An Extended Addressing Mode For Low Power

Atul Kalambur Mary Jane Irwin
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802

Abstract

This paper demonstrates the feasibility of a register-memory addressing mode in microprocessors targeted for low power applications. Using a high level power profiling tool that performs software energy evaluation, the major sources of power dissipation in a typical RISC processor are identified. It is shown that the addition of a register-memory addressing mode can target these "hot-spots" and provide power savings. Two different implementation options are considered and the power-performance trade-offs are evaluated. The reduction in performance is cushioned by the reduced instruction count, and it is anticipated that the overall impact on the total execution time of programs will be acceptable in low power application domains.

1 Introduction

Mobile computing and embedded control applications have led to the emergence of power consumption as a critical design concern for VLSI systems. Micro-processors targeted for such application environments have to be designed with energy consumption as an important parameter in the design space, in addition to speed, area and cost which have traditionally been the parameters for trade off. Circuits that enable a low power implementation of most hardware primitives in typical microprocessors have been developed. However, studies have shown that design decisions at the architectural level can have a significant impact on power consumption.

RISC machines are normally load/store architectures, where all operands must be brought in from memory and placed in the register file before being used in an arithmetic operation. Intuitively, this appears to be expensive with respect to the energy consumed in instruction and data fetch, and in the register file. Adding an addressing mode for ALU instructions to access one operand from memory, leads to a reduced Instruction Count (IC), and hence lower instruction fetch energy. It also reduces the number of register file accesses and leads to a lower register file switching activity.

Hitherto, there has been a lack of effective tools to evaluate a hypothesis of this nature. Using an in-house instruction level power profiling tool, that performs software energy evaluation on a DLX type architecture, we demonstrate that our premise is indeed valid. The register-memory addressing mode will however, result in an increase in clock cycle time, or an increase in the number of clocks per instruction. It is anticipated that this performance loss will be an acceptable trade off in low power applications.

The rest of the paper is organized as follows. Section 2 discusses the simulation engine and methodology used in our study. Section 3 examines power consumption patterns amongst the various modules of a processor. Based on the quantitative insights gained from this power profiling, section 4 presents the case for an extended addressing mode to lower power consumption. Section 5 discusses the implementation and results from our work. Section 6 concludes with further comments on future work.

2 Simulation Framework

The simulation methodology is based on high level power estimation of programs using an instruction level simulator and switched capacitance models for the hardware modules. Two different types of switched capacitance models are used : average switched capacitance per access and input data dependent switched capacitance models derived from actual circuit layouts through IRSIM-cap[10] simulations. Data dependent characterizations may be

- Bit Independent, where the operation does not depend on the values not in the same bit slice (Eg: logic operations). The total switched capacitance is just the sum of the those for the individual bits. These are bit characterized models.
- Bit Dependent, where the operation depends on operations in other bit slices. (Eg: Adders, Decoders). These are characterized by compressed tables of input vector combinations and the corresponding switched capacitance.

Each high level activity (micro-instruction) is simulated and the energy consumed in the activated hardware modules is determined from the switched capacitance models. The sub tasks performed in each pipeline stage are simulated, and the energy consumed in each pipeline stage is determined. Programs can be run on the simulator and statistics on energy consumption in different modules, instructions and instruction categories can be obtained. The simulation framework allows for experimentation with different architectures, hardware implementations, instruction sets, software techniques and capacitance models.

The simulator core is the DLXsim [5], which is instrumented to include energy profiling and changes to the instruction set architecture. A detailed discussion of the power profiler can be found in [1],[2].

3 Energy Consumption Patterns in a Microprocessor

The first step towards optimizing microprocessor architectures for power is to identify the modules in which most of the power is consumed. Amdahl's fundamental law quantifies the principle that the maximum gains can be achieved by optimizing the most common case.

