Exploration of Associative Power Management with Instruction Governed Operation for Ultra-low Power Design

Tianyu Jia, Yuanbo Fan, Russ Joseph, and Jie Gu

Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, IL, USA
{tianyujia2015, yuanbofan2012}@u.northwestern.edu, rjoseph@eecs.northwestern.edu, jgu@northwestern.edu

ABSTRACT

This paper explores a novel associative low power operation where instructions govern the operation of on-chip regulators in real time. Based on explicit association between long delay instruction patterns and hardware performance, an instruction based power management scheme is developed with energy models formulated for deriving the energy efficiency of the associative operation. The proposed system scheme is demonstrated using a low power microprocessor design with an integrated switched capacitor regulator in 45nm CMOS technology. Simulations on benchmark programs show a power saving of around 14% from the proposed scheme. A novel compiler optimization strategy is also proposed to further improve the energy efficiency.

Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Microprocessor/microcomputer applications.

General Terms

Design, Management, Experimentation.

Keywords

Energy efficient computing, integrated power management, low power design, switching regulator, compiler optimization

1. INTRODUCTION

The technology scaling of CMOS integrated circuits (IC) has slowed down in recent years as the conventional CMOS technology approaches its fundamental limit [1]. As the benefits of technology scaling become more expensive to realize, innovative systematic approaches for low power design become crucial to solve the energy bottleneck of many emerging applications, such as wearable electronics, Internet-of-Things, and biomedical devices [2]. Because the power consumption of conventional Very Large Scale Integrated (VLSI) circuits are mainly determined by the operating voltages, supply voltage scaling has been used as a primary method for achieving low power operation. For example, to achieve ultra-low power consumption, tremendous efforts have been put into designing circuits operating at sub-threshold or near-threshold voltages where an optimum energy consumption can be achieved [3-4]. Other advanced circuit techniques such as Razor use error detection mechanisms to remove design margin for runtime variation achieving a 10~30% power saving beyond conventional low power design techniques [5-6].

At the system level, dynamic voltage and frequency scaling (DVFS) has been widely utilized to explore the optimal tradeoff between performance and power [7]. In traditional DVFS, highly efficient switching voltage regulators are deployed on the board shared among multiple chips in order to reduce the silicon costs of electronic components. The traditional switching regulator, buck regulator or switched capacitor regulators normally operate at a switching frequency of several hundreds of kHz to a few MHz limiting its response time to microseconds [8]. As a result, previous DVFS schemes are only controlled at the system level with coarsely defined power states and thus are not capable of performing DVFS down at program level with fine granularity [9]. In recent years, the new trend of integrating numerous on-chip regulators for multi-core processors provides significant flexibility for energy optimization. For example, 48 fast response (sub-ns) regulators with 2 regulators for each logic core and cache were deployed in the 12 cores of IBM Power 8 processor to achieve fast DVFS [10]. Meanwhile, highly efficient on-chip switching regulators have been demonstrated with high configurability and fast response within 2~3ns or even sub-ns [11-12]. Such a fine grid on-chip voltage scaling capability introduces new opportunities for low power electronic design. For example, a physical model and optimization methodology for on-chip switched capacitor regulator was developed to optimize the deployment of on-chip regulators for higher energy efficiency [13]. An ultra-dynamic scheme was proposed to change supply voltage in a multi-V_{dd} configuration using different power switches, which allows the supply voltage to switch within a few nanoseconds leading to enhanced flexibility for DVFS [14]. However, that scheme requires generation and routing of multiple supply voltages to the digital logic and generates large design overhead. While the majority of current energy optimization methodology for power management has remained at system level, a few previous works also explored architecture and circuit level co-optimization based on sophisticated insight into software programs. For example, a previous study shows that significant amount of resonant noise can be removed if the existence of critical instructions can be predicted in a pipeline leading to 10% performance improvement [15]. A Razor based scheme was proposed to reduce timing error rate based on instruction type leading to 80% performance penalty in timing error recovery [16].

Different from the above work, this paper, for the first time, proposes to engage the control of on-chip regulator with the individual instructions inside the software program for ultra-low

DAC '16, June 05-09, 2016, Austin, TX, USA

© 2016 ACM. ISBN 978-1-4503-4236-0/16/06 $15.00

DOI: http://dx.doi.org/10.1145/2897937.2898021

voltage operation. Because a typical clock period of 10~100ns at near-threshold operation exceeds the response speed of several nanoseconds from an on-chip switching regulators, an instruction driven voltage scaling scheme can be utilized to achieve extra energy saving unobtainable from existing low power techniques. This work uses an ARM processor design to evaluate the design tradeoff and energy benefits for the proposed associative power management scheme. The contribution of this work is summarized below: (1) Based on large amount of instruction level timing analysis, a classification of critical instructions was developed to interpret the performance variation of software instructions; (2) An instruction governed power management scheme with supporting circuits was designed to take advantage of the significant amount of instruction level performance variation; (3) An energy model verified by transistor level simulation was derived to quantify the energy saving as well as overhead from the proposed scheme; (4) A novel compiler assisted program optimization method is also proposed to further improve the energy saving benefit.

