社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－110 (10)
2003／5／16

# Variable Pipeline Depth Processor for Energy Efficient Systems

Akihiko Hyodo Masanori Muroyama and Hiroto Yasuura

Department of Computer Science and Communication Engineering, Kyushu University

6-1 Kasuga-koen Kasuga, 816-8580 Japan

E-mail: {akihiko, muroyama, yasuura}@denshi.ac.jp

**Abstract** This paper presents a variable pipeline depth processor, which can dynamically adjust its pipeline depth and operating voltage at run-time, we call dynamic pipeline and voltage scaling (DPVS), depending on the workload characteristics under timing constraints. The advantage of adjusting pipeline depth is that it can eliminate the useless energy dissipation of the additional stalls, or NOPs and wrong-path instructions which would increase as the pipeline depth grow deeper in excess of the inherent parallelism. Although dynamic voltage scaling (DVS) is a very effective technique in itself for reducing energy dissipation, lowering supply voltage also causes performance degradation. By combining with dynamic pipeline scaling (DPS), it would be possible to retain performance at required level while reducing energy dissipation much further. Experimental results show the effectiveness of our DPVS approach for a variety of benchmarks, reducing total energy dissipation by up to 40.4% with an average of 27.4% without any effect on performance, compared with a processor using only DVS.

**Keyword** Energy Efficient Design, Variable Pipeline Depth, Dynamic Pipeline and Voltage Scaling, Optimal Pipelining

## 1. Introduction

In recent years, energy efficiency has become a major concern not only for battery powered or wireless systems, but also in the case of high-performance microprocessors such as multimedia and digital signal processors. Traditionally, microprocessor designs have focused almost entirely on performance, but these shifting constraints are requiring systems to be energy aware in addition to performance, adapting to both application behavior and user requirements. Therefore, it has become an important research goal to develop the methodology of energy efficient design which have high-performance and low energy consumption as well.

Until recently, a large number of energy reduction techniques have been proposed at almost all levels of design hierarchy from the system level to the device level. Above all things dynamic voltage scaling (DVS) is one of the most effective known techniques to lowering energy dissipation due to the quadratic effect on the switching energy. The early works on DVS include theoretical studies [8] and simulations on the potential of DVS techniques [14, 15]. Since then, the practical implementation of DVS technique have already been archived on some commercial processors such as Intel's Strong ARM SA-110 [3], Transmeta's Crusoe [4]. However, DVS technique has the problem that as the supply voltage becomes lower, the circuit delay increases and the performance degrade [10]. Thus these approaches using DVS are unfortunately ineffective when tight deadlines are present in systems.

One way to maintain throughput while lowering supply voltage is to apply pipelining, which is the most popular fashion to increase the performance in modern microprocessor designs. Indeed increasing the pipeline depth is very effective for performance, but this may not always be energy efficient because too deep pipelining likely causes a great deal of useless energy dissipation of additional stall cycles brought by hazards. Current microprocessor designs decide the specific pipeline structure at design time based on the configuration that archives the best overall performance over a range of target applications. Previously, many researchers have addressed the issue of the optimal pipelining both theoretically [12, 16] and by simulation [17], which provide the way to determine the optimum design point of the pipeline depth that gives the best performance. However, fixing the pipeline structure may lead to a loss of an opportunity of further speedup or saving waste of extra energy when running a program which requirements are not well-matched to the particular pipeline organization chosen. To solve this problem, we introduce the concept of dynamic pipeline scaling (DPS).

In this paper, we present a novel dynamically reconfigurable processor, called variable pipeline depth processor, which can alter both pipeline depth and the supply voltage at run-time, termed dynamic pipeline and voltage

scaling (DPVS), considering workload characteristics and user requirements. The combination of DPS and DVS makes it possible to provide desired performance with the minimal amount of useless energy dissipation.

The remainder of this paper is organized as follows. The motivation and model underlying DPVS are discussed in section 2, followed by evaluations are in section 3, and our conclusions in section 4.

## 2. Motivations and Model

This section shows the possible scenario to illustrate the motivation for DPVS technique, and the energy model as a function of the pipeline depth and supply voltage to evaluate proposed architecture.

### 2.1. Motivations for DPVS

Several approaches in the most recent literature on energy efficient architecture point out that there exists a wide variation from one application to another, as well as between different parts of the same application both in terms of the inherent parallelism and in terms of the necessary resources to sustain a given performance level. The work of [13] showed that the amount of instruction level parallelism (ILP) within a single application varies by up to a factor of three. It should be realized that there is a clear and significant difference in the optimal pipeline depth among different types of workloads. When running programs which contain causes of hazard a lot on the deeper pipeline, it may not be possible to archive performance improvement than expected, and what is worse just raising not a little extra energy dissipation.

