Won Chun
6.911 Presentation Write-up

Slide 1: Reversible Computing

Slide 2: Introduction

Compared to some of the other stuff that we'll be covering in this class, I wouldn't consider reversible computing to be particularly weird. It is, however, quite different to what we are used to because it does change the nature of computation at a fundamental level. The approach of this presentation will be to start at the bottom, at the level of physical and computational theory (as described by Feynmann and Bennett), and work through several levels of abstraction to higher level representations. At the intermediate level, we are concerned about the construction of what will be the building blocks of reversible computers: logic gates such as billiard balls, brownian-motion driven biological systems, and split-phase charge recovery logic. At the highest level, we investigate implementations and conceptions of reversible computers such as the Pendulum reversible CPU and the radical design described in "*Optimally Scalable Computer Architecture.*"

Slide 3: Physics for Architects

There is quite a bit that can be said about the physics of reversibility, but most of these details do not interest computer architects. To motivate this discussion, I begin with a simple observation: logic is irreversible in general, but physics is. Consider a two-input, one-output boolean logic gate (NAND, e.g.). Clearly it is non-invertible because you cannot recover one of four inputs from only two distinct outputs. Physics, on the other hand, (both classical and quantum) is reversible in micro-scale; this is evidenced by the fact that the Hamiltonian time-evolution operator is invertible. But computers are physical implementations of logic, which begs the questions: how is reversibility lost, and is it worth reclaiming? It is lost through energy dissipation (usually heat). It appears, then, a reversible computer would yield potential gains in energy/power efficiency and dissipate less heat.

Slide 4: 411 on Dissipation

Irreversibility causes dissipation due to the inherent loss of information (e.g., NAND "forgets" a bit). Essentially, information and entropy are commensurate albeit complementary quantities, illustrated by Landauer's Principle. Systems that do not generate entropy also do not waste heat, and are said to be "adiabatic."

There are other sources of dissipation in traditional CMOS circuits, such as charge sharing or undriven signals discharging. Transistor switching dissipates power because the power rails yield a nonzero potential, and the transistor momentarily becomes a resistor from power to ground. SCRL solves this by resonating the power rails (this also replaces the clock). In principle, if you move the rails slowly enough (quasi statically, as it were) you can achieve arbitrarily close to zero dissipation. In practice, reversible computing allows you to make a much more attractive trade-off between clock rate and efficiency

Slide 5: gniruT Machines

We know that we want computers to be universal, which means that it ought to be able to compute an arbitrary boolean logic function (actually, all you need is two input logic functions), but we've already shown that NAND is irreversible. Are the characteristics of universality and reversibility mutually exclusive? Single input/output reversible gates (not, buffer) are not universal. Double input/output reversible gates (the only irreducible one is CNOT), are also not universal, but there are many three input/output universal/reversible gates. Already, though, it becomes apparent that a reversible computer would require more wires to deal with all of the extra outputs.

Slide 6: Going Ballistic

In saying we want reversible computers, in a way, means we want the operations of computers to more closely mimic physical reality. A simple physical system that can be used as a model of computation is billiard balls bouncing around in a so-called ballistic computer. Basically this is a thought experiment, that smells like an 8.01 problem, involving ideally hard, non-interacting spheres rolling around on an ideally flat plane with perfectly predictable trajectories, isolated from the rest of the universe. These ballistic computers can solve a computation in finite time with zero energy, but then again this perpetual motion machines is quite unattainable. In reality, errors that crop up from the non-ideal (friction, thermal noise) must be solved by adding some energy.

Slide 7: Brownian Notion

Brownian computers run forward and backward freely (driven by brownian motion), but there is a small, scalable driving force that produces a drift velocity towards the results. With zero energy, it takes an infinite time to compute (unless you get lucky enough to violate the 2nd law of thermodynamics), but at least they exist in practice. Consider RNA replication. Enzymes catalyzes both forward and backward reactions, but reactants are consumed due to the elimination of one of the products.

