

# Power Analysis of a 32-bit RISC Microcontroller Integrated with a 16-bit DSP

R.S.Bajwa      N.Schumann

H.Kojima

Semiconductor Research Laboratory, R&D Division, Hitachi America, Ltd.

## Abstract

While power consumption has become an important design constraint very few reports of power analysis of processors are available in the literature. The processor considered is an experimental integration of a 16-bit DSP and a 32-bit RISC microcontroller, ERD1. Simulation based power analysis on a back annotated design is used to obtain data for a set of DSP application kernels and synthetic benchmarks.

## 1 Introduction

This paper presents a summary of the results of power estimation of the ERD1 which is an *experimental* integration of a 32-bit RISC processor and a 16-bit DSP.

The capacitance used in the simulation was extracted assuming a  $0.5\mu$  technology. All power values are reported as normalized quantities and were carried out with the entire programs and data resident in on-chip memory.

The rest of the paper is organized as follows: Section 2 covers some related work. In section 3 the approach and simulation setup is described. Section 4 briefly describes the processor architecture. Section 5 covers the benchmarks used followed by the results in section 6. In section 7 observations based on the analysis are used to suggest improvements.

## 2 Related work

Instruction level power analysis has been considered by measuring the current drawn by a chip while repeatedly executing certain instructions or sets of instructions and this method was used to characterize the Intel 486DX2 and the Fujitsu SPARCite 934 [1]. This approach ascribes a base power to each instruction (or class of instructions) and a state dependent power which depends on the previous state of the processor (instruction and data). This type of analysis has high accuracy and is beneficial in creating an instruction level power model which can be used to optimize code for the

processor. At the same time such a partitioning of power is insufficient from a microarchitect's perspective. The base power is found to dominate which rules out significant gains due to instruction re-ordering and related optimizations and provides little insight into the constituents of base power.

Earlier efforts using switch level simulation (SLS) include an evaluation of the HX24E DSP [2]. Such studies while relatively less accurate provide insight into the power consumption distribution inside a processor at different levels of resolution. In order to effect changes at the architectural level greater resolution is required for some of the high power consuming modules. Continued empirical analysis and validation of the results is important because as the designs of modules and the organization of the processors become optimized for power the consumption distribution changes. The new distributions may not always be predictable hence requiring new solutions to be developed.

Power conscious design efforts have been described in [3, 4] focusing on a circuit and technology oriented approach targeting standard processor building blocks and clock power.

## 3 Approach

Power was estimated by keeping track of the switched capacitance using a switch level simulator. Energy consumed is computed as,  $E_{sw} = \frac{1}{2} C_{sw} V^2$  and summed over all nodes. The simulation was carried out using Verilog and the package that worked with it to keep track of the switched capacitance, PowerSim<sup>TM</sup> (Psim). Capacitance information was used with different levels of accuracy, from pre-layout schematic approximations to post-layout back-annotation. At times the terms energy and power are used interchangeably. If we assume that a programmer will optimize the application program to execute in as few cycles as possible and that the hardware architects have no direct control over the programmer's programming style then optimizing power accomplishes the goal of minimizing energy.

Memory power was determined by generating a simple spice model of a bit storage element and suitably loaded bit lines along with the sense sense amps. Read and write power was estimated by spice simulations of this model. This information was incorporated into a behavioral model of the memory which kept track of the power due to memory accesses. The digital parts of the memory were simulated along with the rest of the digital logic. An off-chip clock supply was assumed and no PLL was needed as a consequence.

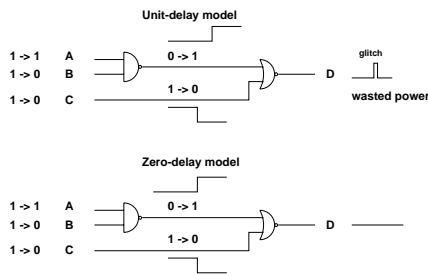


Figure 1: A simple glitching example

### 3.1 Factors affecting accuracy

Power (energy) consumed in a CMOS circuit depends on a number of issues. Some of them that cause difficulties for SLS are the rate (or slew rate) at which the input to the gate is changing (influencing short circuit current), the range of signal swings (influencing the switching power and power due to short circuit current), the presence of transients (influencing the switching power and power due to short circuit current) and finally, the propagation and inertial delays. SLS, if it does not track the history of a node beyond one state cannot take into account the effects of signal correlation when transitioning into and out of a tristate or unknown state. So why choose SLS, mainly because of its speed and because of the ready availability of an appropriate netlist description, in this case a Verilog netlist which was used for functional verification.