As was noted previously, a large number of researches have been done about optimal pipelining in terms of processor performance [12, 16, 17], however, all of them discuss on the premise that the pipeline depth is fixed at the design time. Obviously it may not be energy efficient to keep a fixed pipeline depth over a wide range of workloads since the exploitable ILP vary depending on its characteristics. For this reason, our approach which dynamically adapts the pipeline depth to the certain workload characteristics (e.g., control flow complexity, data dependency) which inherently have an influence on the cycle per instruction (CPI) could archive the improvement of energy efficiency.

To illustrate the key point of the proposed DPVS approach, we present a possible scenario as a motivational example, where several tasks are run on the given scalar processors, shown in Fig. 1. The processor taxonomy in Fig.1 is similar to [6], additionally assuming the base machine has the optimum pipeline depth over the whole of applications.

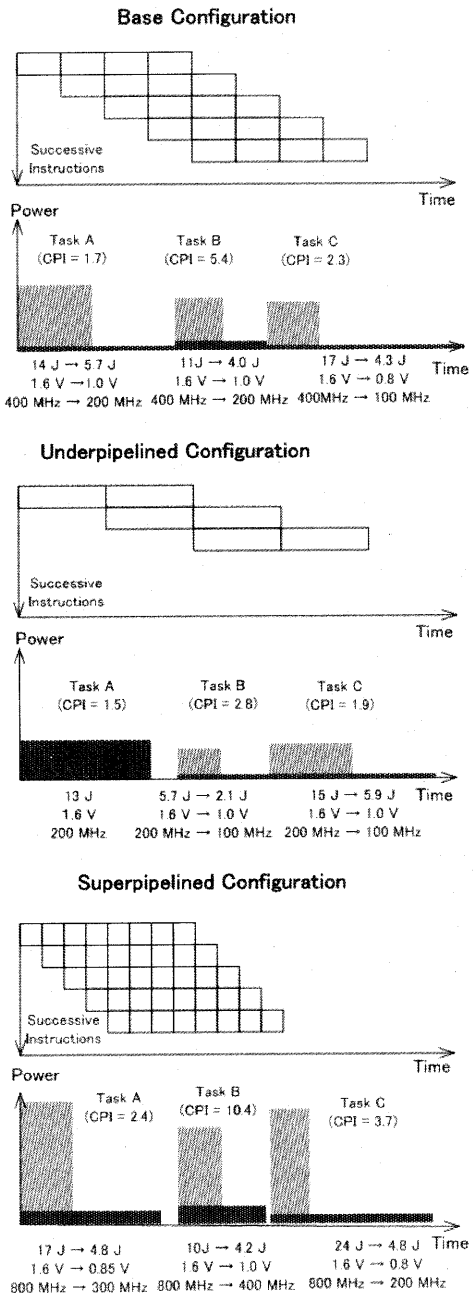


Fig. 1: Examples of energy reduction using DVS technique on several pipeline configurations.

For the reasonable scenario, the example in Fig.1 considers the availability of a discrete and finite set of voltage values that can be chosen [8], along with the corre-

sponding frequencies derived from the information available for Intel's XScale processor [11] letting the maximum supply voltage level be 1.6V. Each task can be executed immediately after its arrival and is required to be finished by the arrival time of the next task. The characteristics of each task A, B, and C are modeled after the algorithm of bit counting, quick sorting, and string searching. The assumption is that one can change the voltage level only once during the execution of a task, and any latch and adaptation overheads are not considered.

Table 1 shows the total energy dissipation of each machine configuration. With the shutdown technique, the system will operate at fixed voltage level of 1.6V. By using DVS, the tasks can be run at the scale the voltage down, resulting in the total energy reduction of 66% in the base configuration, 38% in underpipelined configuration, and 73% in superpipelined configuration. With the variable pipeline depth processor, one can choose the lowest energy pipeline configuration on each task, which allows the further reduction of total energy dissipation by up to 19% for this example.