2. INSTRUCTION AND PERFORMANCE ASSOCIATION

2.1 Preliminary Performance Observation on Low Power Microprocessor

ARM processor has been widely used in low power computing platforms. A single-issue ARMv5 processor is used in this paper as our test vehicle for the proposed scheme due to its popularity in low power applications and its relatively simple structure [17]. The pipeline architecture of ARMv5 processor used in this paper is shown in Fig. 1, which has pipeline stages including instruction fetch (IF), instruction decode (ID), operand fetch (OF), execution (EX), Memory (MEM) and write back (WB). Because this work only focuses on energy improvement of the logic circuits, behavioral models for instruction and data caches are used. Following the Instruction Set Architecture (ISA) defined for ARMv5 architecture, the target pipeline is designed and synthesized using commercial EDA design tools in a 45nm CMOS technology. The design has a nominal supply of 1.1V, operating speed of 1GHz and dynamic power consumption of 54mW. The design is then evaluated at ultra-low voltage condition of 0.5~0.55V using both spice level simulation and static timing analysis with timing library characterized at low voltages. For cycle-by-cycle performance evaluation, the timing is checked with Synopsys VCS gate level simulation which is run alongside the Gem5 simulator [18]. Our workload includes programs in MiBench from every category (automotive, networking, consumer, office, security, and telecommunication) to evaluate the performance of the proposed scheme [19].

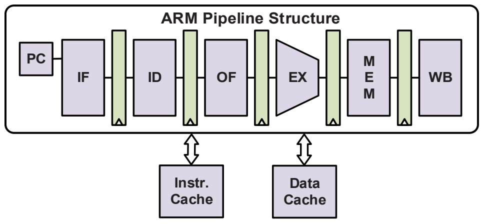
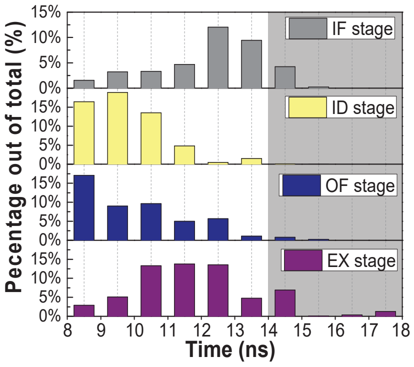


Figure 1. Pipeline architecture used in this experiment.

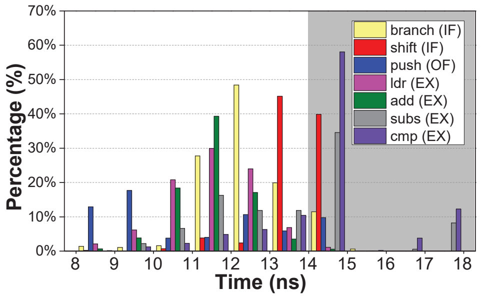
Fig. 2(a) shows the observation of the instruction timing distributions, i.e. circuit delay, at each pipeline stage in benchmark program "stringsearch" at 0.5V. Although the design has been synthesized with the same critical path delay among pipeline stages, a more than 3X of delay variation for individual instructions is observed at every pipeline stage. Similar wide spread of delay is also observed at nominal voltage of 1.1V leading to the hypothesis that different instructions exercise different paths within the pipeline and shows considerably different performance. Other observations include: (1) the instruction delay occurring at EX stage presents the longest delay in the pipeline mostly due to the complex operations in ALU; (2) only a very small number of instructions exercise the longest critical paths. For example, the long delay instructions beyond 14ns occupies only 7.3% of all valid instruction at EX stage, and only 4.5% at IF stage. Overall, 14.3% of total instructions experience delay beyond 14ns although the critical path delay is at 18ns which determines the minimum clock speed. MEM and WB stages in our design do not contribute long timing paths and thus are not included in the results in this paper.

