

Xiang Li

December 25, 2008

This page is intentionally left blank

Contents

2	Open Cores in a Commercial Perspective	5
2.1	Basic Concepts	6
2.1.1	What is Open Core?	6
2.1.2	Formal Definition of Open Source	6
2.1.3	Licenses Involved to Evaluate	7
2.1.4	GNU and FSF	7
2.1.5	Free Software \neq Free of Charge	8
2.2	BSD License	9
2.3	GNU Licenses	11
2.3.1	GPL	11
2.3.2	LGPL	14
2.3.3	Evaluations on the GNU Licenses	16
2.4	The Price for Freedom — Comments on the GNU Philosophy	17
2.5	Developing Open Cores or Not — How Open Source Products could Benefit	20
2.6	Utilizing Open Cores or Not — Pros and Cons	21
2.7	The Future of Open Cores	23
2.8	Conclusion	24
	References	25

This page is intentionally left blank

Chapter 2

Open Cores in a Commercial Perspective

Well, I know the title of the chapter may look weird as it appears in an engineering thesis, but it was really the most primary task had to be clear before my thesis project started.

Unlike universities and academic institutes sometimes dedicate on science research, companies on their behalf always take profits as the top objective in business. So in the companies, every project has to start by satisfying managers with proper answers to the questions like: Why should the company do this? Will we gain more than we pay and the risks we take? . . .

My thesis project also fell into this case because which was performed in a company named ENEA. ENEA sells software and services as their main business, but they were interested in trying hardware at the time of January 2008. The attempt finally gave birth to my thesis project—to compose an embedded platform from scratch by exclusively using open cores.

As a result, to evaluate the feasibility as well as the risks of using open cores in commercial products became the first thing I had to do. This is also a clearly specified task if you take a look at the announcement of the thesis project in Appendix [?]. Otherwise if we cannot prove it is a good idea, all the following implementations will be useless.

Almost all open cores are covered by various licenses, including famous GPL, LGPL and BSD license. Therefore, studies to those licenses which cover the open cores we are going to use will be inevitably involved at the same time when evaluating open cores. In this chapter I will introduce the licenses.

After reading the chapter, I hope my introductions and evaluations to open cores and the licenses could help you to answer the questions: If I am a manager from a company like ENEA, should I develop new open cores? Should I improve existed open cores and integrate them into the next product?

2.1 Basic Concepts

As a company, a good understanding to open cores will be mandatory to make decisions whether we should use them or not. In this section I will introduce several basic concepts regarding to open core that the reader should know.

2.1.1 What is Open Core?

Open Core is a shortening of Open Source Intellectual Property (IP) Core, which is a combination of the concepts of “open source” and “IP core”.

Here I am not going to spend too much words on introducing IP cores as well as why we need them in electronics engineering, because those are quite well known that you can find in plenty of articles like [1, 2]. In short, the increasing complexity of electronic systems and time-to-market pressure force engineers to utilize already made blocks, i.e. IP cores, as many as possible into the next system, which is so called the design reuse methodology.

Actually semiconductor IP cores have a broad definition and can be in any form of “reusable unit of logic, cell, or chip layout design” [3], but in this thesis, I would like to narrow down the concept “core” of the open core as digital logic designed by HDL code and targeting to FPGA/ASIC. This is because all the open cores used in the thesis project are of this type.

2.1.2 Formal Definition of Open Source

When coming to the other concept “open source” of the open core, most people simply just take it literally as “you can look at or get a copy of source code”. However, this is a common misunderstanding.

Open Source Initiative (OSI) [4], an official organization of open source community, has a formal definition to the term “open source”, which can be found on the Internet at the link [5]. I cannot cite the definition here, because it is too long to put into a thesis.

If you haven't read the concept before, you will find the open source is a quite complicated concept than you ever imagine. It does not only refer to the access to source codes, but also defines lots of criteria needed to follow. The reason I'm talking about the formal definition here is because it might be important for you to be aware of the actual effects from those criteria, before you decide to publish your products as open source or utilize something in the form of open source (naturally including open cores).

Personally I hate to define things in such a complicated way, although it is unfortunately the truth for the most definitions. But still, I would suggest at least we simplify the open source in this thesis by replacing the formal definition with "source code that covered by certain open source licenses", so that we can concentrate our study on the open source licenses only.

A list of approved open source licenses can be found in OSI's website [6]. Different licenses in the list may have different rules and regulations, but if a source code is covered by any one of them, you can call it open source.

2.1.3 Licenses Involved to Evaluate

Now we have established the open source licenses as our target. But don't be scared by the long list of the licenses from the OSI. Luckily only few of them are involved to evaluate in our case. They are GPL, LGPL and BSD License.

All open cores we used in the thesis project come from the OpenCores organization [7], which is a dedicated (and perhaps the only) community for IP cores written in HDL. And all of cores we chose are covered by either the LGPL or the BSD license. Due to the LGPL is based on the GPL, I will introduce all 3 licenses in detail in the later sections.

Actually the open cores used in the thesis are not covered by the exact BSD license, but a "BSD-style" license. However the introduction to the BSD license will be still helpful.

2.1.4 GNU and FSF

As the GPL and the LGPL will be introduced later, I would also have to mention several words related to them.