Table 1: Energy dissipation of each pipeline configuration to illustrate the motivation for DPVS

| *Configuration* | | *Task A* | *Task B* | *Task C* | *Total Energy* |
|---|---|---|---|---|---|
| Base Pipeline | 1.6 V | 14 J | 11 J | 17 J | 42 J |
| | DVS | 5.7 J | 4.0J | **4.3 J** | 14 J |
| Under Pipeline | 1.6 V | 13 J | 5.7 J | 15 J | 33.7 J |
| | DVS | 13 J | **2.1 J** | 5.9 J | 21 J |
| Super Pipeline | 1.6 V | 17 J | 10 J | 24 J | 51 J |
| | DVS | **4.8 J** | 4.2 J | 4.8 J | 13.8 J |
| DPVS Model | | **4.8 J** | **2.1 J** | **4.3 J** | **11.2 J** |

## 2.2. Energy Model

The dominant source of energy dissipation in a digital CMOS microprocessor is the dynamic dissipation, which is mainly the charging and discharging of the load capacitances. The dynamic energy dissipation is equal to

$$E_{dynamic} = T_{exec} \cdot \sum_{all\_gates} C_L(k) \cdot Swit(k) \cdot V_{DD}^{\ 2} \qquad (1)$$

where $T_{exec}$ is the total execution time, $C_L$ is the load capacitance of a gate $g_k$, $Swit(k)$ is the switching count of $g_k$ per unit time, and $V_{DD}$ is the supply voltage. It can clearly be said that lowering $V_{DD}$ is the most effective way to reduce the energy dissipation. However, lowering $V_{DD}$ causes increase of circuit delay. The circuit delay can be estimated by using

$$\tau(V_{DD}) = K \cdot C_L \frac{V_{DD}}{(V_G - V_T)^{\alpha}} \qquad (2)$$

where $\tau\ (V_{DD})$ is the propagation delay of CMOS transistor for supply voltage $V_{DD}$, $V_T$ is the threshold voltage, $V_G\ (\sim V_{DD})$ is the voltage of input gate, $\alpha$ is the velocity saturation coefficient and $K$ is the device parameter which depends on the transconductance and the width to length ratio. When transistors are not velocity saturated $\alpha$ =2, as transistors become more velocity saturated $\alpha$ decreases towards 1 and in advanced MOSFETs is about 1.3. Eq. 1 and 2 indicate that there is the essential trade-off between circuit delay and supply voltage. When we assume the dynamic energy is the most dominant and the gates $g_k$ of the circuit form a collective switching capacitance $C$, and for roughly estimation, let the operating clock frequency $F(n,V_{DD})=n/M\ \tau\ (V_{DD})$, where $n$ is the number of pipeline stages, $M$ is the total logic depth of the circuit, the energy dissipation can be expressed as

$$\begin{aligned} E(n,V_{DD}) &= T_{exec}(n,V_{DD}) \cdot C \cdot F(n,V_{DD}) \cdot V_{DD}^{\ 2} \\ &= IC \cdot CPI(n) \cdot C \cdot V_{DD}^{\ 2} \end{aligned} \qquad (3)$$

where $CPI(n)$ is clock cycle per instruction, $IC$ is the total instruction count committed. However, the energy dissipation is not a good enough measure for evaluating energy efficiency, by lowering the supply voltage and the number of pipeline stages, the energy dissipation can be easily reduced with increasing the execution time. Thus when using the energy dissipation as a metrics of estimating energy efficiency, we should think about meeting the deadline ($T_{deadline}$) at the same time. This constraint is expressed as $CPI(n)/F(n,V_{DD}) \leqq T_{deadline}$ -- (4) Our aim is to minimize $E(n,V_{DD})$ subjected to Eq.4 by adapting the combination of pipeline depth and supply voltage to the program characteristics.

# 3. Evaluation

## 3.1. Simulation Framework

We evaluate our approach using the Mibench embedded benchmark suite [7], which is a set of 35 commercially representative embedded programs divided into six categories including: Automotive and Industrial Control,

Consumer Devices, Office Automation, Networking, Security, and Telecommunications, shown in Table 2.

Table 2: Simulated Benchmark Suites

| *Category* | *Number of Bench-marks* | *Description* |
|---|---|---|
| Auto./ Industrial | 6 | basic math, bit counting, sorting, shape recognition |
| Network | 2 | shortest path calculation, tree and table lookups |
| Security | 7 | data encryption, decryption and hashing |
| Consumer | 8 | jpeg, MP3encode/decode, color image conversion, image dithering, color palette reduction, HTML typesetting |
| Office | 5 | text manipulation |
| Telecomm | 7 | voice codec, checksum, frequency analysis |

The architectural simulators used in this study are derived from the SimpleScalar/ARM version 2.0 tool set [9], a suite of functional and timing simulation tools for the ARM ISA. Simulated processor configuration is modeled after Intel's SA-1 StrongARM pipeline [3], found in SA-11xx series of embedded microprocessors, shown in Table3.

