Basic Block Architecture for Power Saving

Dipak Krishnamani, Madhavi Krishnan, Sriram S., Ranjani Parthasarathi

dipak.k@gmail.com, madhavikrishnan@yahoo.com, sriramever@yahoo.co.in, rp@cs.annauniv.edu

***Abstract* - This paper proposes an innovative method for reducing the power consumption by handling the architectural resources at a higher granularity, namely basic blocks. This facilitates the intelligent use of deterministic clock-gating and is used to reduce power in PC address calculation, I-Cache and Functional units. The PC is incremented only once per basic block instead of incrementing for every instruction. In the I-Cache, blocks that are not going to be used in the near future are clock-gated in addition to eliminating dead blocks. The unused functional units are clock-gated to save power. In this architecture, we propose two modes of operation for fetching the basic blocks in order to decrease the latency associated with branch misprediction, thereby saving power. Simulation results show an overall reduction in power of about 21%-26% using this basic block approach.**

***Index terms* – Basic Block, Clock-gating, Power Saving, Program Counter, Instruction Cache, Functional unit, Fetch unit, SimWattch.**

I. INTRODUCTION

Power is one of the key issues in architecture designs today. It is a critical issue in embedded and hand held systems where battery life is important. It is also important in other devices like servers and desktops where cooling costs have been increasing exponentially. Hence, power should be treated as a first class design constraint on par with performance [1]

In this work, we propose a Basic Block architecture to reduce the power consumption. The principal aim of our approach is to handle the entire architecture in terms of basic blocks and use deterministic clock-gating techniques intelligently. We make modifications in the I-Cache, PC address calculations and the fetch unit to support the Basic Block architecture and evaluate the power characteristics.

The general organization of the paper is as follows, section II discusses the relevance of related work in this field, Section III elaborates the concept behind our approach and the power saving techniques in the architectural resources, Section IV provides the implementation details and the results, Section V presents the future work and Section VI concludes the paper.

II. RELATED WORK

Clock power is a major component of microprocessor power. Clock gating is a well-known technique to reduce clock power. In the work on clock-gating techniques, Hai Li et al [2] proposed deterministic clock-gating (DCG) by the key observation that for many stages of the modern pipeline, a circuit block's usage in a specific cycle in the near future is known ahead of time in contrast to pipeline balancing (PLB) which uses a predictive methodology based on ILP, to clock-gate the resources. Their experiments showed an average of 19.9% reduction in processor power with virtually no performance loss for an 8-wide issue out-of order superscalar processor by applying DCG. In contrast, PLB achieves 9.9% average power savings and 7.2% average power-delay savings, at 2.9% performance loss.

In the TRIPS architecture, K. Sankaralingam et al [3] proposed a computing system that outperforms evolutionary architectures on a wide range of applications, achieving single-chip Tera-op performance that scales with advances in semiconductor technology. The TRIPS architecture is fundamentally block oriented and uses grid type architecture. For ILP and TLP programs, blocks commit atomically and interrupt is block precise, meaning that they are handled only at block boundaries. It is noted that the architecture is designed towards higher performance rather than power efficiency.

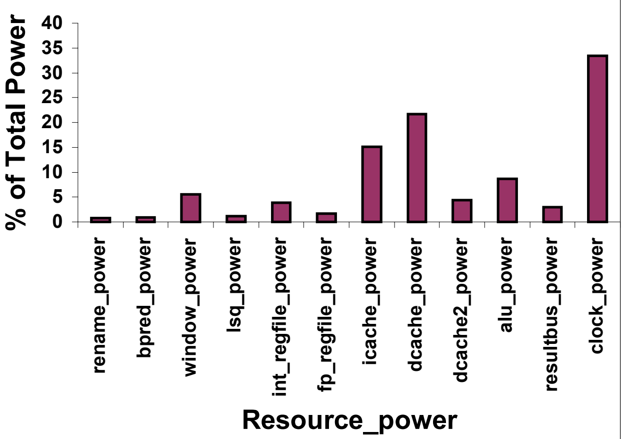
Mohan Kabdi et al proposed DBEC [4], a scheme that consists of invalidating and turning off power to cache lines that are occupied by the "dead" instructions i.e., the instructions that are not "live" at a particular point of program execution. These are the instructions that would not be used again before being replaced in the cache. The effect of this dead block elimination in cache, on both the power consumption of the I-cache and the performance of the processor was studied. The mechanism yielded an average of about 5% to 16% reduction, in the energy consumed for different sizes of I-cache without any performance degradation.