The letter "G" in the GPL and LGPL stands for "GNU", which is a name of a project to develop a Unix-like operating system that is completely free software. The GNU licenses were initially designed to protect the liberty

of the free software in the project without being violated¹. But later they became more and more popular and now are widely used to cover a large proportion of free software all over the world.

Sometimes the GNU is also treated incorrectly as an organization that is responsible for the project, probably because its website is named as www.gnu.org. But actually it is Free Software Foundation (FSF) that takes the role. The FSF is a corporation founded to support the free software movement as well as to be a sponsor to the GNU project. So don't be confused when you see "Copyright (c) <year>, Free Software Foundation, Inc" in the GNU licenses.

Both the GNU project and the FSF were started by Richard Stallman, a legend because of his contributions for the free software movement. In my first draft, I tried to tell a little about the story of Richard Stallman and his GNU, but finally decided to cut this part because it ate up too much space in the thesis. If you feel interested, some of good articles can be found at the following links of Wikipedia [8–13], also at the websites of the GNU [14] and the FSF [15].

2.1.5 Free Software \neq Free of Charge

The free software movement we mentioned above is a social movement aims to prompt people's freedom on accessing and improving the source code of software. If some software truly assures the right to the user, it can be called as free software.

Free software is a concept close to the open source but highlights more on the right of the freedom [16]. Similarly, it is also often misunderstood literally as "the software that is free of charge". As I will go back to this concept again in the later section, please just remember the official explanation to the concept here: "'Free software' is a matter of liberty, not price. To understand the concept, you should think the 'free' as in 'free speech', not as in 'free beer'" [17].

The explanation is meaningful. It implies people could sell free software in case of the freedom is guaranteed, so we can somehow make profits by utilizing free software, so as to the open cores.

So far, all the related concepts have been introduced. From the next section, we will start discussing about the open source licenses, and then make analyses for the open cores in a commercial perspective.

¹Just like the BSD license was firstly used for the BSD—the name of another operating system.

2.2 BSD License

As the BSD License is relatively simple comparing to the GNU licenses, i.e. the GPL and the LGPL, I introduce it first.

The Berkeley Software Distribution (BSD) license gets its name because it was first designed to cover a Unix-like operating system developed by University of California, Berkeley [18]. Later it was revised by removing a clause which was too impracticable and limited the license to be widely accepted. This story can be found in [18, 19]. So now the BSD license is also called the “new” BSD license, or simplified or modified or revised or 3-clause BSD license.

The BSD license is the 3rd most popular open source license according to the article [20], which also mentions that the first two ahead of it are the GPL and the LGPL, and those two account for almost 80% of all open source licenses in use. I don’t know how the data is calculated, nor have no idea whether it is accurate. But it shows the truth that the licenses are quite widely used in the open source world.

The reason that BSD license is popular is because people like it. People like the BSD license is probably because it has very few restrictions, both for authors and (especially) for users. The full text of the BSD license can be found in [21] or somewhere else.

Let’s take a look at the 3 clauses of the license.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- *Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.*
- *Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.*
- *Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.*

The 3 clauses are really merciful which assert only minimum requirements comparing to other open source licenses.

Clause 1 says you cannot remove the text of BSD license from the source code. Clause 2 says if you will redistribute it in the form other than source code, you have to announce that your products contain something designed by someone else. I think both 1 and 2 are the most basic things that everyone should do by conscience, even if they are not required by such a license. Just like when writing an article, you need to write a reference to list all other contributors of your work.

Clause 3 is a little complicated to understand. It is used to prevent the original author's name of the source code from being abused, and the drawbacks of the source code from being incorrectly attributed. For a simple example, let's say I somehow get the source code of a newly designed operating system from Microsoft which is under the BSD license (although this sounds like impossible). If without the limitation of clause 3, I can then announce to users like: Microsoft grants me the right to improve the software; if you feel it is good, that is all my contributions; if you find bugs inside, that are all Microsoft's faults.

Under the 3 clauses, there is a long paragraph of disclaimer in the BSD license. According to it, basically the source codes covered by the license provide no warranties to users. This is good for authors since they promise nothing through the license. And this is fair because most authors give the source codes for free when they choose the BSD license, so of course they should not be responsible for any liability. But for the users, before going to work on the source codes under the BSD license, please note the risks you may take.

Now let's evaluate the BSD license in a business world.

First of all, the license talks nothing regarding to money. This means technically one can develop source codes under the BSD license and sell it at any price he/she wants. Although this is not a usual case, because if one would like to make money by selling products, he/she would rather choose a commercial license stricter than the BSD license which for instance prohibits making copies and redistributions freely.

Another interesting point is that the BSD license sets no limitations on keep using the license after redistributing. This is different from the GNU licenses. As everyone knows, one of the goals that we choose open source products is to modify, improve and then redistribute them, so that benefit through the process. As the BSD license requires nothing on what license will be used for the modified versions, it is totally possible to use another commercial license instead of the BSD license. This is very good news for vendors, because they can take a BSD licensed IP core, integrate it into a new product, and then sell it as the way they want with their own commercial licenses.

