Low Power Design

Lecture #15: Wednesday, 17 May 2000
Lecturer: John Maly
Scribe: Mattan Erez

In this session we discussed three papers:

1. S. J. Montanaro et al. - *A 160MHz, 32b, 0.5W CMOS RISC Microprocessor.*
2. A. Sinha and A. Chandrakasan - *Energy Aware Software.*
3. S. Manne, A. Klauser, and D. Grunwald - *Pipeline Gating: Speculation Control for Energy Reduction.*

1 Introduction

We started by stating the two components to power:

- **dynamic power** due to switches - $\sim cv^2f$.
  It was also noted that $switch\ energy \sim (gate\ length)^3$.
- **static power** due to leakage - $\sim e^{v_{th}}$

And a couple of reasons for why power is a problem:

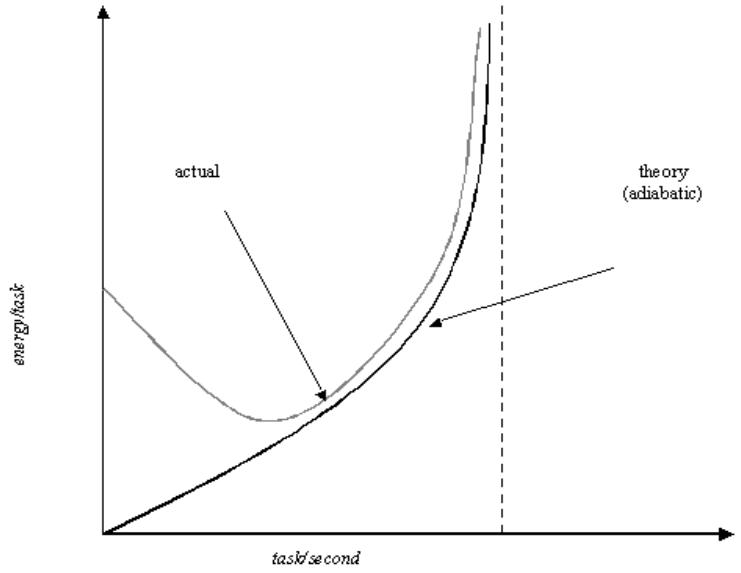
- **battery life**
- **power distribution**
- **heat dissipation** (packaging and cooling problems)

Various metrics regarding energy, power, and performance were discussed throughout and summarized here. We all agreed that the metric should involve energy and not power, since power can easily be reduced by lowering the frequency, but not the energy required to achieve a certain task. Prof. Dally described the properties of theoretical and actual ${}^{energy}/_{task}$ vs. ${}^{task}/_{second}$ curves.

Metrics for trading off power (energy) and performance:

- total energy metric
- performance objective
- try combinations that meet one objective in order to minimize the other (e.g, meet performance then minimize power).

Most of the time was spent discussing the first paper.



2 StrongArm 110

The first observation in the paper is that power and speed optimizations are driven by the same process factors, but for different reasons (as discussed above). For example, Id_{sat} should be maximized for the lowest acceptable Vdd.

The authors commented on their back of the envelope calculations starting from the Alpha 21064 and designing the low power StrongArm. We noted that it was reasonable to base the tradeoffs with such a power intensive - high performance processor, and that the same design team was responsible for both processors. The modifications required to get a 0.5W processor out of the 26W Alpha are summarized in Table II of the paper. The greatest power savings came from reducing Vdd and functionality.

Some of the functionality reduced was in removing the branch predictor. This was done by keeping the pipe short, and reducing the misprediction penalty to a single cycle by calculating the branch target during instruction fetch. This dedicated adder actually increased the power somewhat but performance improvements outweighed this increase.

Keeping the pipe short (5 cycles) also reduces the required clock power, due to the fact that fewer things need to be clocked, and that the frequency is lower.

From the floor-plan: $^1/_2$ RAM, $^1/_3$ pads, and $^1/_6$ processor (3 to 1 RAM to processor ratio), it was apparent that many optimizations should be applied to the memory caches.

