Computing without Breaking a Sweat

Groengemak
Moeiteloos groen

Abstract:

My laptop always seems to be hot and sweaty from doing things, even if I'm not asking it to do anything. This is a senseless waste of energy, and it is largel y due to design choices that stem from an era when energy was in abundent supply. Mobile devices and the end of cheap fossil fuel are two reasons why we should change, and conserve energy.
This report dives into the lowest levels of computing, and explains which energy-efficiency improvements are currently being worked on, and how we should change our style of hardware and software design. We sketch the architecture of future computers, starting at the level of the hardware and working our way up to kernel and applications.

Document descriptive information:

Title:	Computing without Breaking a Sweat
Author:	Dr.ir. Rick van Rein
`rick@groengemak.nl`	
Version:	1.0
Date:	2003/09/07
ASN.1 oid:	N/A
Release:	For Cannondale
Legal status:	Informational, GroenGemak assumes no liability

Contents

1 Introduction

My laptop always seems to be hot and sweaty from doing things, even if I'm not asking it to do anything. This is a senseless waste of energy, and it is largely due to design choices that stem from an era when energy was in abundent supply. Mobile devices and the end of cheap fossil fuel are two reasons why we should change, and conserve energy.

This report dives into the lowest levels of computing, and explains which energy-efficiency improvements are currently being worked on, and how we should change our style of hardware and software design. We sketch the architecture of future computers, starting at the level of the hardware and working our way up to kernel and applications.

Keywords: *low-energy computing, non-volatile memory, solid-state devices, bistable memory, bistable displays, interrupt-driven computing, asychronous electronics, data-flow architecture, clocktick interrupts, power budgetting.*

Recent developments in the world have brought us more awareness of the consumption of energy, and the need to evade it. Mobile devices, even though they can make us more dependent on energy, have the pleasant side-effect of bringing us technology that can do more with less energy. Another area that has long been energy-aware is the area of embedded devices. We can learn from these areas to improve our laptops, desktops and servers.
Improvements are possible at many levels, as the area of computing has originally not been very interested in saving power. It is good to see that a lot of initiatives are being taken by vendors from their niches in the market. But as consumers of devices we should be aware of the possibilities too, so we can choose more responsibly.
This paper starts with some background on the energy problems that are challenging our future as a computer-dependent society. We then turn to technical principles that can help us overcome this stress, followed by concrete techniques that implement these ideas.
It is my express wish to publish these ideas so that they cannot be turned into patents. Patents are intended to advance technology by preventing its inventors from quickly emerging competition, but the investments in the computer industry areusually so low that they need not be protected with patents.

2 Energy crisis

There are a few problems related to our use of energy or, more accurately, our dependency on fossil fuels.

Global warming. Most are aware of the problem of CO_2 building up in the atmosphere, This is a direct result of pumping up organic matter (such as crude

oil) from beneath the Earth's crust and, after playing with it a little, burn it up so it ends up in the atmosphere.

Putting an end to our dependency of fossil fuel means adding no more CO_2 to the outer skirts of our atmosphere, and thus allowing Earth to find a new balance for its temperature, one in which we hope most species can continue to live happily.

Of course, we would be wise to help the balance to be more favourable to us and the species on which we depend (from bacteria to trees and complex animals). The simplest way of doing that is to plant trees and other forms of green plants. We assume all this is well-knowne

Peak Oil. The other problem is not as well-known, but in an economic sense it is much more challanging to our way of life. It is known as Peak Oil.

The production of any resource that is dug out of the Earth takes the shape of the bell curve in Figure 1. In the beginning it is easy to find the resource, but as we advance we are less likely to hit upon new locations offering it, and the production drops off. After about half is extracted, the curve drops to never rise again. This is what geologists have known for long. For Oil, the halfway point is referred to as Peak Oil, and the average geologist estimates it around 2010, with only a few years of variation in the estimates.

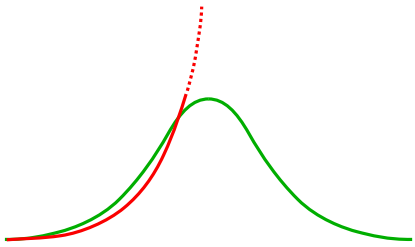

Figure 1: Peak Oil is nearing, while our energy demands rise.

The other, exponential curve in the figure depicts our usage pattern, which rises and rises, due to the growth that we have come to expect from economies. The addition of Chindia to the world market is a good example, the growth of humanity is another well-known cause. But we should also think about our own lives: If we find it normal to consume a little more power every year, we are part of this same problem.