The above observation reveals a drawback of the conventional design strategy where only the worst-case delay is considered even though only 14.3% of instructions exercise the critical paths. If the performance requirement of each instruction could be predicted and associated with the required supply voltages, significant energy saving could be achieved. In this design, we roughly choose 14ns as a critical mask. All instructions at every stage with delay beyond 14ns are considered as "long delay instructions", and further studied and classified in Section 2.2. In our design, additional 10% timing margin is applied to cover the other process-temperature-voltage variations similar to conventional design.

Fig. 2(b) shows the timing distributions of some common instructions at specific stage. It is observed that although there is significant delay variation, some specific instructions or instruction patterns are more likely to produce longer delay. For example, *cmp* and *subs* at EX stage exercise the longest paths beyond 16ns. The reason for this instruction timing variation is primarily rooted in the architecture definition of the processor and will be explained in the following section.



(a)



(b)

Figure 2. Instruction timing distributions for benchmark program *stringsearch*. (a) Distribution at each pipeline stage; (b) Specific instruction distribution at specific stage.

2.2 Classification of Instructions

We performed extensive data and circuit analysis for all long delay instructions and correlated the instruction behaviors with the gate level netlists in the synthesized design of the ARM core. The root causes of long delay instructions are summarized and classified into the following four main categories.

Category 1: Instruction from long execution in single stage

In this ARM processor, compare (*cmp*), subtraction set (*subs*), reverse subtraction set (*rsbs*) and multiply (*mul*) are the most critical long delay instructions. An example of the long delay from *cmp*, *subs*, *rsbs*, is shown in Fig. 3(a). An extra-long critical path inside EX stage is exercised by both ALU operation and computation of conditional flags. As a result, such instructions deterministically take longer time than regular ALU instructions such as *add*, *sub*, etc, even though their delay varies with the operand values during the operation.

Category 2: Instruction from long inter-stage operation

The majority of long delay instructions in this category is associated with *branch* instructions (*beq*, *bne*, *blt*, *bgt*, etc.). When *branch* instructions reach the EX stage, they evaluate the conditional flags. Depending on the evaluated result, the logic recomputes the program counter (PC) and delivers a new PC address, as shown in the critical path for *branch* in Fig. 3(b). These branch instructions introduce delayed operations of PC resulting in late arrival of data at fetch stage. We also observed that long execution of *branch* is likely to happen if the branch is not taken because the PC needs to be rerouted. However, as the *branch* outcome is difficult to predict, we pessimistically classify all branch operation as long delay instructions.

Some complex instructions defined in ARM ISA need two or more clock cycles to complete such as instructions with shift *lsl, lsr*, *push* and *pop*, or load/store instructions with special offset *ldr* r_1, *[pc]* #, etc. For these instructions, the ARM processor will split them into several micro instructions after ID stage and introduce a "stall" at IF stage to prevent the PC from incrementing. Such a stalled instruction has high probability of producing long delay at IF stage because it invokes critical paths from both ID and IF stages, as shown in Fig. 3(b).

Category 3: Instruction with data dependency

Instructions which use values produced by the previous instructions are said to have read-after-write (RAW) dependencies. The data dependency will likely cause a long delay at OF stage. Fig. 3(c) shows such an example when *add* at EX stage writes the results into register r_1, the following *mov* at OF stage requests the r_1 content immediately. When such a data dependency is observed, the pipeline forwards from EX stage to OF stage leading to longer operation at OF stage, as the critical path shown in Fig. 3(c).

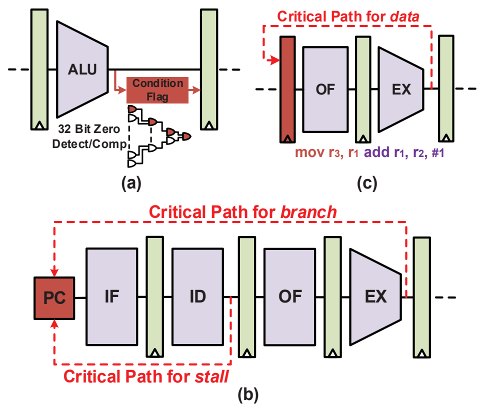


Figure 3. Some typical hardware critical paths that cause long delay instructions. (a) Critical path for EX stage, e.g. cmp; (b) Critical path for branch and stalled operation; (c) Critical path related to the data dependency.

Category 4: Instruction with instruction sequence dependency

Occasionally long delay instructions at IF and OF stage can be caused by specific instruction sequences at ID stage. This happens when decoded instructions trigger long operations such as PC stall or operand forwarding. In some cases, even though such a condition, e.g. stall or forwarding is not eventually formed, a critical path may still be exercised due to switching of sequential logic inside the pipeline. Such special critical paths are only executed from a combination of instruction sequences. This category of long delay instructions is not well defined in ISA but is highly related to the way the synthesis tools optimize the design and thus is highly design dependent. Fortunately, only small and highly predictive numbers of instruction sequences are observed in our analysis. For a specific CPU design, this category of instructions needs to be carefully scrutinized based on the processor physical design rather than the ISA.