The StrongArm 110 features separate 16KB instruction and data caches. The cache are banked to reduce power, with 16 banks per cache. Both caches are indexed virtually (physically tagged), this is common practice for instruction caches, but not for data. This was possibly done to support the main power saving feature of the caches. The power consumption of the cache is greatly reduced by turning on only the bank that is currently being accessed ($^1/_8$ of the cache - one bank for reading and another may be

turned on for writing). This is done by serializing the bank and row decode stages of the address, and results in increased latency. Here, it is important that the entire address be available for decoding, hence the virtually addressed data cache. The other options are to start decoding after the TLB access, which increases latency considerably, or to access all banks, and mux based on the TLB result, which increases power. A side effect of the bank structure was large associativity (32 way set associative, corresponding to 32 lines per bank).

Another important consideration in the design of the StrongArm was to provide different modes of operation whrn the processor is not actively in use. These are:

- **Idle Mode** - the processor is not clocked but retains its state, the PLL remains locked to provide for a fast ramp up. Power is decreased from 450mW to 20mW.

- **Sleep Mode** - the processor shuts down, and only the pins remain powered on to allow the system to operate properly. Power consumption in this mode is only 140μW. Powering the processor back on, requires reading the state from memory and locking the PLL, similar to a reset.

We briefly discussed whether continuous modes of operation should be, and mentioned Burd's work at Berkeley and the next paper to be discussed. In essence what is wanted is for the OS to control the amount of performance needed, and for the processor to match it with the lowest power possible (Vdd and frequency).

It was noted that for the PLL the timing has to be precise but the phase was less important. The crystal used was a commodity one found in all color televisions.

One of the largest power consumers in the Alpha 21064 was the clock. The design team took three approaches to combat this:

- **Clock Gating** - units were disabled by conditioning the clock and not by an enable. The authors commented t hat without the conditional gating the clock power would quadruple.

- **No Cap. Matching for Clocks** - the different drivers were tailored, instead of using a single driver design and matching the capacitance.

- **Reduce Frequency on a Cache Miss** - the caches are blocking, so all activity other than servicing the miss is stopped. Therefore, the clock was switched to run on the lower frequency system clock, saving power.

3 Energy Aware Software

The main idea presented in the paper is that the processor should only provide as much performance as required by the software. The major contribution of the paper is in what energy tradeoffs should be taken:

- for low energy a high duty cycle is best
- run at the maximal frequency for a certain Vdd.

Or, run as slowly as possible at the lowest Vdd while providing just enough performance.

The experiments they carried out were based on measuring power consumption by running placing the subroutine of interest in an infinite loop and monitoring the supply current until a steady value was used.

Figure 2 of the paper should show the implications of equation 2, $E_{tot} = C_{tot}{V_{dd}}^2 + V_{dd}I_{leak}\Delta t$. Which means that the total energy decreases linearly with frequency (for fixed Vdd) but scales quadratically as Vdd is increased (and the frequency is constant). However, since the StrongArm does not allow all voltage and frequency combinations, the figure is based almost entirely on extrapolated data.

4 Pipeline Gating

The authors tried to throttle the advanced mechanisms of a superscalar processor in such a way that power is reduced, while performance is hardly degraded.

specifically, they attempted to curb speculative execution to reduce *extra work* – work performed fetching and executing speculative instructions that are not retired due to branch mispredictions.

This is achieved by *gating the pipeline* – stopping fetching and decoding of new instructions, when the processor is getting too speculative. The processor is considered to be to speculative when the *gating threshold* is reached, which is the number of pending low confidence branches. The prediction confidence was estimated using a number of techniques, such as *static confidence estimation*, *JRS confidence estimation* (Jacobsen et. al.), and others. We did not have time to discuss this, but the results show that for this technique confidence estimators with large *specificity* (the fraction of mispredicted branches that are marked low confidence) work best, and that the *predictive value for a negative test (PVN)* (the fraction of all low confidence branches that were actually correctly predicted) is less important.

The metric used to evaluate the technique was not $^{energy}/_{performance}$, but rather an indirect measure of the reduction in extra work. The best results were achieved when setting the gating threshold to 3. One point to notice is that a threshold of zero is different than not-speculating. They are not allowing any low confidence branch prediction, as opposed to not allowing any prediction, respectively.

An important observation in the paper is Figure 3, the distribution of the number of cycles the pipe is gated for. This shows that most of the time the pipe is gated for fewer than 9 cycles, and supports the claim of little lost performance. These results may change dramatically as the pipeline depth increases and also the time required to resolve a branch.