The Free Hardware Design movement

a smaller sibling to the Free Software movement

Egil Möller
<redhog@takeit.se>

2004-12-16

1 Introduction

Is there a free hardware design movement, and if so, in what ways is it similar to, and different from, the free software movement? This text will show the existence of the free hardware design movement by introducing its main players and their activites, the problems they are facing and atempts to solve them, aswell as how to get started width hardware design using some of the most commonly used free developement tools and components. In addition, it will give a short introduction to hardware-near software as a way of bridging the two movements.

I chose this subject as I have been involved quite some in developing free software, but had nearly no knowledge of hardware design and just some on hardware-near programming, and wanted to learn. It turned out, getting started with the free developement tools available was just as easy, and hard, and fun, as getting started with another programming language, paradigm and toolchain, all at the same time. Google, and the previous knowledge of one website [1] at before-hand, helped a lot.

2 History

In the 70:ies, the The Homebrew Computer Club at Berkley did what could from today's perspective be called free hardware design developement, but without much of the automatic tools of today which narrows the gap between software and hardware developement.

In 1998 Reinoud Lamberts announced to the comp.lang.fpga newsgroup his intention to build a free hardware design community, and set up the website Open Design Circuits for the purpose. The community thus formed never actually created any designs, due to endless flamewars about whether free hardware design needed free software hardware design tools, which did not exist at the time. The community spawned and inspired many later projects.

Sometime arount april 1999 the opencollector.org [2] website was opened and started to publish news about free hardware design. In the first news where an interview about free hardware design with Richard M. Stallmanm which I will talk more about later on, and another [3] with Bruce Perens, originally published in Open Season at Upside Magazine (which is unfourtunately not available online anymore). In August the same year, IBM publishes a free motherboard design for its PPC processor, able to run Linux. Late this year, the Icarus Verilog [4] free Verilog simulator is released.

In mars 2000, discussions begun around creating a new, free CPU architecture, and the Freedom CPU [5] project was born,

In january 2001 the WISHBONE specification is released freely by Silicore Corp, in february the opencores.org domain was registered and the site was up and functional the in april, publishing amongst other things a WISHBONE transport implementation, USB and IrDA interfaces and some microprocessors. The project seems to gradualy have been refocusing from the open developement

[1] `http://opencores.org`

[2] `http://opencollector.org`

[3] `http://www.opticality.com/Press/ZopeCorp/UpsideTechnocrat/view`

[4] `http://www.icarus.com/eda/verilog/`

[5] `http://f-cpu.org`

model to freedom idealism since then. Also the Simputer [6] project is officially launched at the National Institure for Advanced Studies in Bangalore India on april 25th this year.

In july 2004, EE Times publishes an article [7] titled "An introduction to open-source hardware development" introducing the concept and discussing revenue models and economics of the movement aswell as comparing it to the Free Software movement.

In december 2004 Neuros Audio publishes the designs for its Neuros Digital Audio Computer, and the press-release makes it to Slashdot.

The Open Collector history page [8] has more details about some of these events aswell as links to even more in-depth information.

3 Terminology

For all political movements and in all political and technical discussions, it is important to have a clear and consice (unambigous) vocabulary covering the topic of interrest. I pressume the reader is knowledgeable with the software and free software movement vocabularies, and words such as source code, libraries, binaries, user space program, kernel and kernel module.

Nearest to these concepts is firmware, which is a program running inside an embedded system within some hardware device, such as a CDROM reader, harddrive or scanner, governing the functioning of the device and its interaction with the main system and its device-driver for the device. A firmware program is a normal computer program, but it is not executed by the main CPU but by a CPU within the device, and it is also stored in ROM, EPROM, EEPROM or flash memmory within the device. To confuse matters a bit, the term firmware is also often used to reffere to startup-codes for the main CPU stored in ROM, EPROM or flash memmory on the motherboard.

Even closer to hardware is the microcode. Microcode is a program stored in ROM or flash memory *within* the CPU itself, targeted at the native instruction set of the CPU, which implements the instruction set for the CPU exposed to normal programs and the operating system. The main usage for microcode is to implement a CISC architecture on top of a more or less RISCy one. The microcode also hides implementation details of the CPU from the user, such as the number of internal buses and their connections to various registers and other circuits, which are often very obvious from the native instruction set and its encoding.