For open cores, perhaps the most attractive part of the BSD license is that it doesn't ask you to publish the modified source codes when you redistribute (or sell) your products in a non-source form. On the contrary, the GNU licenses do. Thinking about the open cores which are in the form of HDL source codes, after the products are finally built and ready to sell, the open cores will be absolutely not the original HDL codes, but become silicon chips. Because don't have to open source for the silicon chips as they are in the non-source form, companies are able to safely make modifications to the BSD licensed open cores, combine them together with their own patents, meanwhile happily selling the new chips without telling their competitors the design details.

To conclude for the BSD license, it is a quite loosely restricted open source license. Simply put, an modified source code that originally covered by the BSD license even does not have to be open source any more. This makes the license completely compatible for commercial purposes. Companies are welcome to utilize open cores under the BSD license in their business, using their own commercial licenses instead, as long as keeping a reference together with the products which shows certain BSD licensed open cores are contained, such as including a copy of the license in the user manuals.

2.3 GNU Licenses

In this section we are about to introduce the GNU licenses, i.e. GPL and LGPL.

GNU General Public License (GPL) is one of the most popular (and perhaps the strictest) open source licenses published by the FSF. GNU Lesser General Public License (LGPL) is a supplement of additional permissions to the GPL by removing some limitations from it. So to speak, the LGPL is based on the GPL rather than an individual license. So we start with the GPL and then take a look at how the LGPL are "lesser" than the GPL.

2.3.1 GPL

The full text of GPL can be found at [22]. To be honest, it is quite boring to read such long legal document for an engineering student like me especially who is not a native English speaker. It spent me a lot of time to understand what the sentences are really taking about. So in the following paragraphs I will describe my understandings and hope they could give a hand to the reader when start reading the GPL or the LGPL yourself.

The GPL is a very strict open source license that designed to make source code to become free software and keep it as free software forever. According to the formal definition of “free software” [17], a program is free software if its users have all 4 freedoms:

- *The freedom to run the program, for any purpose;*
- *The freedom to study how the program works, and adapt it to your needs;*
- *The freedom to redistribute copies so you can help your neighbor;*
- *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.*

So if you are going to use some source code covered by the GPL, you will have the right to “run, copy, distribute, study, change and improve the software” [17]. While if you decide to use the GPL to cover your designed programs when publishing them, you grants these rights to your users or customers.

The above rules may seem OK since they are basically similar as the BSD license which also gives you the same freedom. But if you feel that is all for the GPL, you made an absolute mistake. Because at the same time GPL grants you freedom, it also defines obligations that you have to perform.

There are 4 main obligations can be summarized among the long text of the GPL: 1) Keep using the GPL forever; 2) Provide source code in any case; 3) Publish the modified parts; and 4) Make the GPL cover the entire new project.

First and foremost, the GPL will keep covering certain source code forever, and there is no way to remove or bypass it since it is applied. If you decide to edit GPLed source code and redistribute it, the GPL will be the only choice of the licenses for the revised work. You cannot use any other alternative licenses to cover the new work instead of the GPL, because “This License gives no permission to license the work in any other way . . .” (section 5). There are many licenses that have no conflicts with the GPL in principle, like the BSD license. This is called “GPL compatible”. For example, you can create a new project by combining 2 separate works that the one is covered by the BSD license and the other by the GPL without any problem. However, only the GPL can be used to cover the new project, but not the BSD license. So once the GPL is there, it will be always there; once you decide to use GPLed source code, you have to keep using the license forever for the improved versions in the future.

The 2nd obligation well expresses the thought of the GPL as an open source license—the source code has to be provided (opened) in any case. If you redistribute in the form of the source code, naturally the codes should be open. Even if you are going to redistribute a product in a non-source form, according the section 6 of the GPL, the source code that covered by the GPL still has to come up with the products. For instance if a company produces and sells software that under the GPL which is in non-source form like binary or executable files, usually this is done by the company that providing either an extra CD including the source code or a web server which allows the users to download those files freely. This obligation makes the GPL stronger than many other open source licenses like the BSD license, which do not ask for an accompanied source codes. When coming to the open cores, this means at the same time silicon chips are sold, certain HDL source codes and/or design details have to be public.

The 3rd obligation makes the GPL even stronger. It forces its users to publish the modified parts when providing the source code along with non-source form products, i.e. show the source codes which generate the final products. So when selling improved open cores you cannot just provide the old source codes without the details of the modifications. When designing a new project by reusing already made blocks, it is a usual case that some necessary changes have to be made so that the blocks can be adapted into the new project. Or, sometimes vendors may spend a lot of efforts on improving the cores to achieve a better commercial quality. However, according to the GPL, no secret is allowed to hide. All these modifications and improvements have to be public, although not everyone might be happy to do so.

Most disagreements and arguments come from the last obligation which called many criticisms [23, 24], that the GPL must cover the entire project, or in another way of saying the GPL is infectious. In section 0 the GPL defines the term “modify”. It says “To ‘modify’ a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy.” And when “Conveying Modified Source Versions”, in section 5 term (c) the GPL makes the rules: “You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, . . . , to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission of license the work in any other way . . .” These clauses make the GPL spread to the parts which could have not belonged to a GPL covered work, so this is called “infection”. When we are going to reuse GPLed source code for a new product, it will be surely fallen into the definition “modify”. And when you redistribute the improved version, section 5 will take effects to force applying the GPL to the whole new work, which means you have to unfortunately open source for the entire work,

including the parts which are not initially covered by the GPL. For a simple example, if your program which has 4,950 lines and a 50 lines function copied from a GPLed software, all the final 5,000 lines of codes have to be open source and public to downstream users, even though the GPLed code only counts 1% in all of codes.