Table 3: Simulated processor configurations

| *Parameter* | *Value* |
|---|---|
| Fetch queue size | 2 instructions |
| Branch prediction | Not-taken |
| Fetch & Decode width | 1 instrutions/cycle |
| Functional units | 1 Int ALU,<br>1 FPMult,<br>1 FP ALU |
| Pipeline depth (variable) | 1, 2, 4, 8, 16 |
| L1 I-cache | 16k, 32-way |
| L1 D-cache | 16k, 32-way |
| L2 Cache | None |
| Memory Bus Width | 4 –byte |

### 3.2. Experimental Results

We have performed extensive simulations of the DPVS approach to assess the effectiveness in reducing energy dissipation. The assumption is that the deadline time is loose enough to apply the DVS technique and the supply voltage can be ideally altered, that is, any voltage value is available and chosen, assuming any adaptation overhead not considered.

The energy dissipation, as mentioned previously, can be estimated by using the equation $E(n,V_{DD}) \propto CPI(n)V_{DD}^2$.

Thus, considering not only supply voltage but also CPI is important for energy reduction. In the following, we present the elasticity of CPI and lower bound of voltage as a function of pipeline depth at fixed throughput.

In Fig 5 we report the elasticity of CPI, which differs vastly depending on the workload characteristics of each category. As shown in Fig. 5, the average CPI of network category benchmarks is the quadruple of that of security category on 16-stage pipeline configuration, while within double on 1 to 8-stage pipeline configurations. This means that the deeper pipeline is very effective in terms of performance, but at the same time exposed the risk of rapidly increase of the useless works than shallower pipeline. Since the current microprocessor design is inclined to implement the extremely deeper pipeline, it would be said that the concept of DPVS would become more important and effective in that situation.

We also show in Fig. 6 the normalized supply voltage required to sustain the throughput when altering the pipeline depth. As it can be seen, the transition of the required supply voltage is almost saturated between 4 and 8 stages in all categories, but there is a slight variation of the saturated voltage level.

Fig. 7 shows the normalized values for EPI of each application category. As shown in Fig. 7, there is a variation of the optimal pipeline depth which archives the lowest energy dissipation. For almost all benchmarks the optimal number of pipeline stages is 4 or 8, while some are 16.

The results presented so far show that obviously the performance/energy characteristic of each category is differing. Thus the effect of adapting the pipeline depth to the optimal is very useful when running a variety of programs on a processor. Additionally, we show the fluctuation of the optimal pipeline depth at each application category in Table 5 and as the breakdown of Table 5, we also show that at each benchmark program in Table 6. From Table 5, we can see the optimal number of pipeline stage is 4 if the pipeline depth is fixed over a wide range of benchmarks. So assuming the pipeline depth is 4, Table 6 shows the maximum energy saving of 40.4%, achieved in quick sort benchmark.

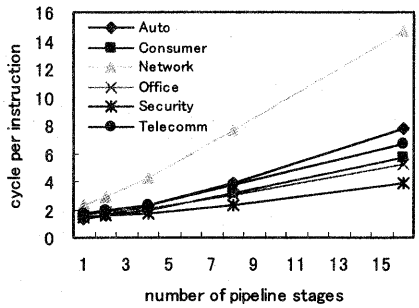


Fig. 5 Cycle per instruction as a function of pipeline depth

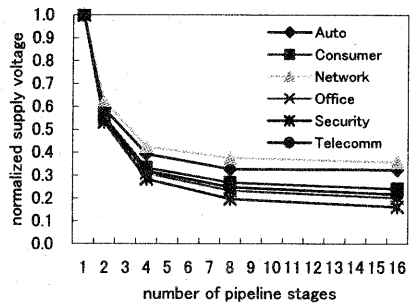


Fig. 6 Normalized supply voltage level required to sustain the throughput as a function of pipeline depth

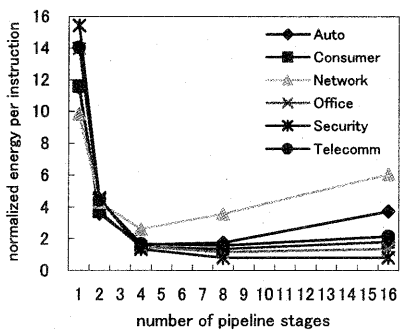


Fig. 7 Normalized energy per instruction as a function of pipeline depth under the fixed throughput

