage can be used, therefore subthreshold current can be negligible. That is the common assumption of the existing techniques for power optimization [1] [4] [5]. However, low power applications have been driving the supply voltage to become lower and lower, which requires the device threshold to be reduced so as to satisfy performance requirements. This leads to dramatic increase of leakage current due to the exponential relationship between leakage current and threshold voltage. Consequently, leakage power (static power) is no longer negligible in low voltage circuits. Two implementations of Intel's Pentium III processor manufactured on Intel's $0.18\mu m$ process are good examples. They are the Pentium III 1.0 GHz B and the Pentium III 1.13 GHz [6]. The Intel datasheet lists the maximum core power consumption of the 1.0 GHz part at 33.0 watts and the deep sleep (i.e. leakage) power consumption at 3.74 watts. The 1.13 GHz processor has a total power consumption of 41.4 watts and leakage power consumption of 5.40 watts. While the total power has increased by only 25%, the leakage power has increased by 44% and comprises 13% of the total power consumption. The dynamic power consumption of the processor core varies significantly depending on the workload while the leakage power consumption is almost constant. Therefore, leakage power is even a larger percentage of the total power consumption on average. Reducing leakage power can be especially important to battery when a system is idle for a long time, such as for mobile phones.

The recent trend indicates that leakage power will likely contribute as much to total power as dynamic power in as little as two technology generations. Therefore optimization techniques for leakage power are also necessary. The device and circuits communities have been concerned with increasing leakage power for several generations. Reference [2] projects dual V_t(transistor threshold voltage) design technique will be widely used to reduce leakage power. As far as we know, prior work on power reduction at the system level has been focused almost entirely on dynamic power. In order to limit dynamic power consumption, techniques such as clocking gating [4] and cache subbanking [7] have been employed. The goal of these techniques is to reduce the number or frequency for switching devices. Optimization of the supply voltage to reduce the power/performance ratio is also performed [5], this has the added benefit of addressing dynamic power consumption, which is proportional to the square of the supply voltage. However, all of these techniques are hardly used to reduce leakage power. In this paper, we present datapath width optimization focusing on both dynamic power and leakage power

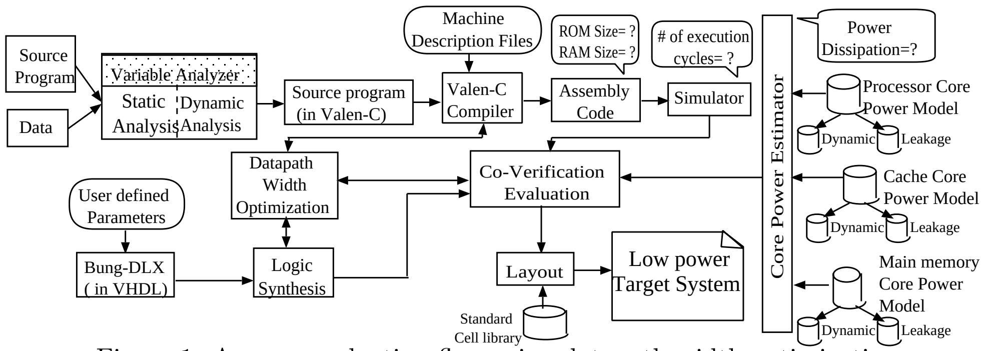

Figure 1: A power reduction flow using datapath width optimization

reduction for system-level design, while providing adequate performance level. Using the technique, system designers can control the design parameters, such as the datapath width of processors freely. Not only the dynamic power but also the leakage power of the whole system is drastically reduced by tuning the parameters of processors and memories tailored for the applications.

The rest of this paper is structured as follows: the next Section 2 presents our technique for dynamic power and leakage power reduction. Experiments and results are shown in Section 3. Finally, Section 4 concludes our work.