The problem of ever-rising demands and known-to-drop offering means that the price of oil can only go up, probably at an amazing pace. This means that we are facing one of the most challenging problems of technology that we have ever seen: To retain what we like so much about technology, but at a fraction of the energy we are now consuming to get things done.

3 Technical principles

A fundamental belief at GroenGemak is that electronic devices should not burn any energy when they have nothing to do. It is surprising to see how often this idea is trodden, simply because it is not taken into account in the common design practices of industry.
A computer should take a rest in between keystrokes, and in between packets arriving on a network interface. Even though this is possible in principle, it is effectively overruled by technical choices. In our era of increasing energy problems, we should look into each of these.

3.1 Sleepy computing

Computers should sleep whenever they have a chance. This means that they would shut down their processors, perhaps stop their clock, and even swap any memory contents out to disk.
These things are best arranged by the computing device, instead of by its user. Chances are that hardly any user is interested in figuring out these details, let alone make the optimal settings.
An ideal solution would be if the processor had some advance knowledge of the expected time it can sleep. For example, if it had an idea how long until the next interrupt from the keyboard could come in, and whether work is being performed by the disk, network or screen. A deeper sleep mode can be assumed if no work is expected for some time; a lighter sleep mode could be tried first and deepened if expected work does not arrive (for example because the user has walked away from the keyboard). Of course the temporary wakeups with the express intent of deepening a sleep mode would be powerful bits of code to optimise in an operating system.
A computer would ideally not have visible modes for standby and off, but it would simply always be, like a table or door. Only when working on it would it burn energy to process input, but otherwise it would simply be off.

3.2 Interrupts driving work

A computer can rest when it has no work to do, thanks to the idea of interrupts. An interrupt is an electric signal that is raised when some exceptional situation has occurred. For example, when a byte has arrived on the serial port to which a keyboard is attached. Such an interrupt is treated as an a-synchronous signal that the processor's attention should be temporarily diverted. And if the processor is asleep, it also serves to wake it up.
Operating systems generally exploit interrupts to the fullest extent. The alternative would have been to use polling, which comes down to looking again and again when something has happened. Imagine looking at a hard disk once every millisecond to see if a block has been read in, and doing this for each block being requested; it would be an utter waste of resources.
Any form of polling is a waste of energy, as well as any other form of work that merely helps to keep a system alive and kicking. Whereas the operating

system is often aware of the need to avoid this, individual peripherals and their drivers often are not. Keyboards are continuously scanned and some mice take pictures of the desk below them, as if energy was free. *Peripherals are one area where design for reduced polling is a good idea.*

3.3 Bistable state

Computer memory can be made in a lot of different ways. In old-fashioned computers there were mechanisms like bubble memory and ring core memory, but these were abandoned because they could not achieve the density and automated production that we required for cheap computers. We ended up with static RAM (based on flip-flops) and dynamic RAM (based on capacitors). This brought us the advantage of compact, silicon-etchable memories, at the expense of using energy to retain their state.

The general idea of bistable state is that a device has two states in which it is stable. Some energy is required to change state, but nothing is needed to keep the current state. This means that state is preserved when a computer is switched off — a feat unimagineable to us now, but quite normal when storing all state in a bistable manner.

State is used in lots of places in a computer. What we see on the screen, is stored in the main memory and on hard disks, as well as in lots of intermediate registers scattered around the hardware. *A system with all storage elements replaced with bistable constructions would not notice being switched off.* Even a system with only main memory changed to something bistable could benefit, as its only response to power loss would have to be to enter suspended state. If a computer could resume quickly after power-on, it could even switch itself off between (say) two keystrokes or two network packets!

3.4 Power budgetting

Computers regulate ACPI, processor speed and so on to reduce their use of energy. This aims for the longest battery life or the lowest energy bill one can get under certain performance requirements.

A different use of these techniques of saving energy would be to consider the available power as a budget, on which the computer has to somehow perform its duties, or stop. For example, if a solar cell provides less power in a dimly lit room, less power is available and so the processor could only run slowly. Weighing the disadvantages of this slow-down against the advantage of not having to drag around batteries could actually end up either way, depending on the user. That balance is bound to change as energy prices make people more aware of their need to conserve it.

When in bright light, the solar-powered machine could perform marvels, including graphical wizardry that it would have bypassed while on a budget. When energy is abundant, animated window closes may be pleasant, but when they slow down text processing we could well do without. All these things can be a result from the idea of power budgetting.

Note that power budgetting is a good reason to reduce the frequency at which a processor runs, whereas saving the total energy used is not. Supposedly, a processor can best run at its maximum speed so it can go back to sleep as soon as possible; this maximum speed however, could well be determined by the power budget on which a system runs.