The Bennett paper suggests how to build a Brownian Turing Machine, which would be a great project if you can eliminate friction.

Slide 8: SCRL

Split-phase charge recovery logic is the most interesting technology yet because it is implemented by ordinary transistors on ordinary silicon. Both Pendulum and Flattop use the 3-phase variant of SCRL. The first important feature is that the power rails “swing” in a complementary fashion: the positive rail from half to full supply, the negative from half supply to ground. Transistors never switch when there is a potential across them, avoiding resistive dissipation; they switch when both rails are at half-supply. The logic value of the output “follows” one of the voltage rails as they split to their final values. The pass gate is gradually turned on to allow the next gate to sample the output, then the pass gate is turned off to isolate (in a tri-state fashion) the input and output. As the voltage rails return to half-supply, the inverse keeper blocks ensure that the inputs are always driven and brought down to half-supply as well. As the voltage rails swing slower, thereby approaching quasi static behavior, SCRL gates dissipate less power.

The pass gates allow pipelining (as well as simplifying the clock distribution), which is a key feature in high-performance computing. In a question, someone suggested that because the voltage rails ought to swing slowly that this would kill the latency of the pipeline. This isn’t exactly a crippling problem, however, because the circuit designer can chose where you place your transmission gates, just as he would for pipeline registers.

Slide 9: Pendulum (the good)

The Pendulum is a reversible computer loosely based on a simplified R2000 RISC architecture. Unfortunately, I wasn’t able to get much better information on the R2000, but it seems that the Pendulum doesn’t “cost” too much more than the R2000 in terms of transistor counts or pin requirements. Most notably, it works.

Slide 10: Pendulum (the bad)

The major problem associated with Pendulum is the lack of tools for either hardware or software developers. Some of these problems aren’t so bad, but many are absolutely heinous. Reversibility places a burden of enforcement somewhere, but it seems to be bad both in hardware and software. Most complex problems are probably better off being solved in software, but this would mean that assembly programs would be very difficult to write and debug, and compilers even more so. Garbage stack management sounds like it would be a difficult problem to solve manually or automatically, increasing the burden on the assembly programmer or compiler writer.

These problems are in addition to the obvious ones of much higher (16~32 times) transistor counts, and laying out many more wires.

Slide 11: Flattop

Flattop is a reversible computer inspired by ballistic computation, but implemented as a 2-D cellular automaton. CAs are nice because they scale quite well and are pretty easy to build. Unfortunately, Flattop is pretty much only useful as a proof of concept. Programming Pendulum might be difficult, but initializing a billiard ball computer to perform useful computations is simply ridiculous.

The power dissipation analysis was very difficult. Since development tools for adiabatic circuit design are basically non-existent, these computations were done by hand.

Slide 12: Computronium

Wouldn't it be nice to have a lump of computronium? You feed it power and a problem and after a finite time it produces an answer. Apparently this is what is offered in "Reversibility in Optimally Scalable Computer Architectures." The logic of the paper stems from the fact that it is an entropy-centric approach to computing. This makes sense given the heat generated/dissipated incommensurate scaling problem. Reversible computers seem to have a distinct advantage here because despite their larger size, they can be more effectively packed closely. This latter quality is very important because the next bottleneck after heat dissipation is the speed of light.

Slide 13: What's Wrong?

The biggest thing wrong with the paper is that there is no machine presented. There is some concern that this might be another ballistic computer in disguise seeming to work within some theoretical basis, but not when grounded in reality. For example, the paper is very concerned about getting power out, but what about power in? The analysis of the design relies heavily on asymptotic analysis, but asymptotic analysis can hide potentially very large constant factors. A particularly memorable calculation involved building a computer the mass of Saturn.

Slide 14: Conclusion

In the end, however the argument is sound. Eventually (and sooner than many people may believe), entropy will become the major limiting factor in computation. The major challenges that remain are the development of both hardware and software development tools for reversible software and SCRL. Reversible computing is an inevitability for large scale, powerful computers.