2 Reducing Dynamic & Leakage Power

Dynamic power consumption is described by the familiar formula, $P_{dyn} = \frac{1}{2} \cdot C \cdot V_{cc}^2 \cdot f$ C is the capacitance of switching nodes, which is roughly proportional to the number of switching devices. V_{cc} is the supply voltage, and f is the effective operating frequency (frequency times activity factor). Leakage power consumption is equal to the product of the supply voltage and the leakage current. To deal with the leakage power at system level, we start addressing leakage power model from a simple equation for estimation in [12], $P_{static} = V_{cc} \cdot N \cdot k_{design} \cdot \hat{I}_{leak}$ where V_{cc} is the supply voltage, N is the number of transistors, k_{design} is a design dependent parameter, and $\hat{I}_{leak}$ is a technology dependent parameter. This model enables high-level reasoning about the likely leakage power demands of alternative designs. The parameters of the leakage power model of the equation may be divided into two groups. The technology parameters are derived from measurements or simulations of individual devices. They are all dependent on a host of lower-level process parameters(e.g., oxide thickness and doping profiles) in complex ways. The design dependent parameters(V_{cc}, N and k_{design}) apply to groups of devices interconnected in a specific design style. Within certain constraints, they are independent of the process technology and may be varied independently. The model suggests different ways in which leakage power may be reduced. One obvious technique that can be employed is to reduce the total number of devices. However, finding opportunities to reduce the device count enough to impact power consumption without decreasing performance or functionality is difficult, normal design practices eliminate obvious redundancy. At system level the number of transistors (represented by N) can often be estimated. Design exploration to get optimized alternatives can be achieved by estimation without reaching the circuit design phase. N is only constrained by the functionality required of the circuit and the available area in which to implement it.

In this paper, we present datapath width optimization to reduce the number of devices(N) by tuning datapath width of processors for each application, to try to ease the problem of leakage power. Because reducing the number of devices directly reduces the switching capacitance(C) and effective operating frequency(f), dynamic power consumption is also reduced.

2.1 Datapath Width Optimization

In the design of consumer electronic systems, designers have to manage rapid increase of complexity of a target system with requirments on high-performance and low-power consumption under the tight constraint of short design time. Therefore, core-based solutions are proposed for embedded system design. We have developed a design platform for embedded core-based systems, which consists of a variable configuration processor called Bung-DLX [8], a multi-precision retargetable compiler called Valen-C retargetable compiler [9], a variable size analyzer [10] and a cycle-based simulator [11]. On the platform, designers have a freedom to determine the datapath width of processor to any bit. 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 reduce both dynamic power and leakage power of the whole embedded systems.

Optimizing datapath for dynamic and leakage reduction can be formulated as *to minimize* $P(w)$, *subject to* $Cycle(w) \leq C_{cst}$. Power $P(w)$ and cycle $Cycle(w)$ are functions of datapath width w, C_{cst} is the constraint on the execution cycle. $P(w)$ includes dynamic power $P_d(w)$ and leakage power $P_s(w)$.

Figure1 shows our design flow and Figure2 describes the overview of our power reduction algorithm. In the initial design phase of our approach, we design a system with a variable configuration processor(Bung-DLX), data RAMs, instruction ROMs and logic circuits. Then we analyze the effective bitwidth of each variable ($EWd(x_i)$) in a given application program (AP). After that, using the results of variable size analysis, we rewrite the application program in Valen-C language, in which we specify the word length of each variable satisfying accurate computation to reduce dynamic and leakage power consumed by redundant bits in the application program. After verifying the functionality of the initial design, we modify several design parameters of the variable configuration processor, including the datapath width w_i, the number of registers and the instruction set. We can tune up the variable configuration processor to reduce the power consumption and get the minimal dynamic power(P_{dMin}) and leakage power(P_{sMin}) while satisfying the system performance constraints($c_k \leq C_{cst}$).