3.5 Ambient energy

A computer would ideally obtain all its energy from its environment. Like an old-fashioned typewriter that would harvest mechanical energy from the typist, leading to the hammering of a letter onto paper, a computer should do everything in its power to harvest energy from its surroundings.
It would not be beneficial to tap energy from things like GSM signals from the air, as this energy is explicitly pumped into the air by another party, and would not reduce the total amount of energy consumed by humanity. For small devices, it may however be beneficial to tap power from their interfacing links, to avoid the need for a power adaptor.
In general, useful forms of energy to tap could be movement of mobile devices and their peripherals, as well as light and heat. These sources are available in some degree of abundance, and when combined with a power budget, could make it unnecessary to ever tap the power grid.
The impact of a switch to ambient energy can be dramatic — where we can now feel constrained to purchase yet another power-slurping device, we would not need to feel any such hesitations with the purchase of a device that relies on ambient energy.
Note that other resources may be put to good use while harvesting ambient energy. Turning CO_2 into O_2 could be a nice side-effect of turning light into sugars that are led into a fuel cell. I would not mind having a laptop that is literally green if that enables photosynthesis in the device!

3.6 Grown-up logic

Logic circuitry is designed with rather simplistic devices at its heart, in support of automatic analysis, and a modular design methodology with components from libraries that can be quickly composed to form larger units. This simplicity for the designer may be another source of energy wasted in all those devices that follow their designs, and that run day and night.
The logic circuits in a computer are wasteful in themselves. Every logic port is an amplifier, which separates its input circuitry from its outputs. See Figure 2 for an example. This means that a current circuit is between between a logic output and the inputs that follow it. Compare this to old-fashioned relay-switched circuitry, which would place lots of switches in series and/or in parallel to obtain a logic function. The current would flow straight from the power supply, through the appropriate switches, to the output. In comparison with this, common CMOS circuitry is leaking energy like a sieve — every elementary logic port being a hole in that sieve. *We should aim for a logic with ports that do not work like an amplifier.*

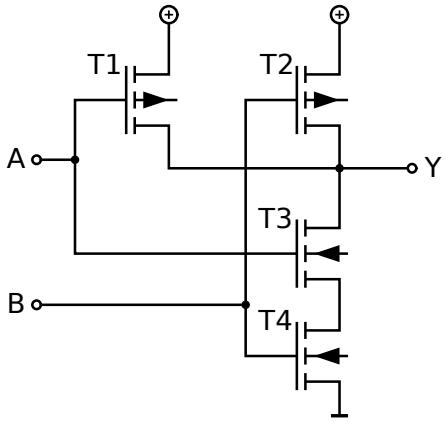

Figure 2: A CMOS NAND port amplifies signals and must be fed.

Another aspect of common logic circuitry is that it is built as a interspersed logic gates and registers. The registers store the results from the logic gates at given intervals. A system-wide clock pulse controls when the registers read the next set of values. Although this makes the timing of the whole device simpler to predict, it also wastes energy, as all registers load new contents at every clock pulse, even when nothing new is presented to them. Every time a CMOS circuit switches it consumes energy, so this turns out to be rather a wasteful simplification in logic circuitry. *We should aim to avoid clocked logic.*

4 Areas of application

Following the principles above, a lot of application areas spring to mind. We will treat them grouped by principle.

4.1 Sleepy computing

As a part of sleepy processors, we would need to handle resumes from suspended modes more elegantly. An important aspect of that would be the state into which devices must be brought when woken up; a driver could for example be asked to store the device state and switch the device off, so that the reverse could be requested upon resuming. This is sometimes very difficult, especially with BIOS-driven bits of hardware, such as VGA.
Suspend in general should become invisible, except perhaps for forgetting any sensitive data such as passwords and private keys. A display that does not need to be powered off could continue to present the same image during suspended mode, so the user would not even see suspension.
We currently need to keep a few devices powered if they are to resume the system at first touch. Any such devices ought to be self-powered in some way or another. For keyboards (or at least for power keys and lid buttons) that could mean using a piezo that creates energy when deformed and loosened, so

that it can spark off other activities. For mice, it could mean that a magnetic principle is used to induce electricity in coils, that could use the energy to transmit signals to the processor. For networking, a new protocol (ATM over I^2C perhaps) could be used to signal the computer, in such a way that the electric protocol supplies the energy used to resume the computer.
Usually, all that is required to resume a computer is enough energy to toggle a relay into the 'on' position, in which it can then happily keep itself.
Unix is sufficiently flexible in this area to adapt. The currently-missing support is always being able to get/set the state of a hardware device/driver.