A variety of programs with different characteristics were executed on the simulator. The architecture considered was a simple 5 stage pipeline DLX processor. The energy consumption in each module was determined. The switched capacitance models for the ALU, instruction decoder, register file and pipeline registers were obtained from layout netlists (2 micron technology) using IRSIM-cap [10] simulations. The instruction and data caches were modeled with an average switched capacitance of 4.5 pf per access, which is fairly typical relative to our implementation of the other modules. A 100% hit rate was assumed for the caches. Note that this provides a best case estimate of the instruction and data transfer energy, and realistic hit rates will lead to higher energy consumption for the memory modules. The instruction decoder used was a full 6-bit decoder, which represents a worst case estimate of instruction decode energy, since the actual size of the instruction set does not require the full decoder. However, the energy consumed in generating specific control signals after instruction decode were not modeled, since they depend very heavily on the low level implementation details not typically available at an architectural level power estimation stage. Sample results from this energy profiling are shown in Figure 1.

It was observed that the highest energy consumers were the instruction and data caches, pipeline registers and the register file . More energy is consumed in fetching and moving instructions and data, than in performing the actual operations on the data. Hence, the design of an architecture for low power should attempt to minimize this component of energy consumption. An implementation of this principle can be seen in Thumb, a low power microcontroller from ARM Ltd, that uses a compressor-decompresser in its instruction fetch path[6]. The advantages of a 16-bit encoding of the 32-bit DLX instruction set have also been demonstrated [3]. The high levels of switching activity in the pipeline latches causes them to be major consumers of power. This is a strong motivation for research in the design of low power latches and flip-flops, such as those used in the Strong-ARM processor [9].

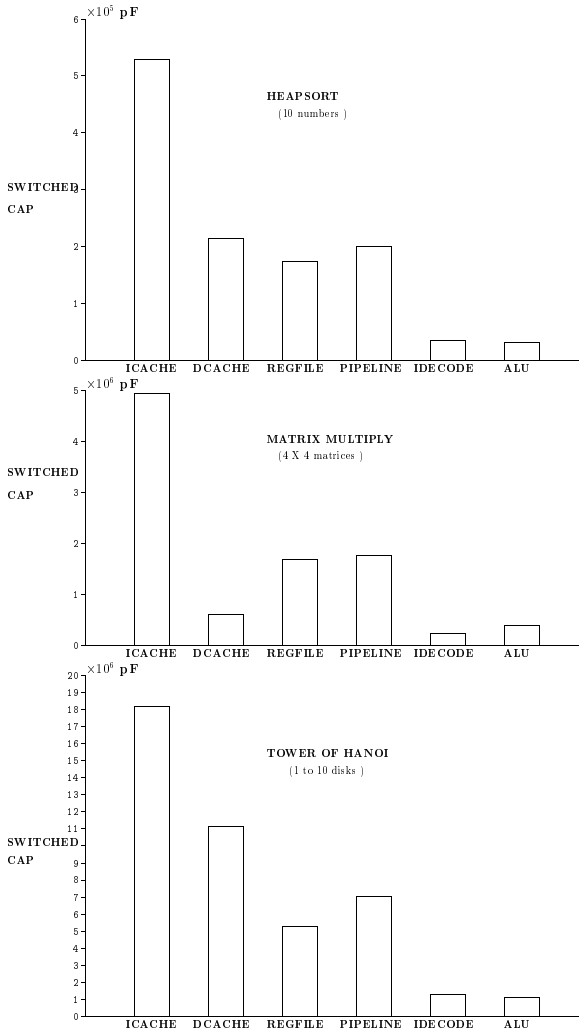

Figure 1: Energy Consumption in the Modules of a Processor

4 An Extended Addressing Mode for Low Power

As seen in the previous section, moving instructions and data between the processor core and memory is expensive with respect to power consumption.