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. 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 some upper bits are useless. C language provides three integer sizes, declared using the keywords *short*, *int* and *long*. The compiler designer determines the sizes of these integer types. We present Valen-C language, by which programmers can explicitly specify the required bitwidth of each integer data type, so it becomes possible to reduce the power of the datapath and the data memory consumed by the redundant bits. The value of the datapath width can be tuned 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. It is the smallest datapath width at which all instructions can remain single-precision. The datapath width of a processor strongly affects the power consumption of the whole system including the processor and 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. So trade-offs exist between datapath width and execution cycles. Trading off the power consumption and performance is an important work for datapath width optimization.

```
Input:
  source program : $AP$
      (variables : $x_i \in X = \{x_1, x_2, ..., x_n\}$, $1 \leq i \leq n$)
  input data : $D_{in}$
  the constraint of cycles : $C_{cst}$
Variable:
  datapath width $w_i \in W=\{w_1,w_2,...w_n\}$
Output:
  execution cycles $c_i \in C = \{c_1,c_2,...c_n\}$
  the minimal dynamic power $P_{dMin}$ when $c_k \leq C_{cst}$
  the minimal leakage power $P_{sMin}$ when $c_k \leq C_{cst}$
  the datapath width $w_k$ when $P_{d_k} = P_{dMin}$, $P_{s_k} = P_{sMin}$
Phase 1 : Variable Size Analysis
  $analyzer \leftarrow (AP, X, D_{in})$
  $return(EWd = \{EWd(x_1), EWd(x_2), ..., EWd(x_n)\})$
Phase 2 : Define Design Parameters for Bung-DLX
  datapath width $w_i$
  the number of registers $n_i$
Phase 3 : Valen-C program
  variable declaration of bitwidth $EWd(x_i)$
  compile the Valen-C source program for customized
        Bung-DLX at $w_i$
Step 4 : Datapath Width Optimization
  for $W \neq \emptyset$
  {
    estimate the execution cycles $c_i$
    estimate the dynamic power consumption $P_{dTot}$
    estimate the leakage power consumption $P_{sTot}$
    $P_{dMin} \leftarrow$ the minimal dynamic power when $c_k \leq C_{cst}$
    $P_{sMin} \leftarrow$ the minimal leakage power when $c_k \leq C_{cst}$
  }
  $return(P_{dMin}, P_{sMin}, w_k, c_k)$
```

Figure 2: Pseudo code of the algorithm for dynamic & leakage power reduction

2.2 Power Estimation Models

We assume that the target system consists a processor, a data RAM and an instruction ROM. The variable configuration processor Bung-DLX is used. The total power consumption, P_{Tot} of a system, is estimated as the summation of dynamic power P_{dTot} and leakage power P_{sTot}.

$$P_{Tot} = P_{dTot} + P_{sTot} \quad (1)$$
$$P_{dTot} = P_{dProc} + P_{dMem} \quad (2)$$
$$P_{sTot} = P_{sProc} + P_{sMem} \quad (3)$$

Dynamic power of a processor P_{dProc}, dynamic power of memories P_{dMem}, leakage power of a processor P_{sProc} and leakage power of memories P_{sMem} are estimated separately. We built the dynamic power and leakage power consumption models of Bung-DLX generated by HITACH 0.5 μm CMOS technology and the dynamic power and leakage power consumption models of memories generated by Alliance CAD System Ver.4.0 with 0.5 μm double metal CMOS technology.

The power consumption in static CMOS circuitry can be divided into static(leakage), dynamic and short-circuit power. Here we just focus on dynamic

Datapath Width(bit)	P_{ci} (mw)	P_{ns} (mw)	P_{dProc} (mw)	Saving (%)
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

Figure 3: Dynamic power of Bung-DLX (V_{cc}=3.3V)

Datapath Width(bit)	$V_{cc} = 5.0V$		$V_{cc} = 3.3V$	
	P_{sProc}	*Saving*	P_{sProc}	*Saving*
32	0.74	-	0.95	-
28	0.64	13.5%	0.80	15.8%
22	0.49	33.8%	0.61	35.8%
19	0.43	41.9%	0.47	61.1%
8	0.19	74.3%	0.20	79.0%

Figure 4: Leakage power of Bung-DLX (μw)

power(P_{dProc}) and leakage power (P_{sProc}). Dynamic power P_{dProc} consists of Cell Internal Power(P_{ci}) and Net Switching power(P_{ns}).

P_{dProc} is shown as follows:

$$P_{dProc} = \frac{1}{2} \times V_{cc}^2 \sum_{net} [C_j \times S_j + P_{ci_k} \times S_k] \quad (4)$$

V_{cc}: Supply voltage
C_j: Load capacitance of net j
S_j: The average number of switching of net j
P_{ci_k}: Internal power of cell k
S_k: The average number of switching of cell k
Leakage power P_{sProc} can be described as follows:

$$P_{sProc} = \sum_{\forall cell_k} P_{CellLeakage_k} \quad (5)$$

P_{sProc}: Total leakage power of a processor
$P_{CellLeakage_k}$: Leakage power of each cell k
P_{dProc} and P_{sProc} are obtained by using Synopsys Power Compiler. After several simulations, we obtained the empirical dynamic power model at several datapath widths for supply voltage $3.3V$ shown in Figure3, where power savings, *Saving* are got by comparing to the dynamic power consumption of the 32bits processor. The empirical leakage power model at several datapath widths for supply voltage V_{cc} of $5.0V$ and $3.3V$ are also obtained shown in Figure4, where power savings, *Saving* are got by comparing to the leakage power consumption of the 32bits processor.