An aspect of power estimation that is particularly difficult to ascertain in SLS is the contribution of glitching because of its implementation specific nature [5, 6]. A zero-delay switching model is assumed to understate the amount of glitching compared to a unit-delay model. The simple example in Figure 1 illustrates this. The usefulness of higher delay models (delays greater than 1) is less certain. In the case of the work presented here library cells had their delays tuned to minimize glitching though it was not always possible to reduce it to insignificant levels. The control and clock network are free of glitching but reducing the impact of glitching in the data-path proved to be quite difficult.

Verilog based SLS assumes a net to be in any one of four different states. These are, High, Low, Tristate or HiZ and unknown. Power consumption due to transitions to and from High/Low states are weighted by unity. For transitions from defined states (High/Low) to undefined state (unknown) and vice-versa power consumption is weighted by 0.5. The rationale behind this is that in a digital circuit, outside of metastable conditions, there are no undefined states therefore, the actual circuit must have resolved the state of the net to either High or Low even if we don't know which. There is a 0.5 probability that the new state is different from the old and hence power due to such transitions is weighted by a factor of 0.5. This can again lead to over-estimation because it assumes independence of the two valid states on either side of an unknown state and ignores any correlations that might be present.

### 3.2 Power estimation and design flow

Power consumption is estimated using three increasingly accurate descriptions of the capacitance. One reason for doing this was to see if we could take information from the early stages of the design, such as schematic information, and use it to predict actual power. As we will see in the subsequent sections this, indeed is a reasonable hope. One should note though that in the future as feature size reduces it is expected that the capacitance due to interconnect will be an increasing fraction of the total. This will have to be taken into account in any scaling factor.

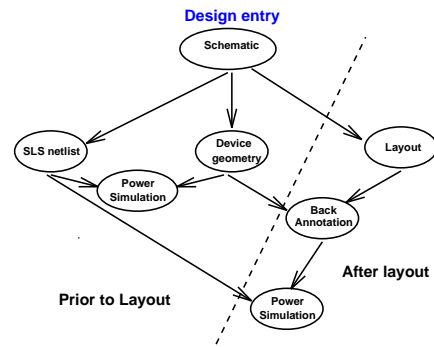


Figure 2: Over all flow of the power estimation approach

Figure 2 shows the dependencies between the different elements involved in the power estimation. Verilog modules can be generated from two sources, which are schematics and CDL descriptions (CDL is a spice like description). Capacitance information used fell into three categories described below.

- **A:** This is a gate model with cell based capacitance information obtained from schematic/cdl description. Library cell elements are characterized as black-boxes which have capacitance associated with their input and output ports. Transistor information is restricted to W and L values and there is no interconnect information available. This approach underestimates the amount of capacitance in the circuit but its advantage lies in the fact that we don't need any information from a layout. Hence simulation based on this model can be done early during the design.
- **B:** This is a gate model with cell based capacitance information obtained from schematic/cdl description and interconnect capacitance obtained from a layout. This approach provides a better estimate of the capacitance because it now adds the capacitance due to interconnect. This approach falls short of complete accuracy because it does not include the capacitance due to the transistor gates and junctions connected to nets internal to the cells.
- **C:** This is the highest resolution model. Each net in the circuit is characterized by a lumped capacitance value which is derived from the interconnect of that net and from all the gates and junctions connected to that net.

Figure 3 shows the actual steps involved in the iterative process. This is provided for the sake of completeness. By examining the results from these three setups we attempt to get information to address the following two questions.

1. Can the first method (A) be used by designers as a guide during the early stages of a design?
2. Of the second (B) and third (C) which is the preferred approach?