A hardware design is to hardware (a physical device) as source code is to a binary program - it is a formal description which can more or less automatically be transformed into a working device. A hardware design may be a PCB board layout, a net list (a list of logic gates and their connections) or a description in a Hardware Definition Language (HDL) such as Verilog or VHDL. When this is defined, the concept of free software very intuitively carries over to this domain too as free hardware design (Sometimes free hardware is used as a synonym for free hardware design, but as hardware usually refers to physical objects, the English language fuzziness over the meaning of free (libre or gratis), becomes an

[6] http://www.simputer.org

[7] http://www.eedesign.com/article/showArticle.jhtml?articleId=22103383

[8] http://opencollector.org/history/index.html

even bigger problem than when speaking about free software, this term should generally be avoided). More confusing however, open source hardware is not the same as free hardware design (except for politics), but denotes hardware where the software programming interface is publicly documented. It is similar to the meaning of "open" in names such as X/Open, Open Windows etc.. A definition covering a subset of open source hardware, open hardware, is a trademark of the Open Hardware Specification Program, managed by Vincent Renardias.

Some of these, and some other definitions, can be found at the opencollector website [9].

4 Philosophy

Open Collector has published an excellent text [10] paraphrasing FSF:s Why software should be free [11], which shows, argument by argument, the similarities between hardware design and software source-code and the need for the freedom for using them both. I will shortly repeat their most important arguments here:

4.1 Why hardware freedom is not important

Richard M. Stallman have from time to time had the question of free hardware design brought up on some of his many lecture tours throughout the world, and also talks about this in an interview [12] on linuxtoday.com published in 1999, and have always, to the hardware enthusiasts disappointment argued against it, or at least argued that it is unimportant. His argument is mainly based on price of copying - the cost of copying the source of a computer program is nearly zero, and the tools for manipulating software are cheap, which means there is a big social value of sharing source code as many people can learn from it and improve on it. As the cost of copying hardware is much greater, and few people knows hardware design, the social benefits are greatly reduced or even nonexistent.

Also, Mr. Stallman argues, it is questionable if hardware designs are at all protected by todays' copyright, and if something similar to the GNU General Public License could be created for such designs. More likely, only a manifestation of a design, such as a specific Verilog source or VLSI layout implementing the design, is protected under copyright law.

In 2002, embedded.com, a news magazine for embedded system developers, published an article titled Open Source hardware [13], stating nearly any and all reasons why hardware design freedom, as opposed to free software, is not important, impossible from an economic point of view and uninteresting to most hobbyists.

4.2 Why hardware should be free

First off, Mr. Stallmans' argument about cost is only valid when the hardware design is used to create an ASIC - when the design is used for an FPGA or CPLD application, the cost for "running" the design is the same as for running

[9] `http://www.opencollector.org/Whyfree/open_hardware.html`
[10] `http://www.opencollector.org/Whyfree/whyfree.html`
[11] `http://www.gnu.org/philosophy/shouldbefree.html`
[12] `http://linuxtoday.com/news/1999062200505NWLF`
[13] `http://www.embedded.com/story/OEG20020524S0078`

a computer program, and the cost of the hardware needed (experiment board with an FPGA or CPLD together with interface circuit and/or some RAM) need not exceed $50 for a smaller setup.

Also there is neither a semantic nor a philosophical difference between a hardware design specified in a hardware description language such as Verilog of VHDL, compiled to bit-format and downloaded to an CPLD or FPGA, and a normal computer program specified in a programming language such as C, compiled into machine-code and downloaded to an embedded computer-system with an EPROM program memory, some RAM and an ordinary CPU. The sole difference is one of programming language paradigm - the difference between a functional and a structural programming language is as big as the one between a structural programming language and a HDL. From a computer science point of view, a HDL is just a declarative programming language, specifying a finite set of O(1) operations, all to be performed at once at ever step of the execution.