III. OUR PROPOSAL

We propose an innovative Basic Block architecture aimed at power saving rather than performance. This model is at a higher level of granularity, compared to the instruction-level handling of present architectures, The primary unit of our architecture, Basic block [5], is a stream of instructions with a single point of entry and exit with no change in flow of control in between. The information about basic blocks that is provided by the compiler is exploited for reducing power consumption. While the concept is applicable for all/many of the architectural resources, to start with, we demonstrate the use of the Basic Block Architecture in a few units. The selection of the units is based on the following

According to Pareto Analysis, 80% of the power is consumed by 20% of the resources [6]. Fig.1 below shows the output obtained by running the benchmark Compress on SimWattch 1.02. It is observed that the power consumed by the clock and cache account for a major portion of the total. Hence, the primary resources that we deal with for power conservation using basic block strategy are: PC Address calculating unit, I-Cache and the Fetch Unit.

Fig.1 Power Distribution for Compress



Power saving is done by two methods:

1. Reducing the power consumption of unused units
2. Reducing the execution time without increasing power consumption

The PC Address Calculating unit, I-Cache and functional units are modified following the first method, the fetch unit modification is based on the second method.

PC Address Calculation

In existing architectures, the PC is incremented for every instruction. In our architecture, the basic block is the primary unit and hence we propose to increment the PC only once per block. This is justified by the fact that there is no change of control flow within a basic block, and on entry, the sequence of instructions in that basic block is always executed. Individual instructions need not be kept track of, as the exception handling mechanism is also handled in a block-precise manner.

Each basic block is annotated with the number of instructions in it and using this information, the PC is updated. The PC Register and its address calculating components can be clock-gated when they are not used. Hence, we effectively reduce the power associated with the increment of PC. The amount of power saved is proportional to the number of instructions within each basic block. On an average, there are 5-6 instructions per basic block. Since the PC is updated only once per basic block and the unit is clock-gated for the remaining period, we get a 60 %–80% reduction in power necessary for PC calculation unit.

I-Cache

The authors of DBEC [4] have reduced power consumption in the I-Cache by switching off dead blocks as a whole. They handle only two extreme cases wherein the block is either alive or dead irrespective of its temporal locality. The shortcomings of this technique are exposed when there is a loop consisting of a large number of basic blocks. All the blocks are kept alive until the termination of the loop. To overcome this, we propose an enhancement in which those blocks that are not going to be used in the "near future" are kept in a standby mode by clock-gating. We handle this by associating a Block Dormant Counter (BDC) with each block in the cache. Once a certain threshold (determined by profiling) is reached, the corresponding cache lines are clock-gated, provided the neighboring block does not share the same cache line.

Functional Units

We propose to select only those functional units that are required for a basic block and clock-gate the rest that are not going to be used in the near future. Only when the functional units are necessary, i.e. during the issue stage, are they returned to active state. This method effectively saves power in the execution stage of the pipeline. The functional units are not immediately returned to the clock-gated pool once they are released. They remain in an active state for a stand by period, before being clock-gated. This will ensure that the switching on and off logic is minimized if the mix of instructions requires similar functional blocks.