2.3.2 LGPL

The last infectious obligation of the GPL is obviously much too strong than many users who are not that enthusiastic in free software movement could accept. Indeed, many companies are afraid that their patents and proprietaries could be violated when combing them with the GPLed things, so they keep themselves away from the GPL. Therefore, a light version of the GPL, i.e. LGPL, is developed by the FSF by removing the infection attribute from the GPL.

The LGPL is previously named as “GNU Library General Public License”, which shows it is primarily developed for libraries. Because the GPL is too strict and infectious, if it is applied to a library, all programs that linking to this library will have to become open source. This is not the case that many developers, especially commercial companies, would like to see. And as a result, the GPLed library may be gradually forgotten by people. To solve the problem, a compromised license, i.e. the LGPL, is designed and used particularly for libraries. Soon, the FSF realized that 1) LGPL can be used for not only libraries but many other software as well, and 2) the choosing between the GPL and the LGPL by authors is a strategy of development [25], but not only depend on whether it is targeted to a library or not, so now the LGPL is renamed to “GNU Lesser General Public License”.

The LGPL is a set of additional permissions added to the GPL. By those permissions, the LGPL removes the last obligation of the four I described above. But this is the only difference between the LGPL and the GPL. All the other three obligations of the GPL still remain.

In section 0, LGPL defines several more terms than the GPL:

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library.

In section 4 “Combined Works” it says:

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications . . .

And in section 5 “Combined Libraries” it says:

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice . . .

By these clauses, the LGPL clearly separates 2 different sets of components, the “Library” that is a LGPLed module and an “Application” from somewhere else which isn’t governed by the license. Since the LGPL is initially designed for libraries, it keeps using the phrases like library and application.

According to the LGPL, it will not affect other modules which just connect to a LGPLed module. We can draw a picture to explain the rules. Suppose we have a software project including 2 existed libraries and a lately designed application based on them. As Figure 2.1 (a) shows, Library A is covered by the LGPL, while Library B is from some company and covered by a commercial license. The application links to the 2 libraries. In this case, the whole project is a combined work, and you are allowed to license the project under the terms you want, as long as these terms do not conflict with the LGPL, and also guarantee that the LGPLed library is still open source.

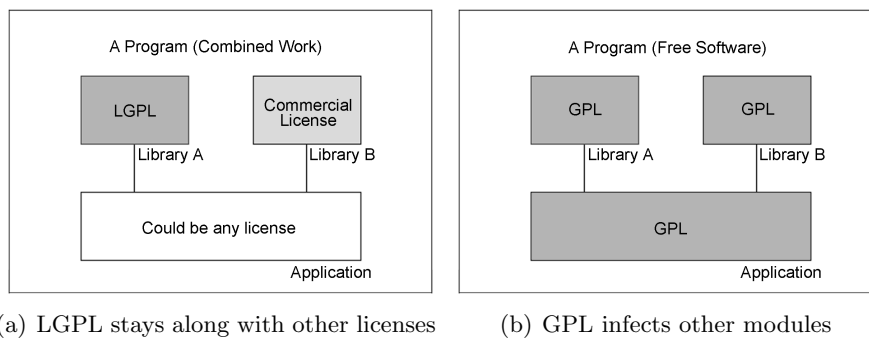


Figure 2.1: Difference between LGPL and GPL

However, the scenario will be totally different if Library A is covered by the GPL. If it is, because the GPL forces the whole project be licensed only in the way of the GPL, all the rest parts of the project, i.e. the Library B and the application, have to become GPLed whatever their previous licenses are. So, this is how the GPL infects.

2.3.3 Evaluations on the GNU Licenses

So far, the introduction to the GPL and the LGPL is finished. It is time to go back to our topic to discuss about how the open cores will be influenced by these licenses.

My answer is that: generally, the open cores that covered by the LGPL are OK to use in commercial products if the company won't spend too many efforts on improving the cores; while the GPLed open cores are not suggested for commercial purposes, unless they are going to be used individually per silicon chip.

Basically the open cores covered by the LGPL are free to use is because they are not infectious like the GPL. So there is no worry when connecting open cores and proprietary IP cores together to compose a larger system. Besides, most likely the utilized open cores will not be modified too much than the initial version. Otherwise companies would rather like to invent a new core than reusing an existed one. So this means open source for the improved open cores required by the LGPL is acceptable for the companies.

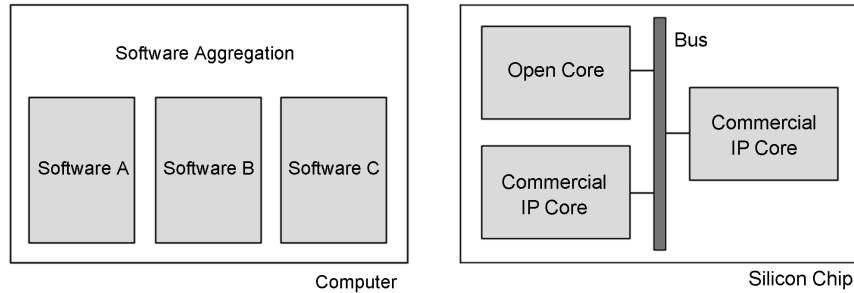
The reasons that the GPLed open cores aren't suggested are 1) the GPL is infection, and 2) the difference between open source software and open cores.