2.3 Instruction Performance Association and Prediction Efficiency

Based on the classification of each long delay instruction category and extensive gate level simulation of long delay instructions, we identify 100% of long delay instructions with pessimism, i.e. potential long instruction is always marked as "long instruction". Table 1 lists summary of benchmark "stringsearch" with most representative instruction sets and their prediction accuracy, which is defined as the number of true long delay instructions over the total pessimistically predicted long delay instructions. It is observed that the prediction accuracy for branch instructions in Category 2 and Category 3 is low due to the delay dependency of operand values or branch conditions. Meanwhile, for the Category 1, 4, and majority of stall instructions in Category 2, the prediction accuracy could be higher than 45%. Overall, the total long delay instructions is 14.29% out of all valid instruction in program "stringsearch". The proposed instruction categories could cover all of these long delay instructions with the prediction accuracy 46.51%. In other words, we could pessimistically mark all critical long instructions with ~100% overhead.

Table 1. Critical instruction sets with their prediction accuracy in *stringsearch*.

Instruction / Pattern	Stage	Category	Percentage out of total	Prediction Accuracy
cmp/subs/ rsbs/mul	EX	1	7.19%	46.28%
Branch	IF	2	0.71%	23.15%
Stall instr.	IF	2	1.94%	52.51%
Push/Pop	IF	2	1.66%	69.57%
Data dependency	OF	3	1.03%	40.58%
Instruction Sequence	IF	4	0.93%	55.39%
Overall	All stage/category		14.29%	46.51%

3. INSTRUCTION ASSOCIATIVE POWER MANAGEMENT

3.1 Proposed Overall System Design

The overall diagram of the proposed system is illustrated in Fig. 4, which including ARM core processor (pipeline), programmable regulator, optimized compiler, and control units. In a deviation from conventional compiler operation, the compiler in our proposed scheme uses the performance association outcome in Section 2 to generate a 2-bit regulator control value encoded into each individual instruction. After the instruction arrives at ID stage, the 2-bit regulator control is decoded and sent to the voltage controller, which issues the regulator to raise supply by either 25mV or 50mV. This 2-bit voltage control encoded in the current instruction set presents a forward-looking voltage setting for the instructions two cycles after and only triggers action of the regulator one clock cycle after it is decoded.

To encode the per instruction voltage level controls, a new operating mode to the ARMv5 instruction set is added, which takes advantage of underutilized ARM condition codes by remapping them to encode the low voltage mode operations. The instruction stream hence contains all of the voltage control information without requiring additional memory footprint or drastic modification to the rest of the ISA. In rare cases when the additional condition codes are actually needed, the compiler may insert a mode switch into the instruction stream as is available in later revisions of the ARM ISA to enable/disable execution of Thumb instructions. This scheme allows us to achieve the benefits of voltage control with negligible impact on overall hardware cost. This is similar to the previous study where benign binary modification techniques have been used to encode information directly into the instruction stream [20-21].

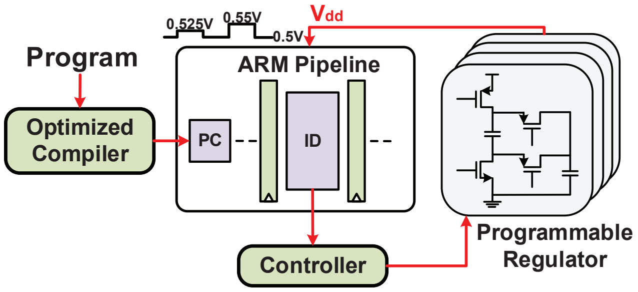


Figure 4. Overall diagram of the proposed system design.

3.2 Integrated Switched Capacitor Regulator

Fig. 5 shows the schematic of the 4-phase time interleaved switched capacitor (SWCAP) regulators used in the paper. The regulator is designed at transistor level in Cadence Virtuoso in 45nm technology and consists of a multi-phase clock generator, four 2-to-1 SWCAP cores and programmable references generators with 25mV resolution. The proposed regulator which supplies power for the ARM core runs from 1.2V supply voltage and can generate output voltages from 0.45V to 0.575V with 25mV resolution with a 200MHz clock. The regulation of output voltage was provided from the activation of switching activity of each SWCAP core based on the voltage comparator output results. The capacitors (~200pF) and switch sizes used in the regulator are optimized to support a maximum of 2.3mA current to the ARM core with nominal usage of 1.65mA current at 0.55V.