More importantly, the rest of the "social benefits" arguments for free software all apply to hardware designs. Proprietary hardware designs suffer from incompatible interfaces, vendor lock-in using "encryption" of designs and interfaces to "protect ones' intellectual property", and from "ugglycode" under the surface, with bugs impossible for the user of the design to debug, resulting from a mentality of "no one will see this code anyway". Some attempts have been made to remedy some of todays' problems with non-standard proprietary and closed design blocks (so-called IP blocks), notably the attempt to create a market for design licenses as a commodity, but all have more or less failed. Problems free hardware designs are not encumbered with.

5 Licenses

The main problem with licensing hardware designs, as Mr. Stallman points out, is that the hardware itself is not a derivative work of the design. Actually, it is no intellectual work protected by copyright law at all, and creating a physical implementation from a design is not an act of copying. It could perhaps be construed to be an act of public performance, but that is unlikely. What this means is that a plain license for a hardware design can not protect from opaque "copies" of the work to be distributed in the form of physical implementations. The problem, seen from a legal point, is similar to the problem for a software license to regulate "application-servers" (such as web-services) used by third parties who do not receive neither the binary program nor the source over the network. A problem unsolved by GPL 2.0, and intensively discussed for the upcoming new version of the GPL.

One solution to this problem is to abandon the usage of a license altogether and instead bind all users to a contract. This model is still flawed however, as there is a misdirection of blame, or at least responsibility if the contract is breached. Each party to such a contract has the responsibility to ensure that everyone who receives a copy of the design from him/her is bound by the contract prior to the distribution. Should a party fail to ensure this, the party can is responsible for breach of contract, and can be sued for damages. But the receiver of the design is not bound to the contract, and can legally use the design in ways not compatible with the contract. As the one who breached the contract most probably did not earn any money from it, and the one who received the design without entering to the contract probably has earned money from his/her use of the design, and as there is no way to legally stop the misuse, the blame is misdirected, from the design author's point of view.

There are also other more or less usable legal tools available such as patents to protect a hardware design. Patents however are hard and expensive to obtain, and require the design to be new and unobvious (the latter two however, shouldn't be that big of a problem with today's patent policies), which makes them unsuitable for protecting free hardware design.

Despite all these problems quite some groups are using the GPL for their designs. This does secure the Verilog or VHDL source code and any layouts themselves, but not their usages in constructing actual hardware. Various other groups have attempted to create other licenses and in some cases contracts, remedying these problems. Most notably the Simputer GPL, which, though told to be written by a lawyer, seems to fail to distinguish between a contract and a license, and the The Open NDA which is really a contract to which anyone who legally receives a design must be bound, and which requires anyone not to distribute it to anyone who do not agree to the NDA.

A list of commonly used free hardware design licenses is available from opencollector [14].

6 Commercial activities

Around free software, there is a large group of service-centered companies selling support, customization and compilation of the software, as well as maintenance out-sourcing. These companies often release improvements to software they use as well as software developed by themselves independently as free software. Similarly, there are companies selling hardware design compilation, consulting and hardware construction services around free hardware design.

One of these where Silicore Corp., which has in the past released several free hardware design IP cores, as well as free interface standards and free software hardware design tools. Unfortunately, Silicore Corp. seems to have gone out of business, or at least discontinued their website. Their press release [15] from 2003 stating their free hardware design intentions is filled with good arguments for their business model and free hardware design.

Another such company, which still exists is OpenSOC DESIGN [16] who recently released a System C to Verilog compiler as free software, as well as some IP cores as free hardware designs.

Neuros Audio [17] is a company selling portable MP3 players, who recently published their deign and firmware as free hardware design / free software.

7 Hardware-near software

Various projects exists for open boot-up/initialization code, on the PC platform referred to as the BIOS, basic input/output system, as these types of programs

[14]http://www.opencollector.org/hardlicense/licenses.html
[15]http://opencollector.org/Whyfree/silicore.html
[16]http://www.opensocdesign.com/
[17]http://www.neurosaudio.com

usually can provide the hardware abstractions they must use internally to load the operating system to a user program or an OS.