Typically, in a load/store architecture, data is brought in from memory into a register file before being used in an ALU operation. From the perspective of power consumption, data is moved from a region where it is sparsely distributed (due to large size of memory) to a region where it is densely packed (due to the relative small size of the register file). It is operated upon in this region and the results are sent back to the memory. This leads to a very high level of switching activity in the register file and the internal busses. This architecture achieves high performance because registers are faster than memory and are easier to address.

Advances in semiconductor technology such as decreasing feature sizes and increasing die sizes, have provided designers with the capability of providing large amounts of memory on the same chip as the processor core. This significantly narrows the speed gap between registers and memory. In addition, new architectures are being proposed that tightly integrate processing power with advanced memory technologies. These architectures address the shortcomings of the CPU-centric design philosophy [7], [8]. In this context, bridging the memory gap may not be such a critical design concern in the future.

If the speed of memory and registers are comparable, and memories are more tightly coupled with processing core, it maybe feasible to examine a register memory architecture for low power applications. These architectures provide higher code density, which leads to lower instruction fetch energy. They also lead to reduced register file usage, leading to power savings in the register file.

We take a step in this direction by augmenting the DLX with a register-memory addressing mode and examining its implications on power consumption.

Consider the assembly code for the operation a = a + b, with and without the register-memory addressing mode. The comparison is shown in Figure 2.

It is generally accepted that the addition of register-memory instructions will lead to a loss in performance. We believe that this will be an acceptable trade off in low power applications. It may also make the control unit more complex, and hence more power consuming. However, the results from section 3 on major consumers of energy indicate that instruction fetch energy dominates, and reasonable decrease in this would outweigh any increase in control unit energy consumption.

Load/Store	Load/Store + Register-Memory
LW r1, d1(r30)	LW r1, d1(r30)
LW r2, d2(r30)	ADDM r1, d2(r30)
ADD r1, r1, r2	SW d3(r30), r1
SW d3(r30), r1	
4 instr memory accesses	3 instr memory accesses - 25% decrease
3 data memory accesses	3 data memory accesses - no change
9 register file accesses	6 register file accesses - 33% decrease

Figure 2: Comparison of Load/Store with Load/Store + Register-Memory

Instruction	Instruction meaning
ADDM Rd,d(Rs)	Rd <- Rd + Mem(d + Rs)
SUBM Rd,d(Rs)	Rd <- Rd - Mem(d + Rs)
ANDM Rd,d(Rs)	Rd <- Rd AND Mem(d + Rs)
ORM Rd,d(Rs)	Rd <- Rd OR Mem(d + Rs)
XORM Rd,d(Rs)	Rd <- Rd XOR Mem(d+Rs)
S__M Rd,d(Rs)	Set conditional: "__" maybe GT, GE, EQ, NE, LE, LT Store Result in R1
MUTLM Rd,d(Rs)	Rd <- Rd X Mem(d + Rs)

Table 1: New Instructions Added to the Instruction Set

5 Implementation and Results

The instruction set of the DLX processor was augmented with 12 new register-memory instructions. Table 1 shows the instructions that were added and the functions they perform. The choice was dictated by the number of available op-codes in the I-format in our version of DLXsim. A complete redesign and re-encoding of the instruction set will be considered for future studies.

Two different implementation options were considered to include register-memory instructions - a 5 stage pipeline and a 6 stage pipeline. The 5 stage pipeline, shown in Figure 3, has an adder to compute the effective address before memory is accessed. This adder maybe in the MEM stage or in the ID stage. In either case, the Clock Cycle (CC) time of the processor will have to be increased to perform this addition. The 6 stage pipe, shown in Figure 3, provides for a separate pipeline stage to compute the effective address. This will enable the same clock cycle time to be maintained as the original load-store machine, but the Clocks Per Instruction (CPI) will increase due to the additional pipeline stage.

Instruction and data memory was assumed to be on-chip caches with 100% hit rate. Average switching capacitance models were used for the caches. The register file was implemented using SRAM type storage elements. Data-dependent models were used for ALU components, instruction decoder, register file decoder and pipeline registers. Average switched capacitance models were used for register file SRAM cell array and caches.