For software, the GPL defines a concept called "aggregate" in section 5: *A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" . . . Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.*

By this definition, the GPL won't infect if any 2 programs just stay together and are not related to each other. For example, a web browser and a media player are stored in the same computer. The license of the media player will not be influenced by the web browser if it is under the GPL. Or if a compiler is GPLed, the source codes being processed by the compiler do not have to be open source.

Unfortunately, the concept of aggregate doesn't work for open cores in hardware world. As everybody knows, although open software and open cores are both in the form of source code at the beginning, open software will be finally compiled into binary or executable format which is still software, however the HDL codes of open cores will be synthesized by EDA tools and transform to hardware at the end. Like Figure 2.2 shows, in a hardware system,

every open core will be connected together via a bus or other interconnection structures. As a result, the definition of aggregate is no longer satisfied in this case, and the GPL will infect the whole system.



(a) Software aren't linked in computers (b) IP cores link with others in chips

Figure 2.2: Difference between Open Software and Open Core

So to speak, GPLed open cores are not suggested to be used because it will force other parts of system to become open source, which is generally the case that companies would not like to see. There are some really good open cores under the GPL like the LEON processor with SPARC architecture [26], but unless you want to use them solely to form a system so that no other cores to be infected, i.e. to make a silicon chip includes only one open core, open cores covered by the LGPL will be always a better choice than the GPL.

To sum up this section, both the GPL and the LGPL are very strict open source licenses. They grant users the freedom on using source codes under the licenses, meanwhile stipulate obligations that have to follow. Generally, the open cores under the LGPL are no problem to be used for commercial purpose, but the GPLed open cores are not because they are infectious. When the products including open cores under the LGPL are published, basically a company has to do the following: 1) announce the open cores contained in the product; 2) attach a copy of both the GPL and the LGPL; 3) publish the source codes that generate the final product.

2.4 The Price for Freedom — Comments on the GNU Philosophy

I have introduced the GNU GPL and LGPL in the previous sections. Now it is time to talk about something in a higher level—the GNU philosophy

we could feel from the licenses. Let's start from a misunderstanding of the free software.

For a long time many people think the word “free” of free software means “free of charge” or “zero price”. So they feel strange and uncomfortable when they are asked to pay for the copies of free software (like some Linux products) from distributors. They may query “Isn't this FREE software?”

Actually this is a common misunderstanding for free software according to the GNU¹. In dictionary there are 2 meanings for the word “free”. The one is “not under control or not subject to obligations”, i.e. freedom, and the other is “available without charge”, i.e. no cost for money. The GNU says clearly in the 3rd paragraph of the preamble of the GPL as “When we speak of free software, we are referring to freedom, not price.” They also emphasize this point many times by saying “Free software is a matter of liberty, not price. To understand the concept, you should think of free as in free speech, not as in free beer.” [16, 17, 27] In fact, the GNU even encourages people to charge as much money as they can by selling free software [27].

From which we can feel that at the same time the GNU is struggling on protecting the freedom of knowledge accessibility, they are also trying to clarify that the free software licenses won't prevent people from making money. So on one hand the GNU believe the knowledge should be the wealth of all human beings and no one should set barriers to limit its propagations, so that people could benefit from the knowledge. On the other hand, they hope the businessmen can charge at any price as they want to get a substantial profit by selling free software as long as they follow the licenses and promise no restriction on the freedom.

This sounds great, like a win-win solution that everyone benefits. The producers get the money and the consumers get the knowledge. But the question is will this become true?

Personally speaking, I totally like this philosophy, because it describes a great idea that everyone shares their knowledge and all the people work together to make the world a better place. I would also like to show my respect and the best praises to those free software developers and to their work, and dream that I could become one of them in future.

However, at the same time I need to point out the problem—a paradox regarding to the concept of “free”. The GNU argues that the freedom has no conflict with making money. But from my perspective, the “freedom” and the “free of charge” are actually the same thing, or would rather say, the “freedom” will finally result in “free of charge”. So to speak, the GNU

¹In this section I use the word GNU to stand for the official organization.

philosophy is incompatible with the business rules that widely applied in our economic society nowadays.

This is easy to understand, because it is our human nature that we will always choose the better way for ourselves. If there is something that you can get freely, no one will pay to get it anymore. Take the air as an example which everyone can breathe. Would you like to spend money on that? If a sales man comes to you and says “Hey, we have a new product called free air’ which has no difference with the normal air but just cost 100 dollars”, will you buy it? So the easier to get something, the cheaper it is, and vice versa. This explains why water in some countries like Sweden is free of charge, but in some others needs to pay, and may be even more expensive than a life in deserts. Therefore, at the same time the GNU licenses is used to grant the freedom to users, it makes free software easier to get. And thus will reduce the price of the free products, meanwhile lower down the profit that developers could have had.