## 4 Processor Architecture

The ERD1 is an integration of a 32-bit RISC processor and a 16-bit DSP on a single die. It is targeted at low power mobile computing and communication devices. Figure 4 shows its main features. The DSP instruction allows overlapped operation of two loads/stores,

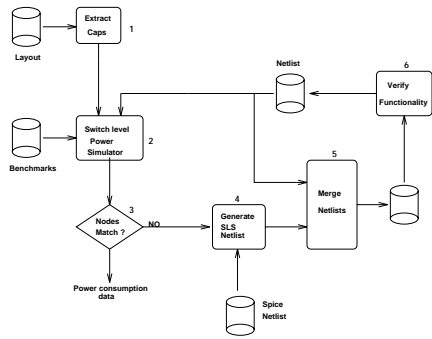


Figure 3: Procedure for power estimation

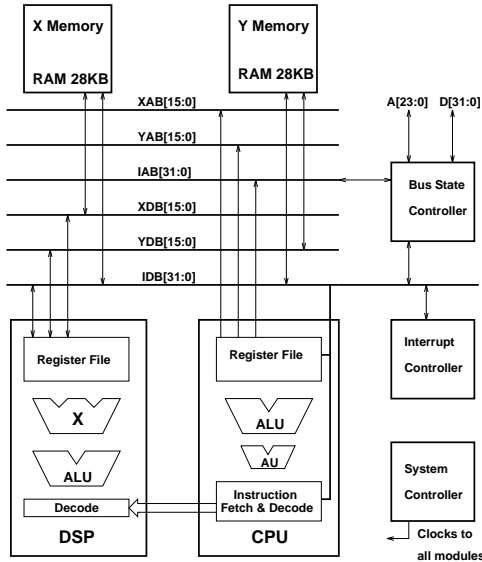


Figure 4: Block diagram of the processor core

two address increments, and independent ALU and multiplier operations. The CPU has 16 general purpose 32-bit registers. Additional control registers include those needed for hardware-looping support and modulo addressing. The DSP has eight 32-bit registers. The DSP also provides hardware support for a single cycle 16-bit by 16-bit multiply. The other main components of the DSP are an ALU and a shifter.

The minimal core configuration that we consider here is the CPU, DSP, the bus state controller, the interrupt controller and the system controller. The bus state controller monitors the IDB bus and arbitrates conflicts and also controls access between the core and offchip storage and the peripherals. The interrupt controller is an essential module for real-time systems. The system controller generates all the clocks on the chip and also performs some of the high level clock gating. On chip memory consists of two 32KB RAM memories.

The CPU has a three instruction buffer through which instructions are streamed. So loops of three or fewer instructions can execute out of this buffer. Instruction memory-accesses are 32 bit operations and a predecode is used to determine whether it is a CPU or DSP instruction, i.e. it operates with a single instruction stream. The CPU performs housekeeping functions during DSP instruction execution.

## 5 The power benchmarks

The benchmarks were chosen by trading off usefulness and computation time. A set of signal processing kernels, a speech codec performing encoding and decoding. A set of simple synthetic programs to exercise the instruction loop buffer and other macros of the architecture such as memories, busses, ALU's and the multiplier. Worst case data is used in an attempt to maximize the switching in the datapath for these benchmarks. The synthetic programs were used to validate and calibrate the simulation model against measurements done on silicon.

The list of benchmarks with brief descriptions follow:

- FIR filter: 18 tap fir filter computing an impulse response, one impulse over 24 samples (large number of 0's in input). The main computation is performed through a subroutine call.
- FIR\_BLK filter: 63 tap fir filter computing on 128 random valued samples.
- IIR filter: 2 biquad IIR filter with random data value for 40 samples.
- LMS\_A adaptive filter: Single precision 10 taps with a square wave input of 40 samples representing 2 cycles of the square wave.
- LMS\_B adaptive filter: Single precision 10 taps with 128 samples of random valued data.
- g711 speech codec: 100 samples of real audio data.
- 256 pt complex FFT:
- *pow1*. Does not use the instruction repeat buffer. The main loop operation is,  $a_0 = m_0 + a_0$ ;  $m_0 = x_0 * y_0$ ;  $x_0 = *r4 ++$ ;  $y_0 = *r6 ++$ ; This operation is a pipelined MAC operation which includes an addition, a multiplication, two loads and two address increments.
- *pow2*. Same as above, uses instruction repeat buffer.
- *pow3*. Uses instruction repeat buffer. The main loop operation is,  $a_0 = m_0 + a_0$ ;  $m_0 = x_0 * y_0$ ;  $x_0 = *r4$ ;  $y_0 = *r6$ ; This operation is a pipelined MAC operation which includes an addition, a multiplication, two loads. Or the same as the previous two except for the address increments.
- *pow4*. Does not use the instruction repeat buffer. The main loop operation is,  $a_0 = m_0 + a_0$ ;  $m_0 = x_0 * y_0$ ; This operation is a pipelined MAC operation which includes an addition, a multiplication. Or the same as the previous one except for the two load operations.
- *pow5*. Same as above but uses instruction repeat buffer.
- *pow6*. Uses instruction repeat buffer. The main loop operation is,  $a_0 = m_0 + a_0$ ;
- *pow7*. Does not use the instruction repeat buffer. The main loop operation is a *nop*
- *pow8*. Same as above but uses instruction repeat buffer.