4.2 Interrupts driving work

When everything must be done with interrupts, it means that nothing at all is allowed to use polling. As soon as a single thing requires polling, there is a need for a computer component to remain active even when no work remains to be done.
Linux made a great step recently, in removing the dependency on the clock interrupt, at least while sleeping. This interrupt, which took place about 100 times a second, would constantly wake up the processor from sleep modes, causing it to waste energy and take time to fall back to deeper sleep states. It is a great development that Linux now stops this timer interrupt, at least when it dozes off to sleep.
In a different place of the computer, a similar problem persists. Dynamic RAM or DRAM for short, is the most common form of memory in use in modern computers. It is built around a capacitor that forgets its contents if it is not refreshed about 1000 times a second. This means that a memory of 1 GB has an internal data displacement of about 1 TB/s for the sole purpose of performing its main task, which is to remember bits. It is not surprising then, that DRAM is quite a fuel burner. Alternatives to DRAM exist, most notably Static RAM or SRAM which needs much less power but does need more chip area, NOR-type Flash for more-or-less static pages, and NVRAM. *It is sensible to start using less wasteful types of memory than the currently omnipresent DRAM type.*
Linux does not support memories of different types being mingled, although the distinction between memory nodes comes close.
Keyboards and mice are generally peripheral devices, and to the processor they appear to cause interrupts only when something interesting is happening to them. But in reality, both types of devices contain a microcontroller that often polls the devices and simply remains silent towards the processor until something has changed. *A redesign of common peripherals seems like a useful challenge in keeping memory consumption low.*
Invisible peripherals can also burn a lot of energy. Wireless protocols such as WiFi and Bluetooth are pleasant to use, but they involve constant polling and in that they expend a lot of energy. Instead of using this to have constant knowledge of peers and link-state, it may sometimes be better to *conserve the energy used for polling wireless links and rely on error handling in higher protocol layers.* (This point is not based on in-depth knowledge of how WiFi or Bluetooth works.)

Generic peripheral interfaces, notably USB, often rely on polling as well. This is used to detect device insertions and removals. It is sometimes done in USB hardware, and sometimes left to the processor. Polling in general is a clumsy technique, and in the case of USB a change to the pull-up on the "D+" and "D-" lines should be able to do the trick, albeit in analog electronic. The idea of reducing the number of pins of a connector helps to make it simple and cheap, but certainly not to operate it cheaply — serial connectors had 9 or 25 pins, but at least some were dedicated to causing interrupts. It should be possible, at least in principle, to detect changes in USB devices' pull-up resistors without falling back to bus polling. *If USB cannot work without polling, we need a new kind of device interface.*

4.3 Bistable state

Bistable state can be exploited in several ways in a modern computer.

Electronic paper. Most notably is the display, which now requires a high-energy backlight. This backlight causes batteries to drain faster (or be heavier), and electricity bills to increase, even it is much better already than a CRT-based monitor.
The trick with bistable screens is that they keep the same picture until a change is passed to them. So, while reading a web page or your email, the display does not consume energy, but bounces back ambient light. Not only does that mean that the screen has stopped to blink at about 50 Hz, but the screen is suitable for reading in more light conditions than LCD or TFT.
Bistable displays are referred to as e-paper or e-ink, and come in a number of different technologies from different manufacturers. The screens can use colours but are usually a bit too slow for video. The quickest ones can refresh the screen twice a second, while 25 times a second would be needed for video display. The screens are probably suitable for GUI use, as these only need to update parts of the screen. While updating the display, they consume between 0.1 W and 5 W — and nothing to retain an image.

Non-volatile RAM. Memories that adopt a bistable mechanism can store a bit without a need to refresh, and are therefore non-volatile, NVRAM. Note that this is often simulated with battery-backed Static RAM, but we are here talking of true NVRAM which needs no battery-backup at all.
Several bistable technologies exist for NVRAM, and they seem to have been adopted by large manufacturers, as well as small startups that found themselves a niche on the memory market. One principle of NVRAM is NanoRAM, which is based on nanotubes that have two stable physical positions, and that effectively act as switches. Another if FRAM of FeRAM, based on ferro-electic principle of a capacitor that has ions trapped in its insulator; these ions can be forced into one of two positions, where it stably rests and influences capacitor behaviour. As a last example, phase change memory is based on the same principles as a rewriteable CD-ROM.

Static RAM or SRAM is a good intermediate solution, provided it is designed for high capacity and low power consumption. Brilliance Semiconductor, Inc. specialise in this niche and have developed a few intruiging devices that are overdue in replacing DRAM. Toshiba has also developed low-power SRAM chips. Even though SRAM is not truly non-volatile, a 512 MB memory built with it could easily be sustained for a month with a rechargeable AA battery.