The Open BIOS [18] is a portable implementation of the IEEE 1275-1994 Open Firmware specification, proprietary implemented on several workstation systems from SUN, Apple Computer and IBM. Open Firmware defines a non-architecture-specific Forth runtime environment for device drivers and boot-up-code, which means a driver could be written for the same external hardware, which will work on machines with different main CPU architectures. The project is partly aiming at the embedded market.

On the opposing end is the LinuxBIOS [19] project, which aims to replace the BIOS code for several existing PC motherboard architectures with a slimmed Linux kernel. This projects gives the system much more flexibility than a standard PC BIOS, as any Linux device driver and user-space program could be used together with it to initialize the hardware and load the real OS. For example, loading the OS over the network or from a USB storage device are easy and straight-forward operations. But most importantly, most PC:s sold comes with a non-free software BIOS program, with all the problems of such software, such as nonstandard interfaces and buggy code.

Most of the work done on replacing the standard PC BIOS has sprung from the need of additional flexibility as to how the OS is loaded. The Linux BIOS project was created to easier manage the boot process of clusters of GNU/Linux computers at supercomputer centers. Others have created more application specific programs, such as Etherboot [20], which is an extension to a standard PC BIOS, which van either be loaded from a floppy, or written to an EPROM placed on a network adapter, which implements a DHCP/TFTP-based network boot process, and GNU GRUB [21] which implements the Multiboot specification, a standard for loading operating system images (Linux and some BSD variants follow this specification), on top of a standard PC BIOS, and allows for multiple operating systems to be hosted on the same machine and chosen between at startup.

Another field of hardware-near software is software for embedded platforms and software "accelerators", often incorporating software and hardware components in the same design. Among such works are the Ogg-on-a-Chip [22] Ogg Vorbis hardware music player and KAD.

KAD [23] for Kernel Accelerator Device, is a project recently announced by the German Open Hardware project. The idea behind KAD is to create a PCI device containing an FPGA, and a Linux kernel module able to reprogram the FPGA at boot-time, loading hardware implementations of any needed algorithms into the FPGA, and thereafter use these algorithms to speed up various tasks in the kernel using interface kernel modules. Ideal uses for this should be stream-crypto and pseudo random number generation. This if nothing else is a good argument for the similarities between software and hardware designs - in this scheme a hardware design downloaded to the FPGA would be used just as a software component, a set of binary code run by a slave CPU which just happens

[18] http://www.openbios.info/
[19] http://www.linuxbios.org/
[20] http://etherboot.org/
[21] http://www.gnu.org/software/grub/
[22] http://oggonachip.sourceforge.net/
[23] http://www.openhardware.de/digital/kad/

to be called “FPGA”, and have a quite strange instruction set.

8 Hardware design

Open Collector [24] is one of the most central web-sites of free hardware design, linking to free tools for hardware design and articles about free hardware design.

8.1 Software for hardware design

The task of creating a new hardware device is comprised by several subtasks such as designing its interface with the outside world, its physical form and look, deciding the type of components and IC architectures to use to implement its functioning, designing the PCB to mount the components on, and designing any IC logic needed.

A usual path for developing a new integrated circuit is to first produce a high-level description of the function of the circuit in some language (usually Verilog or VHDL), and run test simulations of the hardware using that description. Then to synthesize an actual network of logic gates and wires, usually referred to as a netlist. Both these stages are highly automatic and usually in the form of a traditional compiler. This netlist is then refined into a physical two dimensional layout of the logic blocks of the design in a phase called place. This layout is then complemented during the so-called route-phase with locations of the connections between the blocks and between the blocks and the in/out-pins of the circuit. These two phases are often done simultaneously, or at least using the same tool however. This stage can be automated, but a good hacker can make a ten-fold better layout than the automatic placement algorithms, at least for complex designs.Also, this stage differs much between different target architectures, such as CPLD:s, FPGA:s and real VLSI designs for ASICs, the latter being a quite a bit more complex and time-consuming task and is usually not done until the second or third generation of a product. For FPGA and CPLD targets, there is an additional stage, where the layout is compiled into a so-called bit-format, a binary, target specific file-format for download into the FPGA or CPLD using an EPROM burner.