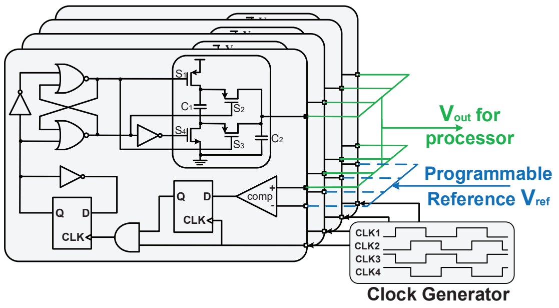


Figure 5. Implementation of the 4-phase 2-to-1 switched capacitor power regulator.

The simulated regulator output waveforms driving ARM core using transistor level schematic is shown in Fig. 6 with voltage level ramping up and down between 0.5V and 0.55V. Under the current regulator configuration and loading with ARM core, it takes approximate half clock cycle ($t_{up,25mV}$=7ns) to raise the supply by 25mV and one clock cycle ($t_{down,25mV}$=14ns) to drop the supply back by 25mV. The supply rise of 50mV requires around twice of the time for 25mV. Such a lead time requires action to be taken at least a clock earlier before long delay instruction reaches its critical pipeline stages. The energy delivery efficiency is also simulated for each output voltage level in the proposed regulator, as listed in Table 2. Although the efficiency generally improves with higher voltage due to less voltage drops across capacitors, when output at 0.55V, the switching loss happens more frequently and dominates the total power loss causing the regulator efficiency to drop.

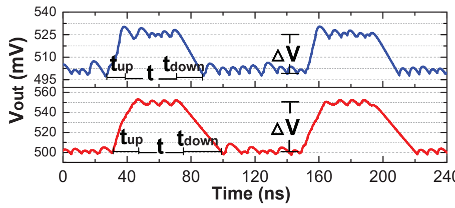


Figure 6. Simulated waveforms of output voltage transition between 0.5-0.525V (ΔV=25mV) and 0.5-0.55V (ΔV=50mV).

Table 2. Regulator efficiency for each voltage level.

V_{out}	0.5V	0.525V	0.55V
Efficiency (η)	71.81%	72.52%	69.65%

3.3 Energy Model of Proposed Associative Operation

For conventional SC regulator circuit, there are several energy delivery loss portion contributing to the total energy loss, including switch conduction loss E_{Rsw}, fly capacitors charging and discharging loss E_{Cfly}, parasitic loss from bottom capacitance $E_{C,bott}$, and the switch gate capacitance loss $E_{C,gate}$, which could be expressed by equations below for one switching activity [13]:

$$E_{loss} = E_{Rsw} + E_{Cfly} + E_{C,bott} + E_{C,gate} \tag{1}$$

these corresponding terms could be derived as following for the proposed SC regulator:

$$E_{Rsw} = P_o \frac{R_{sw}}{4R_L f_{sw}},\ E_{Cfly} = \left(1+\frac{1}{k_{int}}\right)\frac{{I_L}^2}{8f_{sw}C_{fly}f_{sw}} \tag{2}$$

$$E_{C,bott} = C_{bott}V_o^2,\ E_{C,gate} = C_{sw,gate}V_{sw}^2 \tag{3}$$

where R_{sw} and f_{sw} correspond to the switch resistance and switching frequency, and P_o and k_{int} stand for the output power and clock interleaving number.

In order to quantify the regulator output voltage transition loss, the regulator switching activities during the voltage transition time t_{up} and t_{down} should be analyzed. At different output voltage level, the regulator ripple magnitude could be addressed by:

$$\Delta V_{ripple} = \frac{(V_{dd}-2V_o)C_1 - I_L t'_{up}}{k_{int}(C_1+C_2)+C_{load}} \approx \frac{(V_{dd}-2V_o)C_1}{k_{int}(C_1+C_2)+C_{load}} \tag{4}$$

in which term $I_L t'_{up}$ stands for the charge delivered from load within short charging time after each switching. The equation also shows ripple magnitude becomes smaller at higher output voltage level. In order to achieve voltage transition ΔV, define n_{sw} is the regulator switching times during voltage rising time t_{up}, which normally could be estimated as the integer of transition voltage ΔV over the average regulator steps ΔV_{ripple}. After that, the regulator voltage transition time t_{up} and t_{down} are expressed as:

$$t_{up} = \frac{n_{sw}}{k_{int} f_{clk}} \tag{5}$$

$$t_{down} = k_{int}^2 \cdot \frac{\Delta V C_{fly} R_L}{V_o + \Delta V/2} \tag{6}$$