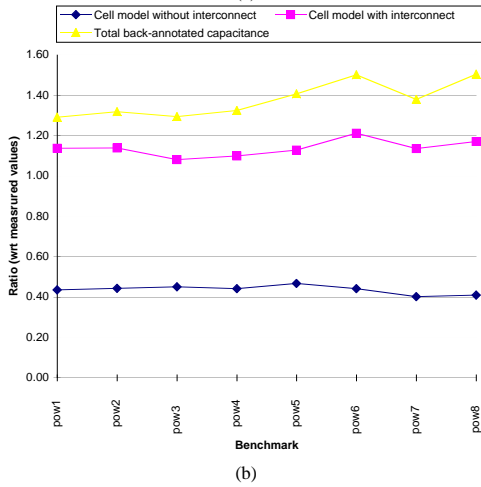
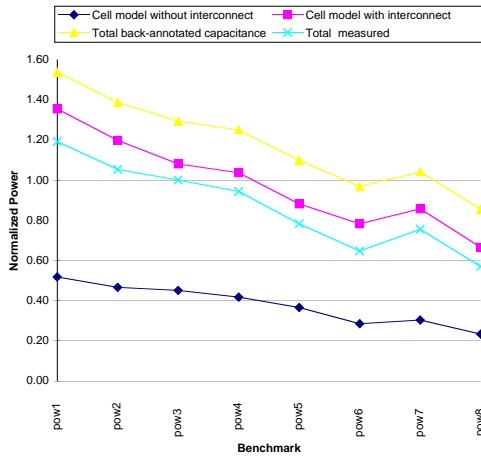


Figure 5: (a) Profile of chip power consumption estimates and the measured power, (b) ratio of measured power to the estimated quantity.

## 6 Results

Actual chip measurements were taken for the powX benchmarks and Figure 5(a) shows the profiles for the three simulated estimates along with the measured data. SLS overestimated power in two of the cases. Figure 5(b) is a plot of the ratio of the estimates to the measured quantity. It is noteworthy that these lines are almost flat. For the case of capacitance model A the scaling factor ranges between 0.4 and 0.45. For the cell-based-with-interconnect model the scaling factor ranges between 1.1 and 1.2. It is the worst in the case of the total back annotation model ranging between 1.3 and 1.5. So scaling factors can be used to estimate power to within  $\pm 10\%$  of actual.

### 6.1 Processor power profile

As seen in Figures 6 and 7 memory and the DSP are the highest power consuming modules. The DSP also shows the greatest data dependent variation mainly due to the multiplier. The memories use very low swing sense amps and use segmented bit-lines whose discharge is minimized by the sense amps. As a result their contribution while high proportionately is low by memory standards.

Top level interconnect connects the modules listed in the figures and includes data, clock and control. Instead of assigning the power

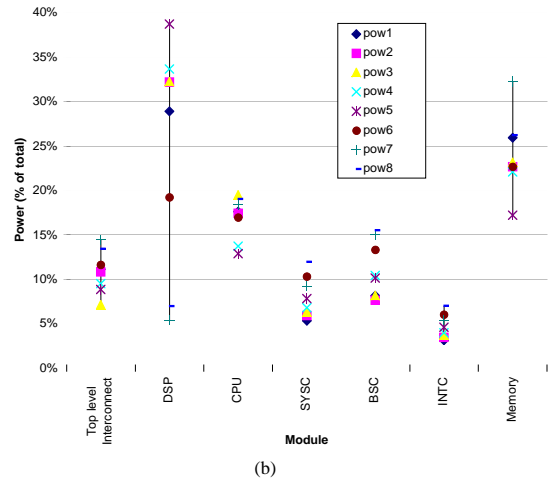
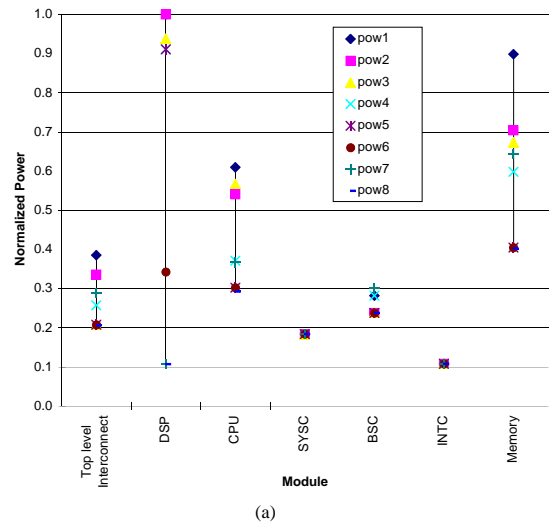