Since the study involved modifications to the instruction set architecture, the use of a standard benchmark suite would involve writing a new compiler. Hence, the feasibility of the architecture was studied with hand-coded assembly programs, before a major task such as the redesign of the compiler is undertaken. Table 2 shows some characteristics of the benchmarks used.

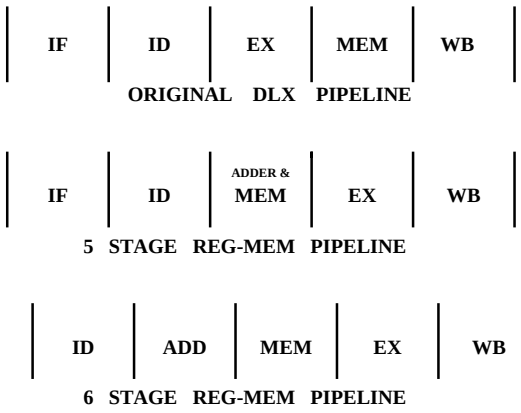

Figure 3: Pipeline Structures for Register-Memory Addressing

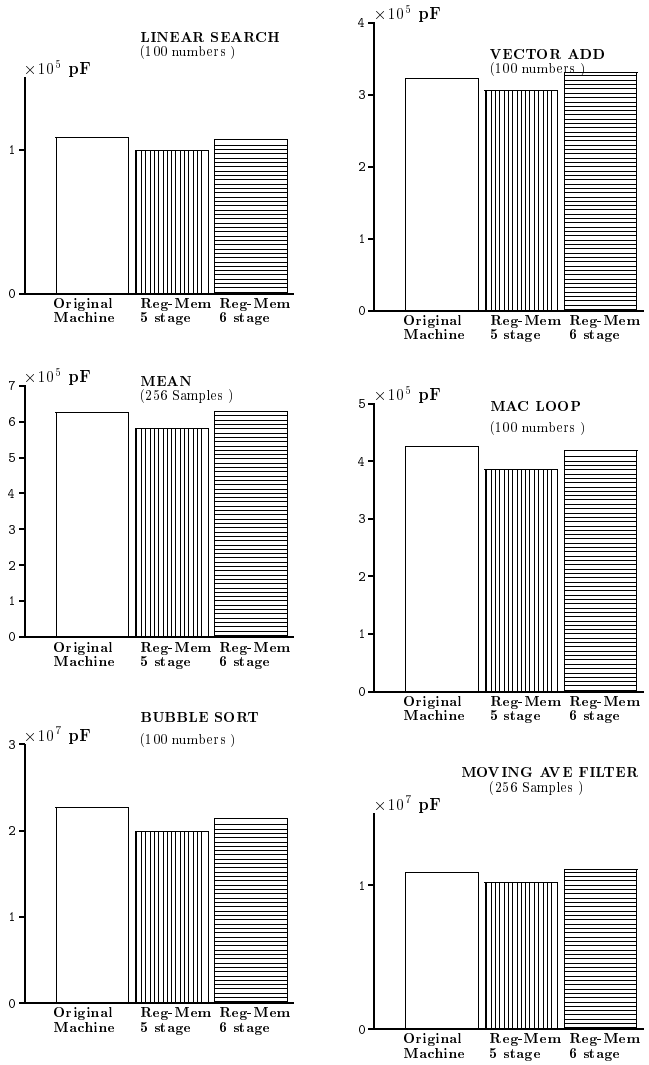