During each voltage transition activity, the extra energy loss in comparison with regular V_o=0.5V operation could be derived as:

$$\Delta E_{loss,trans} = [n_{sw} - (t_{up} + t_{down}) f_{sw,v_o}] E_{loss} + \left[\frac{(V_o+\Delta V/2)^2}{\eta_{v_o+\Delta v/2}} - \frac{{V_o}^2}{\eta_{v_o}} \right] \frac{(t_{up}+t_{down})}{R_L} \quad (7)$$

Equation (7) shows the extra energy loss during the voltage transition comes from both more frequently switching loss and the extra energy consuming at higher output voltage (0.5+ΔV)V, in which the second part dominant more out of the total transition loss.

Based on the ARM instruction observations before, dynamic power strategy is assigned, as listed in Table 3. As long execution instructions at EX stage (category 1) mostly take longer than 16ns, a higher voltage level of 0.55V needs to be utilized based on spice level simulation. For the other categories of long delay instructions which take 14~16ns, a voltage level of 0.525V is applied. The rest of instructions with less than 14ns will use low voltage 0.5V to save power. Thus the dynamic 0.5V low energy saving benefit could be obtained comparing with regular 0.55V operation, as (8):

$$E_{save(\%)} = 1 - \frac{V_{0.5}^2 \times p_{0.5}/\eta_{0.5} + V_{0.525}^2 \times p_{0.525}/\eta_{0.525} + V_{0.55}^2 \times p_{0.55}/\eta_{0.55}}{V_{0.55}^2/\eta_{0.55}} \quad (8)$$

in which p is the percentage of operation at each voltage level and η is power efficiency at that voltage level. In addition, when the dynamic voltage transition loss $\Delta E_{loss,trans}$ is taken into account, the actual power saving will be updated as:

$$E_{save,lossy(\%)} = E_{save(\%)} - \frac{\Delta E_{loss,trans} \times p_{trans}}{(t_{up}+t_{down}) f_{sw,v_o} E_{in,v_o}} \quad (9)$$

in which p_{trans} is the percentage of time a voltage transition of 25/50mV happens, and E_{in,v_o} is the input energy at voltage baseline V_o=0.5V within one switching activity.

Table 3. Dynamic power assignment strategy for different long delay instruction categories in *stringsearch*.

Voltage level	0.55V	0.525V	0.5V
instr. category	Category 1	Category 2/3/4	rest instr.
instr. time	16~18ns	14~15ns	<14ns
Percentage (p)	15.63%	15.09%	69.28%

3.4 Complier Optimization for Associative Power Management

In conventional design strategy, the instruction is bounded by the worst critical path delay on the chip and thus most instructions are treated equally in term of performance and energy from compiler point of view. On the contrary, this work introduces a compiler optimization scheme under which long delay instruction can be replaced by shorter ones to save energy consumption.

In ARMv5 ISA, the checking of "equal to" relationship could be implemented using either *cmp* or *teq* instructions. They are equivalent semantically, while implementation-wise are quite different. *teq* sets the zero status register if two operands are equal, which is commonly implemented using XORs, while *cmp* checks the relationship of greater, equal or less than between two values generally requires subtraction using adders. As a result, *teq* can be operated much faster than *cmp* as no subtraction is involved. Give such timing characteristics, our compiler replaces *teq* with *cmp* whenever it is possible without changing semantic of program.

4. SIMULATION RESULTS

Six benchmark programs under each Mibench category are simulated at gate level to verify the proposed system scheme for 30,000 cycles. Fig. 7 shows the long delay instruction distribution at each pipeline stage for these programs. The long delay instructions vary within 13~18% out of total for different programs, with a majority occurring at IF or EX stages. The long delay instructions vary from 5~15% at IF stage and 3~8% at EX stage. The long delay instructions at OF and ID are only less than 2%. All 100% long delay instructions are pessimistically identified in these benchmarks. The instruction category distribution are shown in Fig. 8(a), with their prediction accuracy for each category given in Fig. 8(b). As expected, the accuracy of branch (category 2) and data dependence (category 3) is mostly lower than 40% as it highly depends on the instruction content or operand. The other two instruction prediction categories could achieve better prediction accuracy between 40~60%, or even higher in some benchmark. The overall long delay instruction prediction accuracy for each benchmark is summarized in Fig. 9. The instructions actual beyond 14ns are around 12~19% out of total instructions, and the overall instruction prediction accuracy is above 40% for all benchmarks.