Figure 6: Modulewise breakdown of chip power consumption for the powX benchmarks, (a) normalized (b) as a percentage of the total

consumed by interconnect to the drivers (since interconnect capacitance is a passive element and hence has no power dissipation) we have chosen to show it separately to provide a better feel for the interconnect's contribution. This still omits the power consumed in activating and driving the wires from the interconnect-power and includes this power in the module where the drivers occur. In this and subsequent figures, interconnect power refers to the power due to the data lines, control, and clock lines that run between the modules shown.

The behavior of the g.711 benchmark is different from the rest because it had a higher percentage of control operations and it rarely uses the multiplier. Most of the benchmarks were typically inner loops or subtasks from actual applications.

### 6.2 Core power profile

Figure 8 shows the power distribution in the core consisting of the CPU and the DSP. Nearly all the benchmarks use the DSP extensively. The power due to the clock gating circuitry has been included in the control along with power due to decoding. Control is a large part of the total, partly due to the fact that the two different instruction sizes and also in part due to the non-orthogonality of the DSP instruction set. The majority of the control and decode

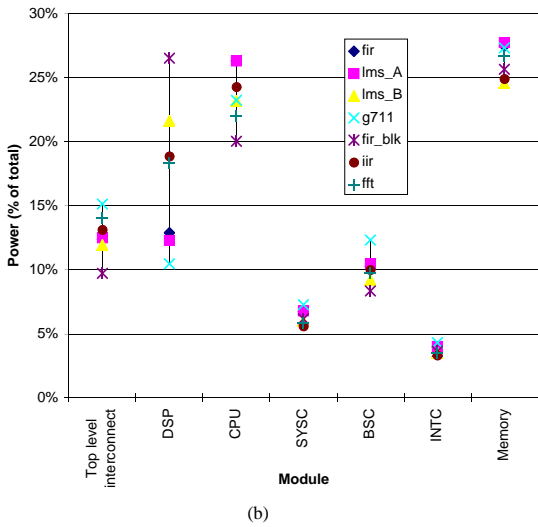
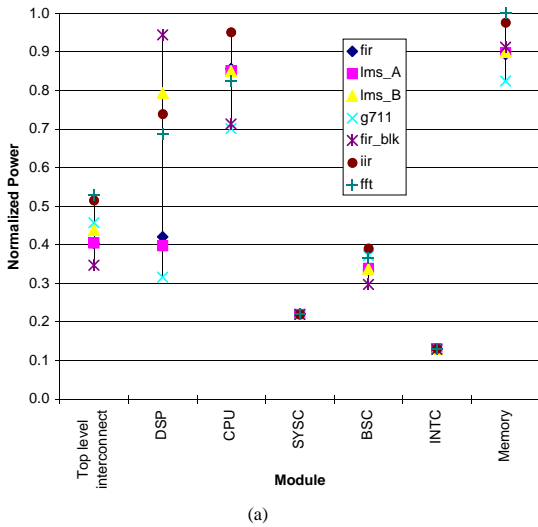


Figure 7: Modulewise breakdown of chip power consumption for the kernels, (a) normalized (b) as a percentage of the total

is in the CPU. This aspect of the power consumption can benefit from kernel buffering schemes such as [7]. During the execution of a CPU instruction the DSP which is primarily a datapath module has its clock disabled consuming very little power. The CPU, because it fetches and decodes instructions as well as generates the addresses for the X and Y memories, is always busy. The power consumed in the register file is mainly due to driving the wires into and out of the register file and is included in the CPU and DSP interconnect power shown in Figure 8. The multiplier in this design was not a Booth re-encoded multiplier but rather a symmetric design.

