A review of electronic engineering design free software tools

C. Medrano, I. Plaza
EduQTech, EUPT
Teruel, Spain
{ctmedra, iplaza}@unizar.es

M. Castro, F. García-Sevilla, J.D. Martínez-Calero
UNED - Madrid, Spain
mcastro@ieec.uned.es,
Francisco.Garcia@uclm.es,
josedanielmartinez@gmail.com

Josep Pou Felix, M. Corbalán
EduQTech, EUETIT - UPC
Terrassa, Spain
{pou@eel.,
montserrat.corbalan@}upc.edu

Abstract— **In this paper, we review electronic design free software tools. We have searched open source programs that help with several tasks of the electronic design flow: analog and digital simulation, schematic capture, printed circuit board design and hardware description language compilation and simulation. Using some rapid criteria for verifying their availability, we have selected some of them which are worth working with. This work intends to perform a deeper analysis of free software tools and select some of them to use in education or in professional electronic design.**

Keywords- electronic design; free software;quality; evaluation.

I. Introduction

Free software is an increasing phenomenon that has attracted a lot of attention in academia, industry and among end-users. Following the definition of the Free Software Foundation (FSF) [1], free software concerns the users' freedom to run, copy, distribute, study, change and improve the software. Beyond its philosophical principles, the way many free software projects are developed, often against traditional software engineering practices, is also a research topic [2].

The interest in free software would not have increased without the success of many projects that are widely used and accepted [3]. GNULinux and the Apache web server are probably the most well known, but there are many others: user applications (e.g., GIMP, OpenOffice), programming languages (e.g., Perl, Python), internet resources (e.g., sendmail, bind), adaptations to embedded systems (e.g., OpenMoko, Nokia 770). However, not all free software programs have reached a high quality level and open source repositories [4] show many abandoned projects.

Regarding electronic design, the following question arises: Which is the state of free software tools for electronic design? Are they useful? The long term goal of the present work is to answer these questions, evaluating the situation of these tools and selecting some of them for education or professional use. In this paper, we present a first review of such tools and perform a preliminary selection. We also point out the main weakness detected.

II. OUTLINE

The steps we have followed are outlined in Fig. 1:

- Search for free software tools: we have searched free software tools that help in typical tasks of the electronic design flow.
- Rapid test and program selection: we have performed a rapid evaluation of the programs found. This has allowed us to select some of them for a further research and deeper evaluation.
- Analysis of rejections and summary of selected programs.

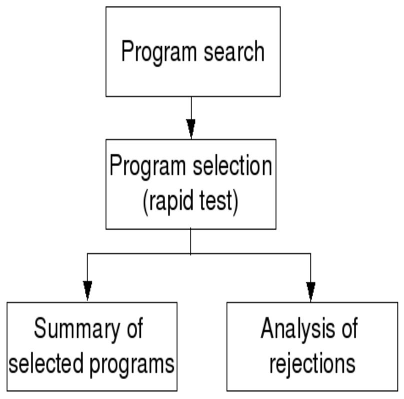


Figure 1. Main steps of the present work

III. SEARCH FOR FREE SOFTWARE TOOLS

A. Topics selected

We have searched for tools that are related to the following tasks:

- Analog simulation (AS): schematic capture with netlist generation, simulation engines and data visualization.
- Digital simulation (DS).
- Schematic capture (SC).
- Printed Circuit Board (PCB) Design.

- Hardware Description Language (HDL) design: compilation, simulation and waveform viewers.

B. *Sources of information*

We have searched in FSF [1], in classical repositories [4] and through the Internet using the Google Web search engine. Examples of keywords used are Electronic Design Automation (EDA), electrical, electronic, schematic capture, simulation, HDL, PCB, etc.

C. *Programs found*

In table I we have listed the number of programs found and the category they are included in. In some cases, they are divided into subcategories. In the analog simulation class, the sum of the programs in the subcategories does not equal the total in the category because there are integrated environments that belong to several subcategories at the same time.

TABLE I. NUMBER OF PROGRAMS FOUND AND THEIR CATEGORY

Category	Number	Subcategory	Number
Analog simulation (AS)	24	Simulation engine	9
		Schematic capture + netlist generation	10
		Data visualization	10
Digital simulation (DS)	13		
Schematic capture (SC)	22		
PCB design (PCB)	12		
HDL design (HDL)	28	Simulation and compilation	25
		Waveform viewer	3
Total	99		

IV. RAPID TEST OF PROGRAMS

Evaluation of software quality is still a controversial issue. There has been an evolution of this concept and the definition depends also on the perspective (architecture, source code, processes or community). ISO/IEC 9126 [5] offers a standard with six quality characteristic divided into several subcharacteristics. This quality model has to be adapted to every program. Other models focus on the free software world, analyzing their typical aspects: Open Business Readiness Ration, QSoS [6] and QualOSS [7]. However, applying those models to all the programs we have found is out of the scope of the present paper. We first have to pick up some of the software we have found. Thus, we have adopted a practical point of view and adopt the role of an electronic designer who wants to select a set of programs for further research. We have then specified five items in order to perform a rapid test.

- Is there a stable version?

A stable version would give confidence to software users. However, this question was not easy to answer. Traditionally release version 1.0 is the first one delivered to customers and considered stable. However, many free software projects follow the “release early, release often” rule [2], which means that the programs are available to the public well before they are considered stable. The Sourceforge repository includes information about the status of the software (alpha, beta or stable). But other downloadable programs have not this information. Therefore, this item was of limited use.

- Can it be easily installed?

If available, Windows versions usually pose no problem. For Linux users, probably the most interested in this kind of software, package management systems are the preferred way to install software because they solve all the dependencies. We have used the “aptitude” utility in the Kubuntu 8.01 distribution [8]. If no package is available, we have tried to compile the program from the source code and tried to solve the problems that appeared.