P_{dMem} and P_{sMem} are estimated as follows:

$$P_{dMem} = P_{dROM} + P_{dSRAM} \quad (6)$$
$$P_{sMem} = P_{sROM} + P_{sSRAM} \quad (7)$$

$P_{dMem}(P_{sMem})$: Dynamic(leakage) power of memory
$P_{dROM}(P_{sROM})$: Dynamic(leakage) power of ROM
$P_{dSRAM}(P_{sSRAM})$: Dynamic(leakage) power of SRAM
P_{dROM}, P_{dSRAM}, P_{sROM} and P_{sSRAM} are obtained from the SPICE simulation of several memories with different configurations. We obtained the estimation models as follows:

$$P_{dROM} = 50.97 * b * \sqrt{N_{words}} + 1.4[pw] \quad (8)$$
$$P_{dSRAM} = 73.9 * b * \sqrt{N_{words}} + 142[pw] \quad (9)$$
$$P_{sROM} = 18.1 * b * \sqrt{N_{words}} + 0.08[pw] \quad (10)$$
$$P_{sSRAM} = 231.0 * b * \sqrt{N_{words}} + 1.1[pw] \quad (11)$$

Where b is the bit width of the memory and N_{words} is the number of words.

3 Experiments and Results

This section reports some experimental results concerning the use of our technique to reduce power consumption based on several real applications. We report both dynamic power and leakage power results. In the experiments, we assumed the target system, a SOC chip, which consists a Bung-DLX processor, a ROM and a SRAM. The ROM and the SRAM are used as instruction memory and data memory respectively. For simplicity, we assumed that no other core is integrated in the SOC chip. Four real embedded applications are used as benchmarks, which are Lempel-Ziv algorithm, ADPCM encoder, MPEG-2 AAC audio decoder, and MPEG-2 video decoder. The cycle count is used to evaluate performance obtained by using the simulator [11]. For dynamic power and leakage power estimation, models in Section 2.2 are used.

Figure 5 shows the estimation results for MPEG-2 video decoder. We analyzed the C source program of MPEG-2 decoder from the MPEG Software Simulation Group using our developed variable size analyzer and got the variable size of effective data width depicted in Figure5(a). This figure shows that there are a lot of variables having many unused bits in MPEG-2 decoder, which originally declared as "int" type. We got average 39% reduction of bits from the variable size analysis. The dynamic power consumption and execution cycles are described in Figure 5(b), and the estimation results of leakage power shown in Figure 5(c). From the figures, we can get the optimal datapath width, where the whole system has the minimization power consumption without performance penalty, 28bits for MPEG-2 video decoder.

Figure6 shows the results of the experiments employed our technique for the benchmarks. In the first table of Figure6 for dynamic power, column $No.Opt.DynamicPower$ including four columns, which show the results for original designs without using our technique. Column P_{dProc} shows the dynamic power consumption of processor Bung-DLX, column P_{dSRAM} is dynamic power of SRAM, column P_{dROM} is the dynamic power of ROM, and column P_{dTot} shows the total dynamic power consumption. The next four columns show the results using our op-