SDD and Flash. Flash occurs in two forms: NOR Flash and NAND Flash. The former is directly attached to a memory bus to function as a memory unit, the latter is more suited for high-capacity, block-addressed storage comparable to a disk. Although Flash is a non-volatile memory, it outranks its use by being rewriteable only a limited number of times. Even if the number is millions for modern equipment, then still it is a severe limitation that care must be taken not to write to it too often.
Flash can take the form of Solid State Disk or SSD, replacing a mechanic disk. As an example, the pSSD construct from SanDisk could be interesting. It is based on Flash and gives a clear indication of how much can be written to it, and for how long. On Linux, the information in `/proc/diskstats` can be used to measure how this would work out in practice. When the lifetime is less than desired, just take a larger module and spread writes evenly.
In general, when using Flash it is important to use wear levelling. This means that writes are spread as evenly as possible over all the blocks in a Flash, so wear and tear does not concentrate in one spot. Alternatively, handling bad blocks gracefully could achieve the same. When depending on a wear levelling algorithm, it should involve static wear levelling, which means that even relatively static bits of data are moved from time to time.
In either case, real-life measurements of these disk alternatives are called for. SSD may have lower peaks than a disk unit, but its average power use may still be higher than a disk's.

Wear levelling in LVM2? Linux has matured a lot in the sense of storage management. With LVM2, a generic switch from logic volumes to physical ones exists that can be managed with great control — ranging from the replacement of a failed RAID drive to the resizing of filesystems on volumes. One more feature for which LVM might be used, would be as a generic place for wear levelling. This idea has not been gone into in much detail yet.

4.4 Power budgetting

Modern computers can be used in different power modes, and doing so can help to reduce the power spent. This can be helpful to limit the power used by a computer. For example, lowering the voltage from 5 V to 3.3 V usually causes currents to drop by the same factor, saving 57% of the power. The downside is that the electronics will not be able to pump electrons around with the same strength, so it takes longer for state to change, and so the frequency must be lowered.

Normal processors, those that run on a clock frequency, are called synchronous and they have been tested to perform well at certain combinations of voltage and frequency. A radically different design strategy is asynchronous design, where every pair of logic blocks communicates about its state alongside the state itself. This is called handshake: Alongside each signal it says 'the signal is available' which the next logic block picks up. After having processed it, it sends back 'done with that' and these so-called handshake signals reset.
An intruiging facility of asynchronous processors is that they need not be tuned to perform at a particular voltage/frequency combination. Instead, they adapt automatically: When power drops, all the circuits slow down to compensate for longer charging times of electronics and when cooled they will automatically speed up as conductance increases. *In an asynchronous device, power budgets would be less of a concern, as the processor would slow down automatically if other devices pulled out so much of the available power that it had no other choice.* All that would be needed is a form of stabilisation against voltage changes occurring too quickly.
Asynchronous processors are often claimed to be more efficient with power, but the involvement has mainly been academic. Cornell works an asynchronous FPGA design and the Manchester School of Computer Science has designed a promising AMULET processor, which in effect is an asynchronous ARM-compatible processor and system-on-chip. As for generic logic on circuit boards, some initiatives in the area of asynchronous FPGA chips are going on as well.
Software can also do very useful things to implement power budgetting. It could for example defer disk updates until enough energy is available, or spread them evenly. Normal disk usage patterns try to avoid switching on the device, but with electronic alternatives such as SSD it may be more useful to spread the updates over time, so as to benefit a power budget.
Screen rendering could also be subjected to power budgetting. For example, dragging a window and seeing the image move might take a lot of energy on a bistable monitor, and budgetting power could mean that only once in a while a new window position was drawn.
Desktops in general would do well to implement a power savings mode, and switch to it automatically when processor utilisation is high. This would respond nicely to the situation where a processor is slowed down due to power budgetting reasons such as solar panels yielding too little power for fullblown operation.

4.5 Ambient energy

Among the most intruiging challenges of modern-day computing should be the exploitation of ambient energy. None of us wants to return to mechanical typewriters, but they had the definite advantage that they would not break your back because of the weight of the battery... not that the typing mechanism's weight would save your back though.
Light and warmth is an obvious source of energy that would not be harmful to extract from the environment. Warmth being the more difficult one, as any method of harvesting that is based on temperature differences, and in

harvesting that the difference is usually anihilated. So we are left with light in all its colourful occurrences, including the colours invisible to us. Solar panels based on silicon, available in hobby electronics stores, harvest about 180 W/m^2. The energy it creates lowers by cosines due to angles with the light source, but an assumption could be that about 100 W/m^2 ought to be possible. On the back of a display, one has about A4 format available, or 210x297 mm^2, which could then yield about 6.2 W. In indirect light, a safer assumption would be 5 W. Could a laptop run on such power budgets? We think so, as detailed in Section 5.