Table 5: Normalized energy dissipation of each embedded category when altering the number of pipeline stage 1, 2, 4, 8, and 16 under the fixed throughput altering the supply voltage ideally

| *Bench-mark Category* | *Number of the pipeline stages* | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| Indus-trial | 11.51 | 3.58 | **1.61** | 1.74 | 3.72 |
| Network | 11.60 | 3.71 | 1.47 | **1.35** | 1.81 |
| Security | 9.85 | 4.30 | **2.60** | 3.56 | 6.05 |
| Con-sumer | 13.98 | 4.44 | 1.51 | **1.16** | 1.35 |
| Office | 15.42 | 4.56 | 1.34 | **0.79** | 0.80 |
| Tele-comm. | 14.05 | 4.43 | 1.66 | **1.56** | 2.15 |
| Average | 12.74 | 4.17 | **1.69** | 1.70 | 2.65 |

Table 6: Normalized energy dissipation of each benchmark, which is the breakdown of each category shown in Table 5

| | 1stage | 2stage | 4stage | 8staeg | 16stage |
|---|---|---|---|---|---|
| bitcount | 13.8863 | 4.3223 | 1.3254 | **0.8873** | 0.9586 |
| qsort | **1.9094** | 1.9355 | 3.2052 | 5.4376 | 15.0462 |
| susan.corners | 14.8158 | 4.0001 | 1.1422 | **0.7456** | 0.7993 |
| susan.edges | 13.4667 | 3.7583 | 1.1479 | **0.7486** | 0.7857 |
| susan.smoothings | 13.4623 | 3.8976 | 1.2341 | **0.8999** | 1.0072 |
| jpeg.decode | 14.9251 | 4.2202 | 1.2562 | **0.9029** | 0.9696 |
| jpeg.encode | 14.6007 | 4.3964 | 1.3364 | **0.9702** | 1.0893 |
| lame | 8.8058 | 3.1961 | **1.6255** | 1.6746 | 2.3314 |
| mad | 11.0197 | 3.6385 | **1.4556** | 1.4360 | 2.0020 |
| tiff2bw | 7.9715 | 2.6370 | **1.5769** | 1.5838 | 2.1988 |
| tiff2rgba | 8.2499 | 3.3064 | **1.7710** | 2.0994 | 3.5338 |
| tiffdither | 14.7611 | 4.3672 | 1.2993 | 0.8935 | **0.8884** |
| tiffmedian | 12.4897 | 3.9871 | 1.4665 | **1.2094** | 1.4431 |
| dijkstra | 12.6812 | 4.1241 | 1.4112 | **1.2493** | 1.6004 |
| patricia | 7.0231 | 4.4758 | **3.7852** | 5.8785 | 10.5018 |
| ispell | 14.6975 | 4.6375 | 1.4811 | **1.0766** | 1.2108 |
| rsynth | 12.4582 | 3.6151 | 1.3434 | **0.9678** | 1.0722 |
| sphinx | 16.7356 | 5.1684 | 1.5758 | **1.1052** | 1.2109 |
| stringsearch | 12.0301 | 4.3287 | 1.6331 | **1.4901** | 1.8876 |
| blowfish.decode | 16.3114 | 5.1526 | 1.5999 | **1.0013** | 1.0419 |
| blowfish.encode | 16.3627 | 5.1647 | 1.6028 | **1.0011** | 1.0393 |
| rijndael.decode | 15.1662 | 4.2152 | 1.1720 | 0.6413 | **0.6174** |
| rijndael.encode | 15.4218 | 4.2852 | 1.1904 | 0.6652 | **0.6417** |
| sha | 13.8804 | 4.0039 | 1.1497 | 0.6883 | **0.6716** |
| CRC32 | 16.7952 | 5.5106 | 1.7669 | **1.1226** | 1.1772 |
| FFT.inverse | 14.3855 | 4.3467 | 1.3864 | **0.9678** | 1.0558 |
| FFT | 14.7003 | 4.4159 | 1.3724 | **0.9221** | 0.9765 |
| adpcm.decode | 15.3829 | 4.0440 | 1.1148 | **0.7080** | 0.7101 |
| adpcm.encode | 16.1001 | 4.1927 | 1.0919 | **0.6773** | 0.7164 |
| gsm.decode | 13.9096 | 4.1241 | 1.2790 | **1.0260** | 1.1753 |
| gsm.encode | 7.0650 | 4.3886 | **3.6043** | 5.5538 | 9.2308 |

In the end, we show the effectiveness of our approach to compare the energy dissipation of the fixed pipeline depth processor, and category specific, that is, the category level optimized pipeline depth processor, and the variable pipeline depth processor, shown in Table 7. We have observed 27.4 % energy saving results.