Several project groups provides tool sets and tool listings. Veripool [25] publishes several Verilog semi-related tools, and Verilog.net publishes a listing [26] of some free-to-use-but-restricted and some free software tools, including translators between Verilog and various other HDLs such as VHDL. The OpenPPC project, which will be explained later on, provides a listing of tools needed to use and improve upon their design, which is a good starting point in getting tool chains for various tasks up and running. It is available [27] from their homepage. Some of the projects linked from there are especially important, and some others not listed there needs mentioning too.

The gEDA [28] project provides an integrated development environment as well as a tool chain for Verilog development and various circuit layout tools.

[24] `http://www.opencollector.org/`
[25] `http://www.veripool.com/`
[26] `http://www.verilog.net/free.html`
[27] `http://www.openppc.org/tools.html`
[28] `http://www.geda.seul.org/`

One of the most important tools in the gEDA suit is the free simulation and synthesis tool Icarus Verilog [29], which works very much like a compiler for a traditional language with multiple back-ends. It can generate stand-alone Linux binaries which simulates the hardware design when run (including debug print-outs to debug signals), and netlists in various formats (including the industry standard "edif"), and it can also do automatic place-and-route. Other included tools are gSchem for circuit schema construction and PCB for PCB layout as well as various file-format conversion utilities.

There is of course also a GNU Emacs Verilog mode to integrate more tightly with traditional software development tools. It is available at verilog.com [30], from where various other Verilog-related information is also available.

For other languages, Équipe Architecture des Systèmes Intégrés et Micro Électronique has released Alliance [31], a tool chain for VHDL, OpenSOC DESIGN has released a System C to Verilog compiler [32], and for general VLSI design, Compaq Western Research Lab [33] has release the Magic VLSI design tool [34].

The gEDA integrated development environment is unfortunately, and very similarly to many other integrated development environments, not of such a high quality. Especially its built-in editor is lacking and its ability to command the tools involved in the development process highly restricted. On the good side however, the other tools included in the gEDA project, such as Icarus Verilog and gSchem, adheres to a high standard, as does the GNU Emacs Verilog mode.

There seems to be two components of a truly free development environment missing - the netlist to bit-format converter and the software to download the bit-code to an FPGA. These softwares are often provided free of charge by the FPGA manufacturer, so it is not that large a hindrance on the development of free hardware designs. But it is an important missing piece, especially since these softwares are only available for specific platforms (such as Linux on the x86 and Windows).

8.1.1 Non-electronic hardware design

Apart from the (digital) electronic design and simulation tools presented above, there exists generalized physical modelling and simulation environments. For example the Modelica language [35], for which a free software simulator [36] has been developed by The Programming Environments Laboratory at Linköping University, which can be used for simulating e.g. ASICs, or any physical model for which a description has been written. Several such descriptions has been released as free software/free hardware design.

[29]http://www.icarus.com/eda/verilog/
[30]http://www.verilog.com/verilog-mode.html
[31]http://www-asim.lip6.fr/recherche/alliance/
[32]http://www.opensocdesign.com/sc2v.tgz
[33]http://www.research.compaq.com/wrl/
[34]http://www.research.compaq.com/wrl/projects/magic/magic.html
[35]http://www.modelica.org
[36]http://www.ida.liu.se/labs/pelab/modelica/

8.2 Hardware description languages

As mentioned above, hardware description languages, HDLs, such as Verilog and VHDL are the highest forms of hardware design descriptions. Thus, to be able to create a new hardware design, one needs to know one of these languages. Fortunately, there are several good free introductions and manuals available on the net for both VHDL and Verilog, often in the form of the web-pages for some university course on hardware design. Dr Gerard M Blair of University of Edinburgh has published a good introduction to Verilog [37] at his homepage, and for general VLSI design and a VHDL introduction, the course documentation [38] published by Jim Plusquellic of Department of Computer Science and Electrical Engineering at University of Maryland, Baltimore County serves well. Other good sources of information includes the web pages publishing tool-listings mentioned above.

8.3 Hardware designs

The Opencores [39] project has gathered a vast amount of various resources related to free hardware design, as well as different component designs such as IP-cores and connectivity designs and specifications, mainly targeted at System On a Chip (SoC) design. It could be seen as a hardware design version of SourceForge or a cross between SourceForge and Freshmeat.