Keyboards generally scan the grid of keys attached to them. If not, then at least the grid is powered all the time, so that pressing or releasing a key causes a change on an input, which is being scanned by a dedicated microcontroller. A keyboard is normally a set of switches on a 2D matrix. Instead of using keys, it could be possible to use piezo-devices, which give a positive pulse when deformed and a negative one when released. At the very least this could evade the need for scanning the matrix for a change of state to passive keys, because the keys would report in a lively manner that their state had changed. The energy from the key being pressed or released might actually suffice to power the sending of the bytes that signal the key press or release. Such a keyboard would not consume any power at all, but it would be a drop-in replacement for existing ones.

Mice are in a similar situation. They sometimes photograph the desk below them and let the processor compare successive shots to detect movement. This is rather wasteful, compared to the classic mouse design which simply uses two rollers to detect that a ball is rolling over a surface. But even this mouse is powered, and must remain so while the computer is suspended. It might prove possible to use magnets in the rollers detecting movements from the ball, and induce electricity in a coil when it rotates. It may even be possible to create enough energy to send a pulse to the processor without needing any further power on the mouse.

4.6 Grown-up logic

We already described at length how asynchronous logic works, and emphesise that the promise is high, especially because only acadamies have spent their limited resources to development of chip technology to find that they could match commercial, synchronous logic. *If the money-power of a large commercial organisation would explore asynchronous logic, we can expect a dramatic improvement in device power requirements.*

It is also a good idea to stop using complex architectures whose only cause of complexity is the history that it embeds. We have the technology to build portable systems, and need not focus solely on one kind of processor architecture. Processors like ARM clearly show how much is to be gained from an approach of computing from a simplicity rather than complexity. If operating systems refuse to follow such developments, they simply deserve be banned from our user's experience... We can enjoy Linux and Mac OS X functioning on ARM processors, but the only Windows platform on them is Windows CE.

It may be a good idea to adopt Windows CE or Symbian instead of the common desktop varieties as a choice for dedicated users of Winodws.
Another aspect that seems long overdue is to reduce the number of amplifiers in a logic circuit. In current circuits, every single port, even as modest as a NAND function, includes an amplifier. This not only causes a drain of energy from the logic circuit to ground (all the input signal is drained, it is not passed on to the output, see Figure 2) but it also makes it necessary to power each and every bit of logic in the circuit. Routing power lines has become one of the bottle necks in the design of logic.
Instead of this, circuits could aim to offer switches that can be cascaded in a number of ways, including serially (old-fashiond AND) or in parallel (old-fashioned OR). As electronic switches are not as black-or-white as relay switches, there is probably a maximum number of switches that can be combined without getting confused by the shades of gray, but CMOS amplifiers certainly accept wider ranges of input voltage than their outputs supply. One of the NVRAM technologies, NanoRAM, actually uses switches and may therefore be a very suitable technology for this style of circuitry.
This idea combines exceptionally well with asynchronous design; since the need for handshake attaches an extra 'enabled' line alongside each 'data' line, it takes two lines to pass one bit. That makes sense, as the conditions are not just '1' and '0' but asynchronous circuits also have to take 'undefined' states into account. Equivalent to 'data' and 'enabled' would be (and this is indeed advocated as an option for asynchronous design) to use a line for '1' and another for '0'. When both are inactive, the signal is 'undefined', and when the proper line is active it will signal either a '1' or a '0'. The most complex elementary logic block to design is an inverter, but it is very straightforward in this style of design: Just swap the two lines. This is especially straightforward in practical on-chip routing of such lines.

5 A concrete laptop design

In this section, we introduce a high-level design of a laptop, based on technology that is available to us at present, or technology that is so nearby that it could easily be developed. The laptop is going to be a bit expensive, but it will be so light and pleasant to use that it is actually interesting for the part of the market that prefers such forms of comfort.
Of course, similar techniques can be applied to servers and desktop machines. Laptops are just the class of machines that are most likely to sell first with these techniques, because people are used to paying more to be able to carry less weight.