Fetch Queue Modification

We suggest two modes of operation – Mode 1 and Mode 2, for the fetch unit. We keep track of the misprediction rate by having a threshold value. When the misprediction rate increases beyond the threshold value we switch to Mode 2 and switch back to Mode 1, if it drops below the threshold.

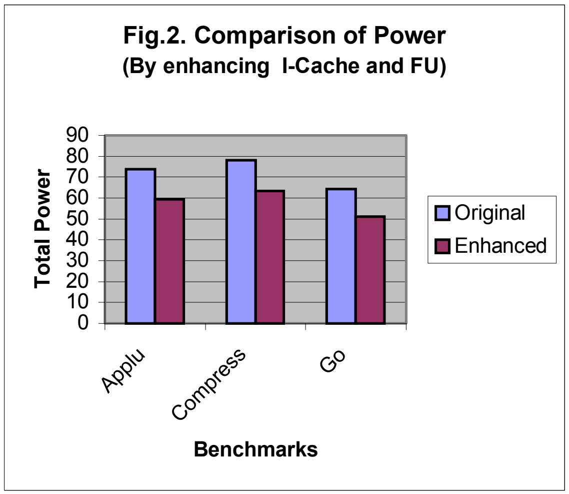
Mode 1:
This is the default mode. As long as the misprediction rate is less than the threshold value the fetch queue logic does not change.

Mode 2:
In this mode, the fetch unit is modified to be a double-ended queue. The blocks are fetched from both the taken and not taken paths of execution [7]. The blocks in the predicted path are appended to one end of the queue while the first block on the alternate path is put at the other end of the queue. In the case of a branch misprediction, the currently used end of the queue can be flushed and the fetch can proceed from the other end of the queue. Thus, this method eliminates the time wasted in fetching new blocks on branch misprediction and also the need for NPC register. Thus we intend to reduce the branch penalty. The extra cycles that would have been required for the next block to be fetched are eliminated, thus saving power.

IV. IMPLEMENTATION AND RESULTS

Sim-wattch 1.02 was used to evaluate the power characteristics of our idea. The modifications proposed in the I-Cache and Functional units were implemented in the simulator. The simulator was then run to gather statistics like individual power usage and average power usage by the different units. The effect of the enhancements to the PC and the Fetch unit were derived from the power calculations on the appropriate power distribution for those units obtained from the ouput of SimWattch. The total power consumption was obtained and compared with the power usage of the run without any improvements.

From the statistics obtained, the total power consumed after our enhancement to I-Cache and Functional Units, is plotted against that without the enhancement as shown in Fig.2. It is found that there is a reduction in power for the benchmarks considered.



From the output of SimWattch, the power saved in the PC address calculation unit, by incrementing the PC only once per basic block and the power saved in the fetch unit, by reducing the number of clock cycles associated with the branch misprediction, are estimated. The resulting improvement is included in the total power consumed and the graph is shown in Fig.3.

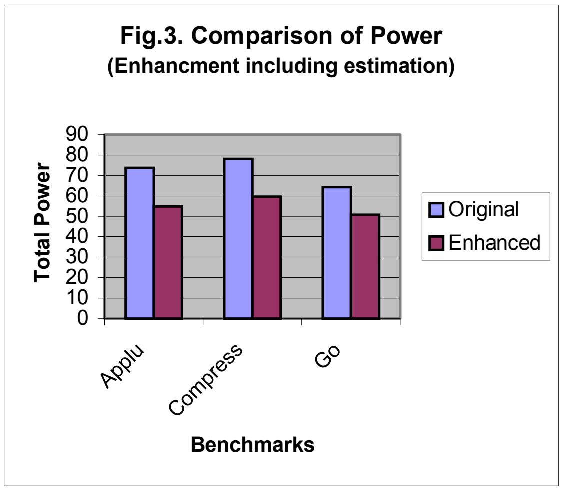
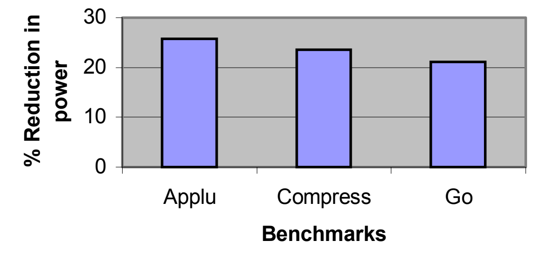