Most important when designing a hardware system is the interconnectivity between separate subsystems. For SoC design, the subsystems to be interconnected are IP-cores such as a CPU, a UART and on-chip-memory. Several free and semi-free SoC-bus specifications exists and have been compared [40] by Rudolf Usselmann of the Opencores project for use by that project. Out of those compared, IBM CoreConnect, ARM AMBA and Silicore Corp.:s WISHBONE [41], the WISHBONE SoC-bus was selected for the Opencores project, mainly for its great flexibility (it can be used with a variety of connection topologies, bus speeds and widths) and it being the most free and well documented specification.

Other IP-cores available from the Opencores project are 12 arithmetic cores, 3 prototype boards, 25 Communication controllers, of which 13 are WISHBONE compliant, 2 coprocessors, 6 crypto cores, 3 DSP cores, 2 ECC cores, 1 library, 7 memory cores, of which 1 is WISHBONE compliant, 27 microprocessors,of which 6 are WISHBONE compliant, 9 SoCs, of which 6 are WISHBONE compliant, 5 system controllers, of which all are WISHBONE compliant and 2 video controllers, of which one is WISHBONE compliant.

Among these are interfaces to USB 1.1 [42] (slave), Ethernet 802.3 [43], IrDA [44], PCI [45] (bridge), VGA [46] as well as microprocessors such as OpenRISC 1200

[37] http://www.see.ed.ac.uk/~gerard/Teach/Verilog/
[38] http://www.csee.umbc.edu/~plusquel/vlsi/
[39] http://opencores.org/
[40] http://www.opencores.org/projects.cgi/web/wishbone/soc_bus_comparison.pdf
[41] http://www.opencores.org/projects.cgi/web/wishbone/
[42] http://www.opencores.org/projects.cgi/web/usb1_funct/overview
[43] http://www.opencores.org/projects.cgi/web/ethmac/overview
[44] http://www.opencores.org/projects.cgi/web/irda/overview
[45] http://www.opencores.org/projects.cgi/web/pci/home
[46] http://www.opencores.org/projects.cgi/web/vga_lcd/overview

[47] and Z80 [48].

8.3.1 "Real" computers

Most free hardware designs are targeted for the embedded systems field, as the entry-barrier to this field is much lower than for the desktop computing field. However, even in this field there is work on free hardware, which is particularly important due to the ubiquitous nature of desktop computers. Some of these projects aims to design a whole new system, others to replace one or more parts.

The LEOX [49] project aims to build a free set of software and hardware components mainly for embedded systems. The projects main component is the LEON core [50], a free AMBA SoC-bus compliant VHDL implementation of the open SPARC V8 [51]. A similar project, which however does not attempt to re implement an existing standard, but to create a new RISC based CPU design, is the Freedom CPU [52] project.

The Simputer [53] project aims to build a hand-held low-cost alternative to the PC. It is a full system containing both free hardware design and free software components. Ideologically it is more focused on narrowing the digital divide between developing countries and the west than providing a truly free platform, however.

The aim of the OpenPPC [54] project is a bit narrower - to build a free motherboard design for the IBM PowerPC architecture able to run PPC Linux.

9 Conclusions

From this vast material of existing free hardware design, free software hardware design tools and free firmware projects, we can only conclude that this is a very live movement, and far from as unimportant as Mr. Stallman argues. However, the movement is much younger than the Free Software movement, probably due to cost/availability and awareness issues. The movement is also much more chattered than the free software movement.

License and source for this document

The source for the current version of this document is available from the authors' website [55].

[47] http://www.opencores.org/projects.cgi/web/or1k/openrisc_1200
[48] http://www.opencores.org/projects.cgi/web/wb_z80/overview
[49] http://www.leox.org
[50] http://www.leox.org/resources/hw.html
[51] http://www.sparc.org/
[52] http://f-cpu.org
[53] http://simputer.org/
[54] http://openppc.org/
[55] http://redhog.org/Projects/School/FreeSoftware/freehardware.lyx