Figure 4: Switched Capacitance for Different Architectures

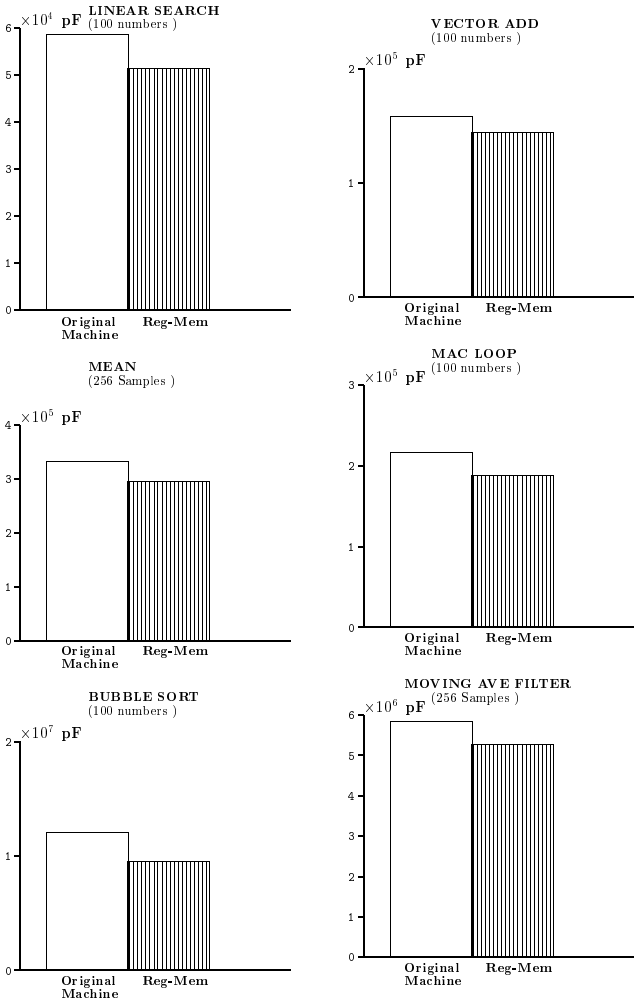

Figure 5: Reduction in Instruction Fetch Energy

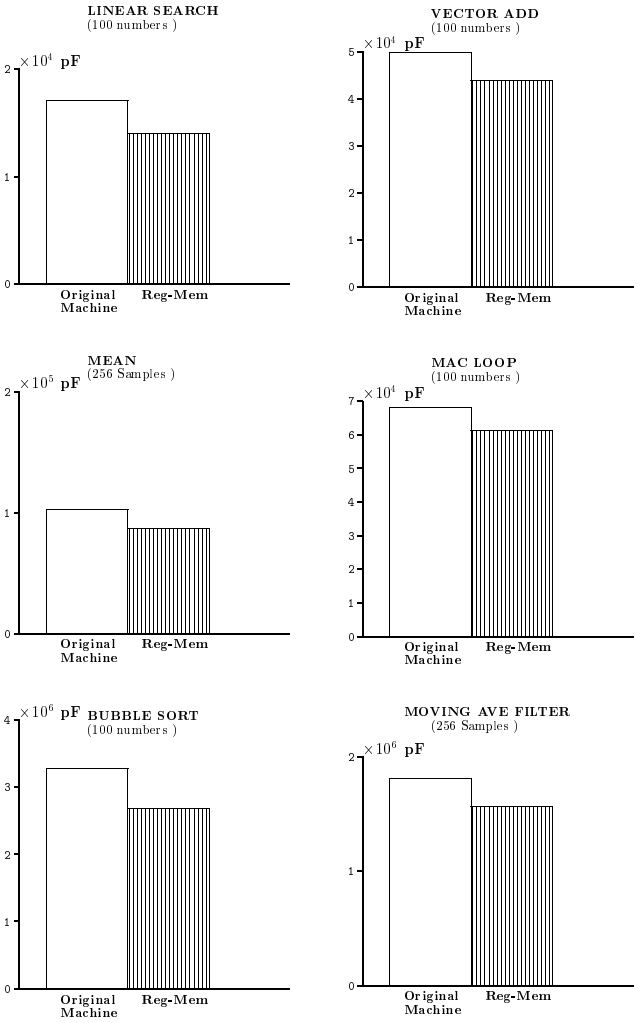

Figure 6: Reduction in Register File Energy

A 30% reduction in switching capacitance was observed when a load-load-alu sequence was replaced by a load-alu_memory sequence, as shown in Figure 2. Figure 4 shows the some the results obtained by running the hand-coded assembly programs shown in Table 2 with the different architectures.