Fig.4 shows the percentage improvement in the total power consumed for three benchmarks. It can be seen that our enhancement results in a power reduction of about 21%-26% with no significant reduction in performance.

Fig.4. Power Reduction Statistics



Benchmark	Applu	Compress	Go
% Power reduction	25.74	23.62	21.14

V. FUTURE WORK

We aim to employ power reduction techniques for the entire architecture with virtually no compromise in performance. The enhancements include designing an architecture that supports higher-level granularity of basic blocks at all levels of processing, including the entire pipeline, D-Cache and Register files.

VI. CONCLUSION

In this paper, we have presented an architecture that uses basic blocks to reduce the power consumed. An analysis of the results upon simulation of the idea shows good promise. The novel idea of using basic blocks instead of individual instructions can be extended to all processing environments.

REFERENCES

[1] Trevor Mudge: "Power: A First class Design Constraint for Future Architectures", IEEE Conference, HiPC, India, 2000, pp 215-224.

[2] Hai Li, Swarup Bhunia, Yiran Chen, T. N. Vijaykumar, and Kaushik Roy: "Deterministic Clock Gating for Microprocessor Power Reduction"*, ECE Department, Purdue University*

[3] K. Sankaralingam, R. Nagarajan, H. Liu, J. Huh, C.K. Kim D. Burger, S.W. Keckler, and C.R. Moore. "Exploiting ILP, TLP, and DLP Using Polymorphism in the TRIPS Architecture", *30th Annual International Symposium on Computer Architecture (ISCA)*, pp. 422-433, June 2003. PDF http://www.cs.utexas.edu/users/cart/trips

[4] Mohan G. Kabadi, Natarajan Kannan, Palanidaran Chidambaram, Suriya Narayanan, M. Subramanian, and Ranjani Parthasarathi, School of Computer Science and Engineering, Anna University, "Dead-Block Elimination in Cache: A Mechanism to Reduce I-cache Power Consumption in High Performance Microprocessors", HiPC, India, 2002

[5] Alfred V Aho, Ravi Sethi and Je.rey D Ulman: "Compilers: Principles, Techniques and Tools," Addison-Wesley, ISBN : 817-808-046-X, Third Indian Reprint 2000.

[6] Russell, Roberta S. and Taylor III, Bernard W. Operations Management. "Pareto Analysis: Selecting the Most Important Changes to Make." November 2002. http://www.mindtools.com/pages/article/newTED_01.htm

[7] Fetch bottleneck and Branch penalty reduction using 2 instruction pre-fetch queues Guru Prasadh V.Vnkataramani, Hemanth Kumar Manoharan, Ranjani Parthasarathi presented in Poster presentation session in HiPC 2003

Dipak Krishnamani is currently doing B.E. computer science and engineering at College of Engineering, Anna University, Chennai, India. His fields of interest include computer architecture and logical reasoning.

Madhavi Krishnan is currently doing B.E. computer science and engineering at College of Engineering, Anna University, Chennai, India. Her fields of interest include computer architecture, operating systems and compiler technology.

Sriram S is currently doing B.E. computer science and engineering at College of Engineering, Anna University, Chennai, India. His fields of interest include computer architecture, operating systems and human-computer interaction.

Ranjani Parthasarathi is currently professor of computer science and engineering at the School of Computer Science and Engineering, Anna university, Chennai. She received her Ph.D. degree from the Indian Institute of Technology, Madras. Her fields of interest include computer architecture and reconfigurable computing.