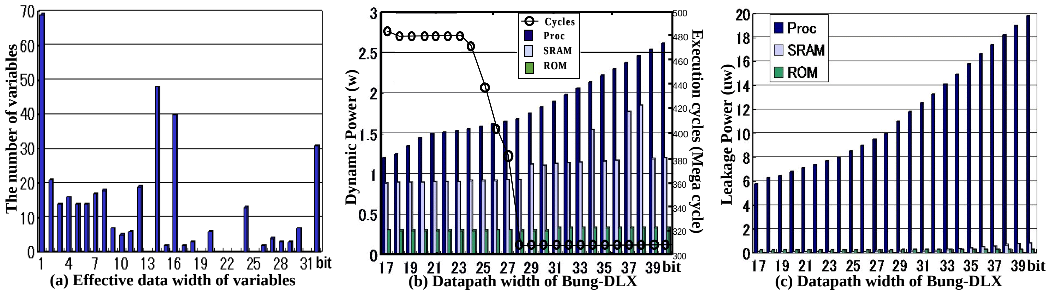

Figure 5: Results of MPEG-2 video decoder

Application	No. Opt. Dynamic Power(mw)				Opt. Dynamic Power(mw)				Sav. %
	P_{dProc}	P_{dSRAM}	P_{dROM}	P_{dTot}	P_{Proc}	P_{dSRAM}	P_{dROM}	P_{dTot}	
Lempel-Ziv	645.8	330.2	66.98	1.04w	208.3	184.6	31.4	424.3	59.2
ADPCM	155.5	82.5	118.5	356.5	77.5	51.28	70.3	199.1	44.2
Mpeg2AAC	589.3	247.6	337.23	1.17w	503.6	194.5	301.7	999.8	14.5
Mpeg2Video	2.05w	1.16w	349.68	3.56w	1.68w	930.72	305.97	2.91w	18.3

Application	No. Opt. Leakage Power(μw)				Opt. Leakage Power(μw)				Sav. %
	P_{sProc}	P_{sSRAM}	P_{sROM}	P_{sTot}	P_{sProc}	P_{sSRAM}	P_{sROM}	P_{sTot}	
Lempel-Ziv	3.8	36.7nw	59.7nw	3.90	1.28	19.8nw	24.9nw	1.32	66.2
ADPCM	0.95	8.8nw	0.11	1.06	0.48	5.7nw	61.7nw	0.55	48.1
Mpeg2AAC	3.1	28.3nw	0.30	3.43	2.3	21.5nw	0.26	2.59	21.5
Mpeg2Video	13.3	0.21	0.31	13.82	10.0	0.18	0.27	10.4	24.7

Figure 6: Power results for benchmarks

timization technique($Opt.$). The last column *Savings* shows the dynamic power reduction compared to the designs without using our technique($No.Opt.$).

The results of leakage power obtained are listed in the second table of Figure 6. Column $No.Opt.LeakagePower$ show the results for original designs without using our technique. Column P_{sProc} shows the leakage power for processor Bung-DLX, column P_{sSRAM} for SRAM, column P_{sROM} for ROM, and column P_{sTot} shows the total leakage power consumption. The next four columns show the results using our technique($Opt.$). The last column *Savings* shows the leakage power reduction compared to the designs without using our technique($No.Opt.$).

4 Conclusions

This paper proposes a system-level design technique focusing on both dynamic and leakage power reduction, which can suit the complexity of embedded systems and stringent time-to-market constraints. The presented technique of datapath width optimization can reduce both dynamic and leakage power consumption. Our experimental results show that for a given application we can reduce significantly the power consumption. We demonstrate power savings without performance penalty average about 34.1% of dynamic power, and 40.1% of leakage power, which based on a number of real embedded applications.

References

[1] L.Benini, R.Hodgson and P.Siegel. : *System-level Power Estimation And Optimization*, International Symposium on Low Power Electronics and Design, pp.173-178, Aug.1998.

[2] S. Borkar, "Low Power Design Challenges for the Decade", Proc. of Asia South Pacific Design Automation Conference, pp. 293-296, 2001

[3] C. Svensson and D. Liu, " 'Low Power Circuit Techniques,' in Low Power Design Methodologies", pp. 37-64, Kluwer Academic, Norwell, MA, 1996.

[4] V.Tiwari, D.Singh, S.Rajgopal, G.Mehta, R.Patel, F.Baez, "Reducing Power in High-Performance Microprocessors", Proc. of the 35th Design Automation Conference, pp. 732-737, 1998.

[5] 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.

[6] Intel Corporation. Pentium III Processor for the SC242 at 450 MHz to 1.13 GHz Datasheet, pp. 26-30.

[7] U.Ko and P.Balsara. : *Energy Optimization of Multilevel Cache Architectures for RISC and CISC Processors*, IEEE Transactions on VLSI Systems, vol.6, no.2, pp.299-308, June 1998.

[8] 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.

[9] 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.

[10] H. Yamashita, H. Yasuura, F. N. Eko, and Yun Cao, "Variable Size Analysis and Validation of Computation Quality", Proc. of Workshop on High-Level Design Validation and Test, pp.95-100, 2000.

[11] 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.

[12] J. Adam Butts and Gurindar S. Sohi, "A Static Power Model for Architects", Proc. of the 33th Annual International Symposium on Microarchitecture, pp. 191-201, 2000.