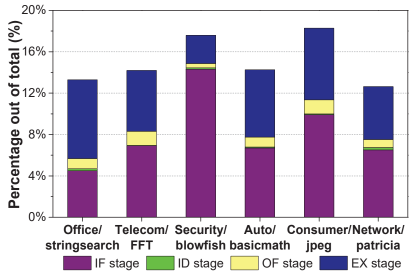
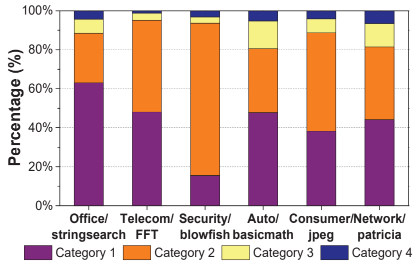
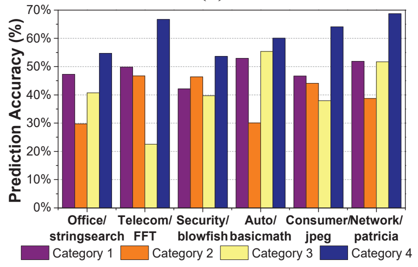


Figure 7. Long delay instruction distributions at each stage.



(a)



(b)

Figure 8. Long delay instruction category distributions (a) and their prediction accuracy (b).

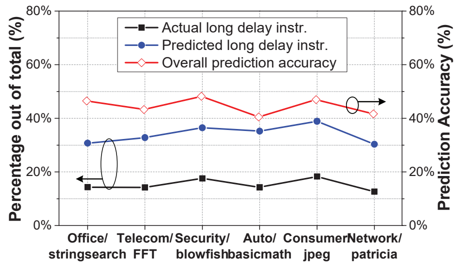


Figure 9. Overall instruction pattern prediction accuracy for each benchmark.

Based on these instruction prediction, the proposed dynamic power management scheme is implemented on each benchmarks. The power saving benefit is obtained by comparing the proposed dynamic power strategy with regular 0.55V operation, as the original power saving in Fig. 10. Another interesting observation is many long delay instructions are adjacent or compatible predicted. As example shown in Fig. 11, instructions *ldr, cmp, lsr, bne* are adjacent predicted as "long delay instructions", and *lsr* at IF stage and *ldr* at OF stage are compatible predicted within same clock cycle 5. This kind combination of instruction prediction saves voltage transition loss and reduce total predicted instruction number, which contributes around 2% additional power saving. Besides, with current optimized compiler replacing 3~9% long delay instructions (*cmp*) with shorter instruction (*teq*), another 1.5~2.8% power saving is gained. Overall, around 14% power saving is achieved from the proposed scheme. The proposed system is simulated in the Cadence Virtuoso AMS mixed-signal environment with full transistor level schematic of ARM pipeline and regulator. The prediction and dynamic adjustment is realized by a voltage controller controlling the regulator references. In order to avoid process variation and noise effects, 10% clock margin is added as the conventional clock strategy. As shown in Fig. 11, the ARM core power Vdd has been successfully adjusted based on the critical instructions in the pipeline stages. Loop instruction cases are also observed which request repeatable voltage changing.

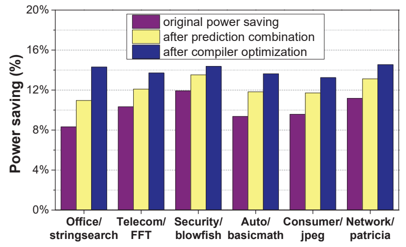


Figure 10. Overall power benefit from the proposed scheme and the improvement from the compiler optimization.

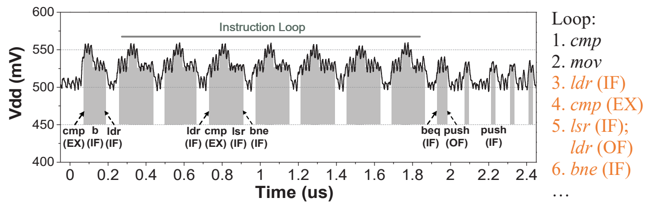


Figure 11. Spice level simulation of the proposed power scheme, with loop instructions case shown.