Power. This abstract design uses solar cells mounted on the back of the screen to gather energy. At 100 W/m^2, a relatively low estimate, it should be possible to get about 5 W out of such surfaces. We will assume that these cells are produced with energy that comes from solar cells themselves. We will also assume that something has been arranged to enjoy the solar cells for longer

Power gained	mW	**Power spent**	mW
Solar cells	5000	Processor	1100
		Display	1000
		Memory	81
		Disk	400
Piezo keys	neutral	Keyboard	neutral
Magnetic mouse	neutral	Mouse	neutral
		Margin	2419
	5000		5000

Figure 3: Laptop Power Balance

than the lifetime of an average laptop. As an alternative, a crank handle can also supply a similar power, which could be used anywhere to charge a capacitor, not a battery.
The 5 W means we are on a tight power budget, considering that a modest standard laptop still consumes 30 W when it is merely running. It would be helpful to have a power budgetting daemon run on such a laptop.
The major places where a laptop spends its energy are its screen, processor and memory. These are all open for improvements. We summerise our findings in Figure 3.

Screen. The screen can be replaced with an e-paper screen. Models in colour have been demonstrated, as well as A4-sized models. It would appear that an e-paper screen can do its work well enough. These screens take about a second to update, but in normal use we hardly ever need to change large portions at once, so GUI use should be just possible. Actually, to view a movie it would undergo 25 updates a second, so it will proabably be just a few years before all screens will go e-paper. For non-gaming use, we could probably choose them today.
We are likely to choose a model from Nemoptic, as they seem to provide larger displays than eInk. To update their displays, only a few Joules are needed; count on 5 J for a full display, which can be spread as 5 W in 1 s or 2 W in 2.5 s, and so on.

Processor. A bold choice concerns the processor of our laptop: it will not be one that is compatible with the incongruent line of i386-processors. Their use of energy simply does not work for us. We would prefer an i.MX31 from FreeScale, for example; this is an ARM11-based System-on-Chip that runs at 1.1 W, including drivers for its peripherals. It runs around 750 MIPS or, perhaps more insightful, it is the processor that is also used in Apple's iPhone.
The ARM1136JF-S processor embedded on the device is able to run Java bytecode as an alternative to the ARM bytecodes. This may be a practical way of getting more functionality to run on the machine than anything else.

Making the choice for an ARM-processor does not mean that it is made impossible to run Windows; Windows CE is available for ARM-processors and it is a more resource-aware option than Windows Vista.

Memory. Ideally, we would want to replace both memory and disk with one of the emerging technologies for NVRAM. In practice however, these are too expensive to replace the 512 MB DRAM that we could propose for a modest laptop, let alone overtake the hard disk as well.
A solution that is practical at present is the use of Static RAM, with a tiny power budget to keep its contents in suspended mode. If the power is known to go down soon (when running on a small energy reserve stored in a capacitor) it is necessary to swap all the NVRAM-stored data to a swap area on disk.
A good example is the BH62UV1691 memory chip from Brilliance Semiconductor, Inc. This device presents 2 MB SRAM and consumes only 9 mW when active, 18 μW in standby, and during suspend it can go as low as 1.8 μW. Its area is 6x8 mm^2, so a DIMM-sized module could hold 256 MB. Not as compact as DRAM, but not disastrous at all, and certainly workable for a laptop. A memory of 512 MB would consume 81 mW if 4 chips are constantly being interrogated for a 32 bit bus, and 0.46 mW in suspended mode.
Toshiba makes another chip, the TC51WHM616AXBN65 and -70 which provide 8 MB of what they call pseudo-static RAM. It consumes 260 μW in standby and can go as low as 13 μW in deep sleep. Its area is 8x11 mm^2, so a DIMM-sized module could hold 512 MB.
Other current-day options exist, such as using Flash instead of DRAM to save on the refreshing it needs. Although this could be interesting to store portions of code that are otherwise often loaded into main memory, it will be hard to find a Flash memory that is as energy-efficient as the named Static RAM modules. Perhaps Spansion's EcoRAM (which is Flash on a DIMM) could be sufficiently efficient, but lacking technical information their product may just as well be a marketing strategy.

Disk. A first remark for disk is that it should not use SATA, as this interface consumes about 1 W more than its parallel IDE/ATA counterpart – enough to run an extra processor! An SSD is a drop-in replacement for disk, and a solution for the IDE interface is then preferred over an SATA or USB pluggable solution.
Connecting Flash memory directly to a system can be done in a number of ways. CompactFlash cards are electrically compatible to IDE, so they can be connected to an IDE controller. The i.MX31 can also interface directly with NAND Flash devices, which are block-addressable Flash devices. It is not possible to run code from such devices, but a flash filesystem such as jffs2 could well run on them.
Finally, there is the flash-on-DIMM solution that Spansion sells under the name EcoRAM. This is NOR-flash, which means that it is directly addressable as DRAM would be on a DIMM. It comes in such large sizes that it can easily replace a modern hard disk.