Table 7: Normalized energy and reduction ratio of each processor type under the fixed throughput, supposing supply voltage can be altered continuously

| *Processor Type* | *Energy* | *Reduction Ratio* |
|---|---|---|
| Fixed pipeline depth (4-stages ) processor | 1.6989 | 0.0000 % |
| Category specific (optimized) pipeline depth processor | 1.5138 | 10.8930 % |
| Variable pipeline depth processor | **1.2331** | **27.4176 %** |

## 4. Conclusion

Novel energy efficient processor architecture of dynamically adjusting the pipeline depth and the supply voltage, depending on both workload characteristics and user requirements is proposed. From the analytical energy model described in section 2.2, it could be said the pipeline depth adaptation gives lower benefits than DVS when used in isolation under loose timing constraints, but DPS gives mixed benefits when used combining with DVS, which we call DPVS.

We have demonstrated the further energy saving opportunity is widely open by using DPVS from the experimental results on a variety of benchmarks and have observed a decrease of up to 56% with an average of 27% in energy dissipation, compared with the variable voltage processor with fixed pipeline depth all the time during executing benchmarks.

## References


[1] D. Albonesi, "Dynamic IPC/Clock Rate Optimization", In *International Symposium on Computer Architecture*, pages 282-292, July 1998.

[2] A. Klauser, D. Grunwald, S.Manne, "Pipeline gating: Speculation control for energy reduction", In *International Symposium on Computer Architecture*, pages 132-141, July 1998.

[3] Intel Corporation, "Intel StrongARM SA-110 Microprocessor Datasheet", 1999.

[4] Transmeta Corporation, "The Technology Behaind the Crusoe Processor Whitepaper", 2000.

[5] D. Burger, T. M. Austin, "The SimpleScalar Tool Set, Version 2.0", *Computer Architecture News*, pages 13-25, June 1997.

[6] N. P. Jouppi, "The Nonuniform Distribution of Instruction-Level and Mahine Parallelism and Its Effect on Performance", *IEEE Transactions on Computers*, December 1989.

[7] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite", In *IEEE 4th Annual Workshop on Workload Characterization*, December 2001.

[8] T. Ishihara, H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors", In *International Symposium on Low Power Electronics and Design*, pages 197-202, August 1998.

[9] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0", Technical Report TR-97-1342, University of Wisconsin Madison, CS Department, June 1997

[10] A. Chandrakasan, S. Sheng, R. Brodersen, "Low power CMOS digital design", *IEEE Journal of Solid-State Circuits*, April 1992.

[11] Intel XScale Microarchitecture, http://developer.intel.com/design/intelxscale/benchmarks.htm.

[12] S. R. Kunkel and J. E. Smith, "Optimal pipelining in supercomputers", In *International Symposium on Computer Architecture*, pages 404-411, June 1986.

[13] D. W. Wall, "Limits of instruction-level parallelism", In *International Conference on Architectural Support for Programming Languages and Operating Systems*, November 1991.

[14] K. Govil, E. Chan, H. Wasserman, "Comparing algorithms for dynamic speed-setting on a low-power CPU", In *International Conference on Mobile Computing and Networking*, pages 13-25, November 1995.

[15] T. Pering, T. Burd, R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", In *International Symposium on Low Power Electronics and Design*, pages 76-81, August 1998.

[16] A. Hartstein, T. R. Puzak, "The optimum pipeline depth for a microprocessor", In *International Symposium on Computer Architecture*, pages 7-13, May 2002.

[17] E. Sprangle, D. Carmean, "Increasing processor performance by implementing deeper pipelines", In *International Symposium on Computer Architecture*, pages 25-34, May 2002.

[18] C. H. Hsu, U. Kremer, "Compiler-directed dynamic voltage scaling based on program regions", Technical Report DCS-TR-461, Department of Computer Science, Rutgers University, November 2001.

[19] C. H. Hsu, U. Kremer, M. Hsiao, "Compiler directed dynamic frequency/voltage scheduling for energy reduction in microprocessors", In *International Symposium on Low Power Electronics and Design*, pages 275-278, August 2001.