5. CONCLUSION

In this paper, a novel instruction governed real-time ultra-low power management scheme is proposed and explored. Based on the extensive instruction timing analysis, an association between instruction patterns and hardware performance was established, long delay instruction prediction accuracy achieves higher than 40%. Fully integrated circuit level simulation with optimized switching regulator and dedicated instruction controller was performed in a 45nm CMOS technology to verify the proposed scheme in ultra-low power operation at near-threshold regime. Implementation on benchmark programs with an ARM microprocessor design showed an extra energy saving of about 14% using the proposed dynamic power scheme. A novel compiler assisted program optimization was also introduced to further improve the efficiency of the proposed scheme by 2~3%.

6. ACKNOWLEDGMENTS

This work is partially supported by NSF grants CCF-1116610 and CCF-1533656.

7. REFERENCES

[1] International Technology Roadmap for Semiconductor (ITRS) report, http://www.itrs.net/home.html

[2] D. Sylvester, et al., "IoT Design Space Challenges: Circuits and Systems", *Symposium on VLSI Technology*, 2014.

[3] Srinivasa R. Sridhara, et al., "Microwatt embedded processor platform for medical system-on-chip applications," in *Proc. IEEE Symp. VLSI Circuit*, pp.15–16, 2010.

[4] Alice Wang and Anantha P. Chandrakasan, "A 180 mV FFT processor using subthreshold circuit technologies," in *Proc. IEEE Int. Solid-State CircuitsConf.*, pp. 292–293, 2004.

[5] Shidhartha Das, et al., "A Self-Tuning DVS Processor Using Delay-Error Detection and Correction", *IEEE Journal of Solid-State Circuits*, vol. 41, no. 4, pp. 792-804, Apr. 2006.

[6] Shidhartha Das, et al., "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance", IEEE Journal of Solid-State Circuits, vol. 44, no. 1, pp. 32-48, Jan. 2009.

[7] Hugh Mair, et al., "A 65-nm Mobile Multimedia Applications Processor with an Adaptive Power Management Scheme to Compensate for Variations", *IEEE Symposium of VLSI Circuits*, pp. 224-225, 2007.

[8] Online resource, TI white paper, "Choosing the Right Variable Frequency Buck Regulator Control Strategy", https://www.ti.com/seclit/wp/slup319/slup319.pdf

[9] Gordon Gammie, et al., "A 45nm 3.5G Baseband-and-Multimedia Application Processor using Adaptive Body-Bias and Ultra-Low-Power Techniques", *International Conference on Solid-State Circuits*, pp. 258-259, 2008.

[10] Z. Toprak-Deniz, et al., "Distributed System of Digitally Controlled Microregulators Enabling Per-Core DVFS for the POWER8 Microprocessor", *International Solid-State Circuits Conference (ISSCC)*, pp. 98-99, Feb. 2014.

[11] Rinkle Jain, et al., "A 0.45-1V Fully-Integrated Distributed Switched Capacitor DC-DC Converter with High Density MIM Capacitor in 22nm Tri-Gate CMOS," *Journal of Solid-State Circuits*, vol. 49, no. 4, pp. 917-927, 2014.

[12] Hanh-Phuc Le, John Crossley, Seth R. Sanders, Elad Alon, "A Sub-ns Response Fully Integrated Battery-Connected Switched-Capacitor Voltage Regulator Delivering 0.19W/mm^2 at 73% Efficiency", *International Solid-State Circuits Conference (ISSCC)*, pp. 372-373, Feb. 2013.

[13] Pingqiang Zhou, et al., "Optimization of On-Chip Switched-Capacitor DC-DC Converters for High-Performance Applications", *IEEE International Conference on Computer-Aided Design (ICCAD)*, 2012.

[14] Benton H. Calhoun and Kyle Craig, "Flexible On-Chip Power Delivery for Energy Efficient Heterogenous Systems", *Design Automation Conference*, 2013.

[15] Meeta S. Gupta, et al., "An Event-Guided Approach to Reducing Voltage Noise in Processors", *Design, Automation Test in Europe Conference & Exhibition*, pp. 160-165, 2009.

[16] Jing Xin, Russ Joseph, "Identifying and Predicting Timing-Critical Instructions to Boost Timing Speculation," *MICRO*, pp. 74-85, 2011.

[17] Online resource, ARM, "ARMv5 Architecture Reference Manual", https://silver.arm.com/download/download.tm?pv=1073121

[18] Online resource, http://www.gem5.org/Main_Page

[19] Online Resource: http://wwweb.eecs.umich.edu/mibench

[20] A. J. Zambreno, et al., "Flexible software protection using hardware/software codesign techniques" In *Design, Automation and Test in Europe Conference and Exhibition*, pp. 626-641, 2004.

[21] A. Meixner, et al., "Argus: Low-cost, comprehensive error detection in simple cores." *In Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, pp. 210-222. IEEE, 2007.