It is observed that the 6 stage pipeline often consumes as much or more energy than the original DLX, even with the use of register-memory instructions. This is partly because the high switching activity in the additional pipeline stage offsets the decrease in instruction fetch and register file energy with the register-memory instructions. Another reason is that, in order to implement register-memory instructions, we place the ALU stage later in the pipeline, causing the source registers and immediate field to be held for additional stages. This increases the bitwidth of some pipeline latches when compared with the original DLX, resulting in a higher contribution from pipeline latches towards the total energy consumption. For the 5 stage pipeline, the overall reduction in switched capacitance ranged from 5% to 13% (Figure 4).

Figure 5 shows the change in instruction fetch energy for different programs. Reduction in instruction fetch energy ranges from 9% to 20%. Figure 6 shows the reduction in register file energy, which ranges from 10% to 18%. The energy consumption of the pipeline latches and the instruction decoder usually increase. While the reasons for increased pipeline energy have been discussed earlier, the instruction decode energy increases due to the increased size and complexity of the instruction set. Instruction decode energy increases by up to 12% (Figure 7) . However, considering the relative weights of instruction fetch energy and instruction decode energy in the overall power consumption equation, the increase in instruction decode energy will have a very marginal effect.

The major limitation is the increase in pipeline energy (Figure 8). When regular D latches were used, pipeline energy for the register-memory architecture increased on the average, by 3.3% for the 5 stage implementation, and by 44.8% for the 6 stage version. The large increase for the 6 stage pipeline is due to the extra pipeline register. When low power latches such as those used in the Strong-ARM were used, the average increase in pipeline energy was 2.6% for the 5 stage pipe and 42.6% for the 6 stage pipe.

Program	IC old	IC new	% Reduction in IC	% Register-Memory Instrs
Vector Addition	1099	1000	9.01	10
Bubble Sort	84121	66644	20.8	8
MAC Loop	1415	1315	7.1	7.6
Linear Search	409	358	12.5	14
Moving Ave Filter	40714	36618	10.1	11.2
Mean	2316	2060	11.1	12.4