Figure 9 shows bus power in relation to chip power. Here the difference between the benchmarks lms\_A and lms\_B is entirely due to the difference in data.

## 7 Observations and Areas for Improvement

In this section some observations are made along with suggestions for reducing power consumption from an architectural and circuit stand point.

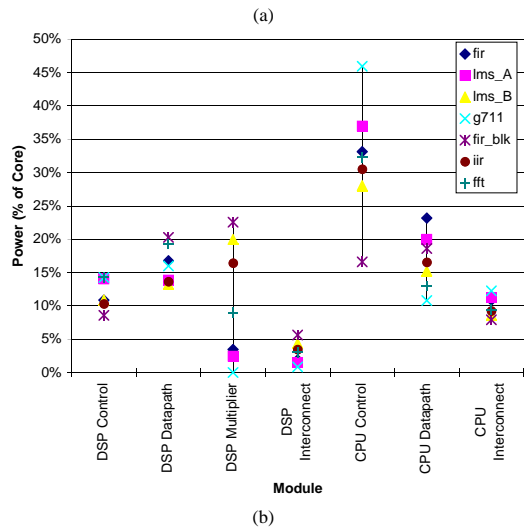
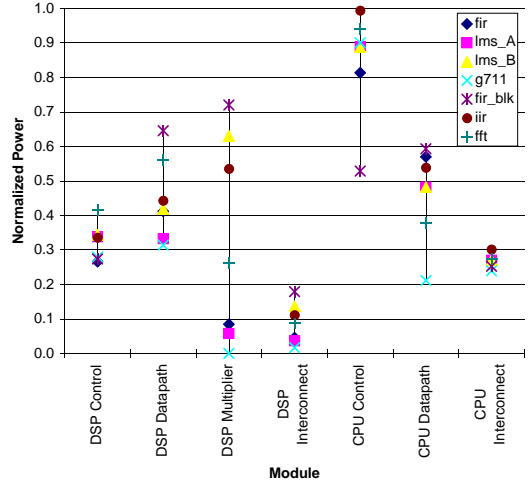


Figure 8: Distribution of core power consumption, (a) normalized and (b) as a percentage of the core.

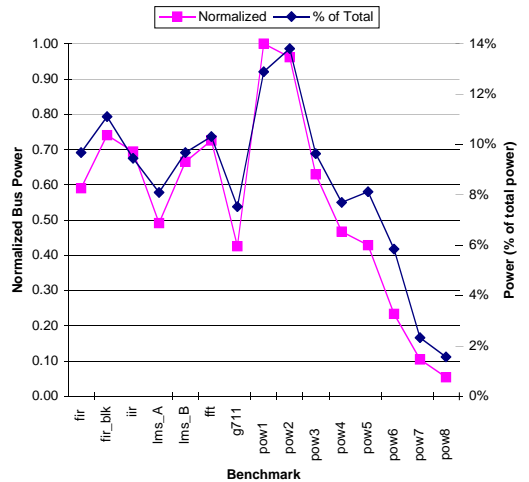


Figure 9: Profile of bus power consumption

## 7.1 Clock power

System-wide clock power varied from 30% to almost 50%. The benchmarks with higher clock power percentages tended to not be very power hungry and had lower overall power consumption. In fact it was observed that the total clock power (which includes global, local and gated) was quite stable across all our benchmarks (showing less than a 10% variation in absolute value). In the decode and control logic of the CPU and DSP anywhere from 20% to 40% of the power was due solely to the clock. Part of the reason is that as the overall power has reduced due to careful design of the datapath units and incorporation of aggressive clock gating, the share of clock power has risen.

## 7.2 Memory idle power

Even for those programs that required no memory accesses, between 8% to 12% of the power was consumed by the memory. This power is primarily clock power in the digital logic associated with the memory and used in addressing and transferring data. Introducing clock gating can reduce the memory idle power to almost zero.

## 7.3 MAC operation

Consider the single instruction loop implementation of the FIR filter operation.

$$a_0 = m_0 + a_0; m_0 = x_0 * y_0; x_0 = *r4 + +; y_0 = *r6 + +;$$

This operation implements in software a pipeline of a MAC operation i.e., the result of the multiply in the current operation is accumulated in the following cycle. Power can be reduced by folding the accumulation into the multiplier array. The idea of folding is not new but it should be strongly considered for low power implementations. With such a provision the power savings come from reduction in the data path power (primarily due to the ALU) and due to fewer register file accesses. This can lead to a power reduction in the range of 5% to 7%.