Now let’s think about what will happen from a business perspective. Suppose we are a software company named A Company that is going to sell a newly developed brilliant product. Since impressed so much by the free software movement we decide to use the GPL to cover our source code to make the product free software. By selling each copy of the software we will receive 1,000 dollars. That sounds perfect! Everything is fine in the first several days because the product is quite excellent and has a good market, until another B Company appears. The B Company was our A Company’s customer at the beginning so they got a copy of the source code together with the product. However, soon they start selling a similar product easily developed based on our source codes, which costs only 500 dollars. This is immoral, however amazingly we cannot prevent them according to the GPL. This is because 1) the GPL asks us to provide the source codes when selling products; 2) the licensees are allowed to redistribute (convey) the source code under the GPL; 3) “You may charge any price or no price for each copy that you convey ...”, from the section 4 of the GPL. All of above show that the B Company’s behavior is legal! So now, assuming you are the next customer coming to buy the product, there are 2 choices between A and B, even if you know that B is the immoral one, what do you do?

The story above includes only 2 competitors A and B. Actually if a 500 dollars price is still high enough for a proper profit, more companies will continue joining in to sell the products at an even lower price. All of these actions will force the price of the product to become lower and lower, until the profit is low enough that no one else would like to waste time on doing these things any more. Figure 2.3 shows the trend.

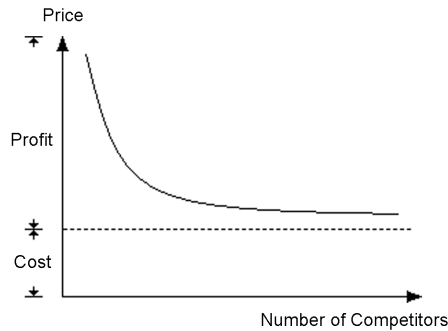


Figure 2.3: Increasing Competitors Forces the Price Lower

Besides, in our modern society the Internet is so popular to everyone. If the source code is freely to get from the network, people will likely download it instead of buying it if they feel the price is too high. Although usually it is a skill to compile the source code that not everyone understands, many of them would still tend to find the instructions and to learn rather than pay for the product to the companies.

In a sentence, open source makes products unprofitable. This is the price paid for the freedom. That's why most free software is free or has a very low price, because the market forces it to be. That's why most big companies do not support free software movement nor would like to make their products open source, because it will reduce the money they could have earned.

2.5 Developing Open Cores or Not — How Open Source Products could Benefit

I mentioned 2 questions at the beginning of the chapter to figure out. Now it is time to solve the first one: As a company, should we develop new open cores for sale?

Generally speaking, the answer is NO if you expect to make money with the cores, because we have discussed in the above section that making a commercial product open source will result in less profit than not doing so.

However, the open source things are still good in some other ways. In this section, I will introduce how the open source products could benefit the business.

The first benefit is that, developing open source products is always a good

strategy to make a company attracting more publicity. Many open source products are considered with a better quality¹ because bugs are easy to find out in case of open source, so usually they have a good reputation and people always show respects to them. Therefore, open source things are easier to be propagated through the Internet without much advertising. The reason makes producing an open source product a helpful way to announce the company itself. This can be compared to the discount information of the supermarkets. Every store frequently puts up some eye-catching posters or slogans like “buying a dozen coke gives 70% off” etc. When people are attracted into the store, they are likely to buy many things else expect for the discounting products. So, open source products could take this role as advertisements.

The second benefit gained by open source products is that this is a good way to sell services. Because it is most likely that many users who are going to use open source things may not know how, or a company that going to include an open core into their next product may not understand the design details of the open core and how to adapt it, in such cases a good market emerges by selling services, especially for those consulting companies which have a good background on providing services. In fact, almost all companies who provide IP core products also support services at the same time.

The third benefit is that developing open source products may give a push on selling related physical products. The idea of GNU philosophy actually makes the knowledge free to get, but after all the source codes have to run on some physical things somehow. If the software is for free, we can earn that part back by selling hardware products. So silicon companies might like open cores like the LEON processor if they can increase the sales of chips. Or embedded company may be happy to develop open source software for their hardware system, because this won't affect the sales of the products like digital cameras, mobile phones or PDAs etc.

2.6 Utilizing Open Cores or Not — Pros and Cons

In this section I will try to answer the other question: if it is not suggested to develop open cores for sale, how about utilizing the existed ones? Is it good idea or not?

Please note the difference. “To develop” open cores means to design an open core from scratch for sale (or help selling), while “utilizing” open cores means to reuse open cores got from somewhere else into our next products.

¹Just think about Linux versus Windows.

To answer the question will result in an evaluation of the pros and cons of open cores. This is why we will discuss about the advantages as well as the possible risks of using open cores in this section.

The most obvious advantage that using open cores could bring is to accelerate the design of new products. To reuse blocks that have been designed is much better than re-design them from the beginning.

Another most attracting point of open cores is the price, because most open cores are free to get. This lowers down a lot the threshold of the money to start a new project, and thus especially good for small companies. Take our thesis project as an example, all we needed for the project are just a develop board which cost \$329 dollars, a PC, and several cables. All the IP things are free of charge. If we were using commercial cores, the price must be considerably too high to afford.

The third advantage is that open cores are adaptable. Because all the source codes are open, necessary changes can be easily made to integrate open cores into new systems. Comparing to commercial cores, usually they have certain techniques to hide these design details which users cannot touch. To make changes on these cores therefore costs quite long period which requires the help from the vendors.