An important warning is due when replacing disks with Flash-based solutions. Flash in itself is not extremely low-power, and especially SDD devices are known to have lower power peaks than efficient disks, but they do draw their power more continuously, leading to a bigger overall use. Still, this could prove useful when running under a power budget. Our current design runs under solar power and is thus better off with a constant-average current flow than with a irregularly-peaking pattern.
We will choose 16 GB pSSD instead of a disk, needing 1 mW in standby up to 0.8 W when it is reading and writing at the same time. Note that it is possible to budget power, as reading and writing only each consumes 0.4 W.

Specialised controllers A few specialised controllers are needed to make the most of this laptop. First, the power budgetting can benefit from specialised circuitry that continuously monitors the power obtained from the solar cells. When that drops, the processor should be slowed down, and preferrably also the speed at which the screen is updated. Additional capacitors are needed to store enought energy to last the time needed to slow down. When these are charged again, the processor might be sped up.
In addtion to this, a UPS-alike circuit should take care of power loss, which means loosing all energy or so much of it that the laptop cannot continue to run. When this happens, the contents of SRAM must be saved immediately to the pSSD device, using the operating system's mechanism for suspend to disk. When the power comes back, the processor can be resumed to continue as before.
To make it possible to resume quickly, it is good if the BIOS is prepared to load processor state and stack pointers and such from disk, and take off from the point where the computer was suspended. The operating system must then take care of resuming all the devices through their drivers. When done well, the e-paper screen need not even blink to show that the system went down. All the user would notice is that the computer would stop to process key strokes and network packets while it was power.

User experience. This device could cause a rather different user experience comapred to a classical, power-hungry laptop.

- The laptop would be suitable for off-grid use because it relied on ambient energy;
- The laptop would not rely on heavy batteries and chargers, but on light-weight capacitors, solar panels and perhaps a crank handle;
- The laptop would always be on, albeit at a speed that matched the energy harvested from the environment;

We feel that such devices could create a market of their own, probably replacing common laptops and reader devices in due course.

6 Conclusions

As explained in this paper, there are many ways in which our computer industry can improve the efficiency of the devices it produces. We will need to do this to overcome the danger ahead of a total energy crisis. In addition, it will mean that we can continue to enjoy devices in our immediate surroundings, to arrange those things for us that we prefer to have interact with us, rather than passively wait on paper.

Unix would seem to be ready for most changes. A few things it has not been prepared for are (1) supporting saving and loading of device state in general, (2) supporting a mixture of alternatives besides plain DRAM, and (3) support for power budgetting, perhaps in a daemon.

Links

`http://groengemak.nl/nl/article/2008-05-19-10-05.toekomst-computer.html` — Related article on our website.

`http://www.digit-life.com/articles2/storage/hddpower.html` — Power use by disk devices

`http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX31&nodeId=0162468rH311432973ZrDR` — i.MX31 information

`http://spansion.com/about/press_kit_cd.html` — Spansion EcoRAM (only a press kit).

`http://www.sandisk.com/OEM/ProductCatalog(1410)-SanDisk_pSSD_Solid_State_Drive.aspx` — SanDisk pSSD

`http://www.nemoptic.com/` — Nemoptic makes larger screens, and colour screens.

`http://www.eink.com/` — e-Ink constructs somewhat faster displays.

`http://www.toshiba.com/taec/components/Datasheet/51WHM616AXBN.pdf` — A ‘pseudo’ SRAM device by Toshiba.

`http://www.farnell.com/datasheets/104519.pdf` — An SRAM device by Brilliance Semiconductor, Inc.

`http://www.digit-life.com/articles2/storage/hddpower.html` — Power use by disk devices

`http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX31&nodeId=0162468rH311432973ZrDR` — i.MX31 information

`http://spansion.com/about/press_kit_cd.html` — Spansion EcoRAM (only a press kit).

`http://www.sandisk.com/OEM/ProductCatalog(1410)-SanDisk_pSSD_Solid_State_Drive.aspx` — SanDisk pSSD

`http://www.nemoptic.com/` — Nemoptic makes larger screens, and colour screens.

`http://en.wikipedia.org/wiki/NVRAM` — Forms of non-volatile memory

`http://www.ramtron.com/files/tech_papers/FerroelectricTechBrief.pdf` — FRAM explained

`http://www1.cs.columbia.edu/async/publications/davis-nowick-intro-tr.pdf` — Excellent introduction into asynchronous design by the universities of Utah and Columbia.
`http://vlsi.cornell.edu/fpga.php` — Cornell research on asynchronous FPGA logic
`http://intranet.cs.man.ac.uk/apt/projects/processors/amulet/` — AMULET is an ARM-compatible processor and SoC from the Manchester School of Computer Science.

Notes:

info@openfortress.nl

http://openfortress.nl

Groengemak

Moeiteloos groen