Table 2: Statistics of Sample Benchmarks Used

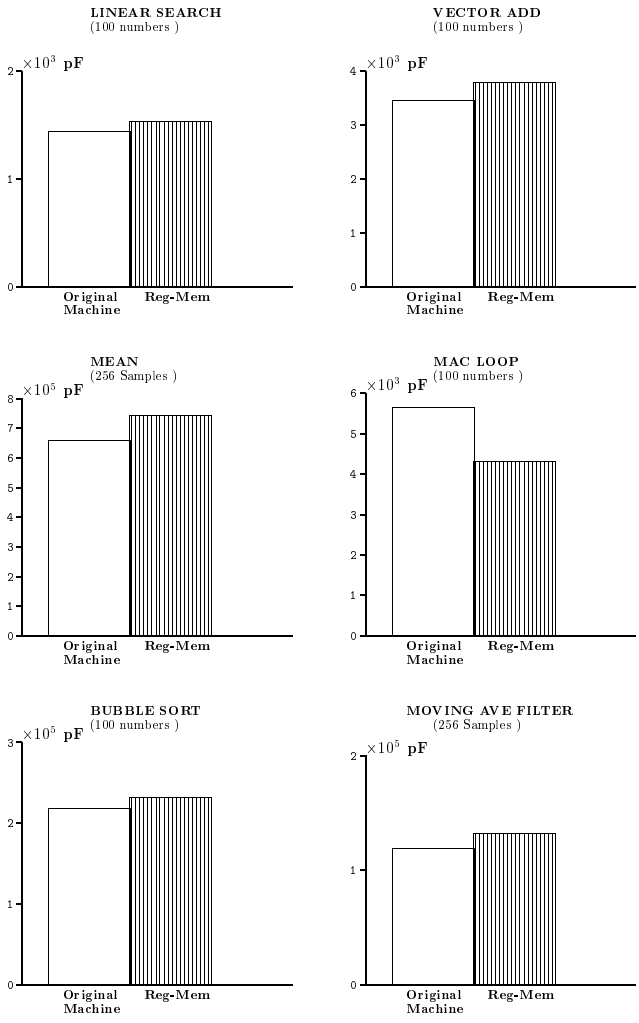

Figure 7: Change in Instruction Decode Energy

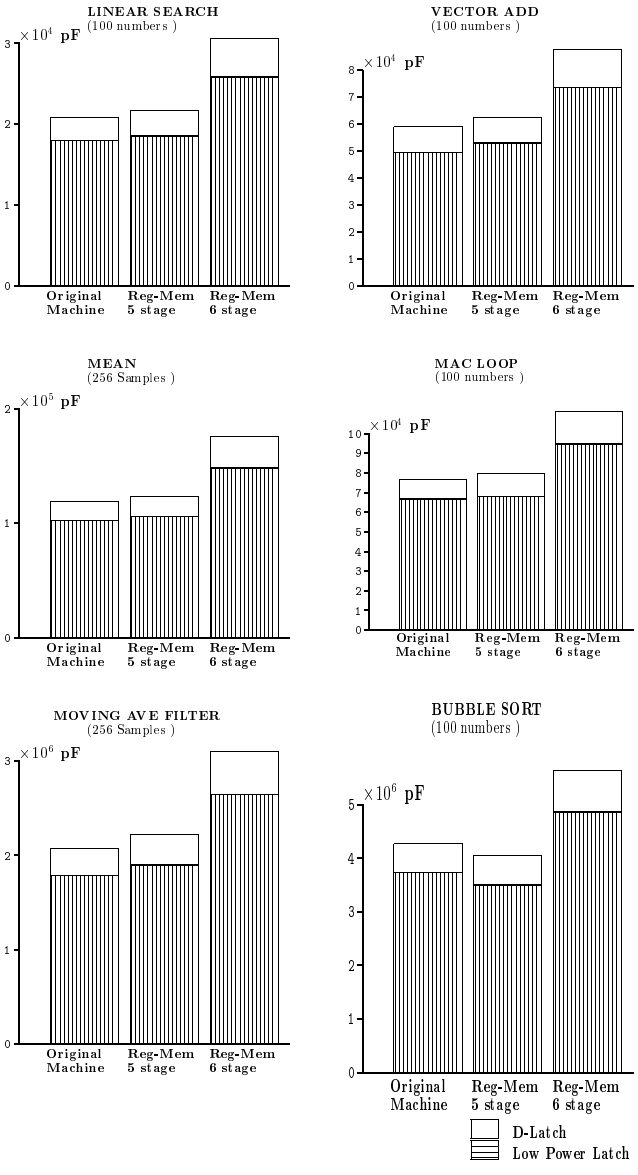

Figure 8: Change in Pipeline Energy

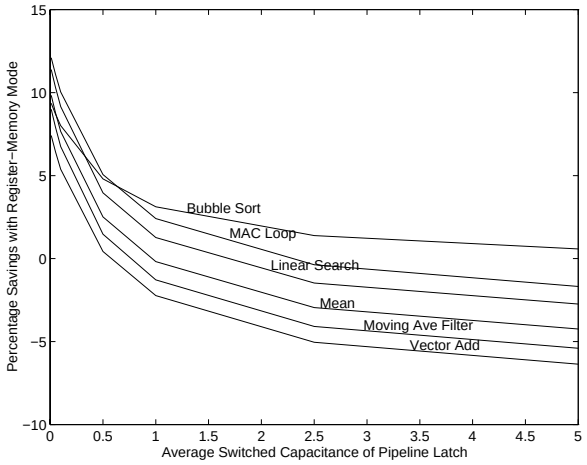

Figure 9: Effect of Pipeline Latch Capacitance on Power Savings

Figure 9 shows how the percentage savings in total energy depends on the energy consumption of the pipeline latch. It is observed that the energy saving drops very sharply with increase in pipeline latch switched capacitance.