And one more advantage that often be ignored is that open cores could make users feel safe. Because open cores have all their details public, users are able to know the design detail, although they may not necessarily and usually won't do so. This makes it impossible for the developers to stand on your back and involve some harmful designs in the products but without telling you. This also makes open cores well suitable for security critical products that used for national or military purposes.

At the mean time open cores also have disadvantages that need to take into consideration.

The most impressed one for me is that open cores are often less supported than commercial products. So it depends on the capability of the develop team in a company to decide how fast the open cores could be adapted into new products. Here the "less supported" may represent in many aspects like less documents, no telephone support etc. Most open cores are designed by engineers or fans individually for the purpose of interest. After a tough design work, only a few of them can still have passion on working for services like writing good documents. For example, in our project we used an open core named OpenRISC. I followed a long time to do exactly as its document said but there were always problems. Finally it turned out that the date of the document was written at the year 2001 while the latest revision of the

source code I was using was at 2006. Except for the documents, when you have any questions regarding to the open cores, there is no way to find a technical support. Although there may be a forum on the Internet which you can post a message for discussing, most likely you may not get the feedback in an expected time.

Another disadvantage is that open cores are often less verified than commercial cores. Because verifying hardware cores needs quite a lot of complicated procedures and equipments, only the powerful companies could do these things well. They can even build a real chip to test the quality of the cores, But this seems too hard for individual developers.

Besides, please notice that the using of open cores could suffer higher risks on law or patent issues. These risks may come from 1) the charge from other business competitors, big companies, and open source opponents; 2) The open source licenses were initially designed to cover software. When coming to the hardware, they may not be that reliable on protecting open cores; 3) the explanation and execution of the law of the licenses may vary from place to place.

Above all, I want to say the question “should a company improve existed open cores and integrate them into next products?” has actually not an absolute answer. It depends on the judgments of the smartest project managers. It is definitely not an easy decision for a company to use an open core which is not familiar before. The answer could be affected by the factors like budget, time, product volume, design team, as well as many others.

However, there is one definite conclusion I can draw here, that every company should keep an eye on open cores. I know big companies often have specialized departments and engineers which continuously collecting the information of IP cores and evaluating them, so that they could find the cores they need at the first time and utilize them to accelerate the developing. If you pay efforts on studying and testing open cores, maybe some of them will be exactly what you want at the next time.

2.7 The Future of Open Cores

As a part of open core, it is interesting to discuss about the future. In this section I will give my prediction.

Basically, I am sure that open cores will keep growing. First this is because people need to reuse cores to accelerate the system development. Second, due to the source codes of open cores are fully open, everyone could get in

touch with the open cores and work together to improve them. So although there are some open cores may look not well enough today, they will become better and better. If take the free software community which is growing all the time as a reference, we can foresee that open cores will follow the same track, because both of them are open source.

But at the same time open source is against the business rules, as we discussed this in previous section, i.e. make knowledge unprofitable. So open cores will not be easily accepted by big companies, nor will be welcomed by investments, because in most cases the investments only like things or industries which could make more money back. This means there will be no strong stimulus which forces open cores to develop in a rapid speed.

In a little bit detail, I guess those open cores that have simple structure, like UART, mouse controller or something similar that is easy to design and implement, will get a better change to become popular. They are easy to be developed by small teams or even individual engineers and are easy to achieve a good quality. As they are free and good enough, why not use them to accelerate new system developing. On the contrary, the complex cores like processors will be held tightly by big companies for still a long time. This is because complicated cores have much more profits than simple cores. Suppose if all of our computers are using open core processors but not the products from Intel or ARM, this will make them crazy.

So my prediction for the future of the open cores would be: generally speaking, open cores will grow up, but could not be too fast. And simple cores will get a better change to become popular.

2.8 Conclusion

In this chapter we talked about open cores in a commercial perspective.

Firstly we talked about some basic concepts about open cores like what are they. My definition is that the open cores are IP cores whose source codes are covered by open source licenses for example the BSD license or the GNU licenses.

Then we introduced the BSD license, the GPL and the LGPL in detail, also pointed out that there will be no problem to use the open cores that covered by the BSD license or the LGPL into commercial products. But the GPL is not suggested mainly because it will infect other parts of systems, and force them to become open source too.

After that we discussed a little about the free software philosophy, which depicts an attracting image that our society could be. However, I feel that is impractical because it makes knowledge free to get and thus unprofitable, which against the business and economic rules in our society. So this philosophy will not be appreciated by those big companies which earned a lot of money on selling proprietary products.

Due to the same reason, I do not suggest a company be involved on developing a new open core for sale, since it will not earn enough money back. However this is not absolute, I also listed some of good effects that selling open cores could bring.

Regarding to the question that should a company utilize existed open cores into next commercial products or not, my answer is that, there is no definite answer. This is the responsibility of the smartest project managers, because using open cores do have advantages but also take risks. So it is a practical question that depends on the different situations.

We also talked about the future of open cores. I guess open cores will keep growing yet not too fast, because there is currently no strong power (big companies, huge investment) that pays enough attention on pushing open cores.

Perhaps the only definite conclusion I am sure in this chapter is that everyone should keep an eye on open cores, because the next person who benefits from the cores maybe you.