## 7.4 Address generation for the DSP

The CPU generates addresses for the X and Y memory during DSP operations. A dedicated address generator is used for the Y memory while the CPU's ALU operates as the address generator for the X memory. Due to the extra routing and control involved in the ALU it consumed between 3 to 4 times more power than the dedicated address generator. Providing a very good case for the provision of a second dedicated address generator. This can reduce overall power by just under 5%.

This has implications for the design of ALUs for embedded systems. The ALU is used for performing additions for an overwhelming majority of the time. Typically the addition operation in the ALU requires the operands to flow through a logic network (which can otherwise perform boolean operations) in order to get to the adder. This is wasteful and a better option would be to separate the adder from the ALU.

## 8 Summary

In this paper we have shown that a switch level simulation based power estimation strategy which can be automated to a large extent is viable and accurate to within  $\pm 10\%$  after a simple scaling in spite of its many limitations. This is useful during the early phases of a design as only schematic information is sufficient to estimate the

power with reasonable accuracy. The CPU's behavior in this processor reflects its use primarily for control and address generation. Standby power in the digital circuits of the memories is found to be non-trivial. Consolidating the multiplier and accumulator functions by folding the accumulation into the partial products array will avoid the use of the ALU for accumulation with no significant rise in power of the multiplier. Dedicated address units and the possibility of splitting the logic and arithmetic functions of the ALU should be considered.

Our experience reinforces the belief that the large variation in power consumption in the datapath is data dependent. Bus power is not negligible. For the kernel benchmarks it reaches upto 10% of the chip power. The bus power in this design was impacted by the heavily used shared IDB bus. In this analysis, the power consumed in controlling and switching the bus drivers is not included in the bus power though that is also influenced by the overall organization of the bus and system architecture and should also be considered.

## 9 Acknowledgement

We thank A. Kiuchi for providing us with measurement data, and K. Nitta for his help with obtaining the netlist and in helping us understand the design. We would also like to thank A. Shridhar, Y. Hatano and T. Baji for their comments and suggestions in preparing the manuscript.

## REFERENCES

- [1] V. Tiwari and S. Malik and A. Wolfe. Power Analysis of Embedded Software: A First Step Towards Software Power Minimization. *IEEE Trans. on VLSI Systems*, 2(4):437–445, Dec. 1994.
- [2] H. Kojima, D. Gorny, K. Nitta, and K. Sasaki. Power Analysis of a Programmable DSP for Architecture/Program Optimization. In *IEEE Symposium on Low Power Electronics, Digest of Tech. Papers*, pages 26–27, Oct. 1995.
- [3] Dan Dobberpuhl. The Design of a High Performance Low Power Microprocessor. In *IEEE Symposium on Low Power Electronics and Design, Digest of Tech. Papers*, pages 11–16, Aug. 1996.
- [4] G. Gerosa, S. Gary, C. Dietz, D. Pham, K. Hoover, J. Alvarez, H. Sanchez, P. Ippolito, T. Ngo, S. Litch, J. Eno, J. Golab, N. Vanderschaaf, and J. Kahle. A 2.2 W, 80 MHz Superscalar RISC Microprocessor. *IEEE Journal of Solid State Circuits*, 29(12):1440–1454, Dec. 1994.
- [5] L. Benini, M. Favalli, and B. Ricco. Analysis of Hazard Contributions to Power Dissipation in CMOS IC's. In *Proceedings of the International Workshop on Low Power Design*, pages 27–32, April 1994.
- [6] M. Favalli and L. Benini. Analysis of glitch power dissipation in CMOS IC's. In *Proceedings of the International Workshop on Low Power Design*, pages 27–32, April 1995.
- [7] M. Hiraki, R. S. Bajwa, H. Kojima, D. Gorny, K. Nitta, A. Shridhar, K. Sasaki, and K. Seki. Stage-Skip Pipeline: A Low Power Processor Architecture Using a Decoded-Instruction-Buffer. In *IEEE Intl. Symposium on Low Power Electronics and Design, Digest of Tech. Papers*, pages 353–358, Aug. 1996.