The performance trade-off involved in using an extended addressing mode depends on the method of implementation. Consider the example of the moving average filter program. In the original load-store machine,

CPU time= IC * CPI * CC
= 40714 * CPI_{old} * CC_{old}

For the machine with register-memory instructions added, if we use a 5 stage pipeline,

CPU time= 36618 * CPI_{old} * CC_{new}

The CPI does not change, and we note that CC_{new} can be 10% slower without any performance degradation.

If we use a 6 stage pipeline,

CPU time= 36618 * CPI_{new} * CC_{old}

In this case the clock cycle time does not change and CPI_{new} can be 10% greater with no performance loss. A similar analysis can be performed for the other programs.

Since, the power savings achieved with the 6 stage pipe are much smaller, the 5 stage pipe is likely to a better choice in the power-performance trade-off.

6 Concluding Remarks and Future Work

Movement of instructions and data between memory and processor core is shown to be a major source of power consumption in a microprocessor. A small complex instruction set maybe competitive in the low power application domain, since it offers higher code density and lower levels of register file activity. We have augmented the DLX with a small number of register-memory instructions and used a simulation based methodology to show that power savings can be achieved with this extended addressing mode. There is, however, a power versus performance trade off involved that needs to be evaluated based on the performance constraints of a particular application environment. We have evaluated two different implementation options and shown that fine grain pipelining is not suitable for low power system design. It is also to be noted, that as the instruction set becomes more complex, so does the control unit, and the power it consumes would increase as a result. The power consumption in the control unit is a function of the size and complexity of the instruction set, and the optimal point beyond which control unit power dominates over instruction fetch energy needs to be investigated.

7 References

1. Huzefa Mehta, Robert Owens, and Mary Jane Irwin. "Instruction Level Power Profiling." *International Conference on Acoustics, Speech and Signal Processing*, 1996.

2. Huzefa Mehta. "System Level Power Analysis." *PhD Dissertation, Pennsylvania State University,* 1996.

3. J. Bunda, D. Fussell and W.C. Athas. " Energy-efficient instruction set architecture for CMOS microprocessors." *Proceedings of the 28th Annual Hawaii International Conference on System Sciences, 1995.*

4. John L. Hennessey and David A. Patterson. " Computer Architecture - A Quantitative Approach 2ed ." *Morgan Kaufmann Publishers Inc.*, 1996.

5. Larry B. Hostetler and Brian Mirtich. " DLXsim : A Simulator for DLX."

6. Liam Goudge and Simon Segars. "Thumb : Reducing the Cost of 32-bit RISC Performance in Portable and Consumer Applications." *Proceedings of IEEE COMPCON*, 1996.

7. Ashley Saulsbury, Fong Pong and Andreas Nowatzyk. "Missing the Memory Wall : The Case for Processor/Memory Integration." *International Symposium on Computer Architecture*, 1996.

8. David Patterson, Thomas Anderson and Katherine Yelick. "Intelligent DRAM (IRAM)." *http://iram.cs.berkeley.edu.*

9. Dan Dobberpuhl. "The Design of a High Performance Low Power Microprocessor". *International Symposium on Low Power Electronics and Design 1996*.

10. A. Salz and M. Horowitz. "IRSIM : An Incremental MOS Switch-Level Simulator." *Proceedings of the 26th Design Automation Conference 1989.*