Reference:

- [1] Michael Keating and Pierre Bricaud. *Reuse Methodology Manual for System-on-a-Chip Designs*. Kluwer Academic, 3rd edition edition, Jun. 2002.
- [2] R. K. Gupta and Y. Zorian. Introducing core-based system design. *IEEE Design & Test of Computers*, 14(4):15–25, Oct.–Dec. 1997.
- [3] Webpage, *Semiconductor Intellectual Property Core*, from Wikipedia, http://en.wikipedia.org/wiki/IP_core, Last visit: 2008.10.21.
- [4] Website, Open Source Initiative (OSI), <http://www.opensource.org/>, Last visit: 2008.10.21, OSI is a non-profit corporation formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open-source community.

-
- [5] Webpage, *The Open Source Definition*, from OSI,
<http://www.opensource.org/docs/osd>, Last visit: 2008.10.21.
 - [6] Webpage, *Open Source Licenses*, from OSI,
<http://www.opensource.org/licenses>, Last visit: 2008.10.21,
The licenses approved by the OSI comply with the open source definition.
 - [7] Website, OPENCORES.ORG,
<http://www.opencores.org/>, Last visit: 2008.10.21,
OpenCores is a loose collection of people who are interested in developing hardware, with a similar ethos to the free software movement.
 - [8] Webpage, *GNU*, from Wikipedia,
<http://en.wikipedia.org/wiki/GNU>, Last visit: 2008.10.21.
 - [9] Webpage, *GNU Project*, from Wikipedia,
http://en.wikipedia.org/wiki/GNU_Project, Last visit: 2008.10.21.
 - [10] Webpage, *Richard Stallman*, from Wikipedia,
http://en.wikipedia.org/wiki/Richard_Stallman,
Last visit: 2008.10.21.
 - [11] Webpage, *Free Software Foundation*, from Wikipedia,
http://en.wikipedia.org/wiki/Free_Software_Foundation,
Last visit: 2008.10.21.
 - [12] Webpage, *Free Software*, from Wikipedia,
http://en.wikipedia.org/wiki/Free_software,
Last visit: 2008.10.21.
 - [13] Webpage, *Free Software Movement*, from Wikipedia,
http://en.wikipedia.org/wiki/Free_software_movement,
Last visit: 2008.10.21.
 - [14] Website, GNU Operating System,
<http://www.gnu.org/>, Last visit: 2008.10.21.
 - [15] Website, Free Software Foundation (FSF),
<http://www.fsf.org/>, Last visit: 2008.10.21.
 - [16] Webpage, *Why "Open Source" Misses the Point of Free Software*,
by Richard Stallman, from GNU, [http://www.gnu.org/philosophy/
open-source-misses-the-point.html](http://www.gnu.org/philosophy/open-source-misses-the-point.html), Last visit: 2008.10.21.
 - [17] Webpage, *The Free Software Definition*, from GNU,
<http://www.gnu.org/philosophy/free-sw.html>,
Last visit: 2008.10.21.

-
- [18] Webpage, *BSD Licenses*, from Wikipedia,
http://en.wikipedia.org/wiki/BSD_licence, Last visit: 2008.10.21.
- [19] Webpage, *The BSD License Problem*, from GNU,
<http://www.gnu.org/philosophy/bsd.html>, Last visit: 2008.10.21.
- [20] Webpage, *The Modified BSD License—An Overview*, from OSS Watch,
<http://www.oss-watch.ac.uk/resources/modbsd.xml>,
Last visit: 2008.10.21.
- [21] Webpage, *The BSD License*, from OSI,
<http://www.opensource.org/licenses/bsd-license.php>,
Last visit: 2008.10.21.
- [22] Webpage, *GNU General Public License*, from GNU,
<http://www.gnu.org/copyleft/gpl.html>, Last visit: 2008.10.21.
- [23] Greg R. Vetter. “infectious” open source software: Spreading incentives or promoting resistance? *Rutgers Law Journal*, 36:53–162, 2004. SSRN:
<http://ssrn.com/abstract=585922>, Last visit: 2008.10.21.
- [24] Webpage, *Microsoft’s Ballmer: “Linux is a cancer”*, Jun. 1st, 2001,
from Linux, <http://www.linux.org/news/2001/06/01/0003.html>,
Last visit: 2008.10.21.
- [25] Webpage, *Why You Shouldn’t Use the Lesser GPL for Your Next Library*, from GNU,
<http://www.gnu.org/licenses/why-not-lgpl.html>,
Last visit: 2008.10.21.
- [26] Website, Gaisler Research,
<http://www.gaisler.com/>, Last visit: 2008.10.21,
Gailer Research is a privately owned company that provides IP cores and supporting development tools for embedded processors based on the SPARC architecture.
- [27] Webpage, *Selling Free Software*, from GNU,
<http://www.gnu.org/philosophy/selling.html>,
Last visit: 2008.10.21.
- [28] Webpage, *The GNU Manifesto*, from GNU,
<http://www.gnu.org/gnu/manifesto.html>, Last visit: 2008.10.21,
This is an old and good article suggested to read, which well describes the GNU’s philosophy. An interesting point is in the section “Won’t programmers starve?”, by saying “just not paid as much as now” it reflects that the GNU do realize that the free software movement will lower down the money people could earn.