- Is the learnability good?

This is probably one of the most important points to attract new users. We have looked for user’s manuals, “getting started” guides, wikis or web pages, examples etc. They should at least allow the user to start working with the tool.

- Can it work with a simple example?

We tried simple examples in order to check that the program is not only installed, but that can run simple simulations, draw simple schematics, etc. This is similar to the idea of the “hello world” program that can be found in many introductory tutorials of programming languages.

- Is it likely to be maintained in the future?

Any potential user should be interested in the project future and the probability that it will be debugged, improved, updated or extended. To answer this question, we paid attention to the frequency of new versions, the number of developers, the number of messages in mailing lists or the news that appeared in the corresponding sections.

The goal of this five item test is to determine whether or not it is worth evaluating the program in depth.

V. ANALYSIS OF REJECTIONS

In this section, we analyze the reasons that led us to reject many of the projects found. Table II summarizes the results. The terms used in the table together with further comments are explained in the next paragraphs.

TABLE II. MAIN REASONS TO REJECT PROGRAMS

Problem	AS	DS	SC	PCB	HDL	Total
Abandoned	8	4	13	6	8	39
Does not install	1	0	1	0	3	5
Poor learnability	3	3	2	0	1	9
Failure to work with simple examples	2	0	1	0	1	4
Other	3	3	1	3	7	17
Total	17	10	18	9	20	74

In the group of abandoned projects we have included projects that have had no activity (news, releases) for more than two years. In this group there are also projects that classify themselves as in 'planning' state. However, this state coincides with a lack of activity, at least in the cases we found. As can be deduced from the table, there is a high number of abandoned projects, a 53 % of the total of rejections. This is clearly a waste of effort. New developers should adhere one of the lesson of [2]: *"Good programmers know what to write. Great ones know what to rewrite (or reuse)"*.

With respect to the installation, many of the programs are available as Kubuntu packages, what makes the installation very easy. For the rest, source compilation is possible if the libraries it depends on are installed previously. We have been able to compile most programs successfully. A few of them showed compilation errors, such as undefined symbols, which prevented us from getting an executable.

There are not many programs that fail when dealing with simple examples like an RC circuit, HDL code with basic logic, etc. At least, the projects that are not abandoned and that can be installed are able to perform some operations.

Poor learnability is associated with a lack of documentation. Unfortunately, there are nine projects that are rejected mainly due to the poor documentation. In many cases they are hosted by attractive web pages, they are updated regularly and they can be installed without problems. But the section documentation is still under construction, or shows a similar message, so the newcomer hits the wall when trying to build up his own examples. From the user point of view, this is really disappointing. Electronic designers are not always programming experts and looking into the source code can be very hard.

The row entitled "Other" in table II accounts for other reasons to reject a program that are not in the previous categories. In some cases the programs are very basic, e.g. only a few gates in digital simulation, or they are oriented towards kinds of design not considered here, e.g. full custom design. There are also many examples of exotic approaches. For instance, programs that help to create modules for other tools, to debug or to write code. Integration of Spice-like simulation or hardware description with general purposes languages (Tcl, Ruby, Python) is a key point of several projects. Although they are remarkable efforts, we have stuck to the typical approaches to the electronic design topics considered in this paper.

We should mention that we have rejected two popular programs. Both of them are active and have been developed for many years. The first one is *Xcircuit* [9]. We could not make the examples work due to a segmentation fault. However this can be due to a packaging problem with Kubuntu and other Linux distributions should be tested. We could not compile the source code successfully either. The second one is *Alliance* [10], a set of CAD tools for VLSI design that includes a VHDL compiler and simulator. There is no package available in Kubuntu, and we also failed to compile the source code. This should be further researched.

VI. SELECTED PROGRAMS

The number of programs that we have selected is shown in table III .

TABLE III. SUMMARY OF NUMBER OF PROGRAMS SELECTED

Category	Programs selected	Total	%
Analog simulation	7	24	29
Digital simulation	3	13	23
Schematic capture	4	22	18
PCB design	3	12	25
HDL	8	28	29
Total	25	99	25

A. *Analog simulation*

We have found two motor simulations, *ngspice* [11] and *gnucap* [12]. They are both command line programs. *Ngspice* is based on the well-known Spice3 simulator originated at the University of California. Since 1999, its code has been bug-fixed, cleaned, ported to the GNU/Linux platform and improved with models and simulation options. *Gnucap* is the Gnu Circuit Analysis Package. It is not based on Spice and the engine is designed to do true mixed-mode simulation.

Gwave [13] and *Kjwaves* [14] are tools to view analog data from the output of simulations. *Gwave* is part of the gEDA toolkit. *Kjwaves* is written in Java and has a nice user interface.

Schematic programs that can output Spice netlists are also considered in this section. *Gschem* [13] is a graphical tool which is part of gEDA to facilitate the input of components. Once the schematic file and the required symbols are ready, they can be converted into netlists using the *gnetlist* [13] command. *Kicad* [15] is an open source program for the creation of electronic and schematic diagrams and printed circuit boards. It has an option for the creation of Spice netlists.

Finally, we have found an integrated environment which has its own flavour. It is called Quite Universal Circuit Simulator, *Qucs* [16]. It can simulate analog and digital circuits. It deviates from typical Spice-like simulators. The entry is graphical and the simulation itself can be seen as an embedded object in the main window, Fig. 2. The user documentation is very poor, but there are some specific tutorials that allow new users to run simple examples. Despite the documentation, we have kept this project as a valuable tool, due to its nice and intuitive graphical interface.

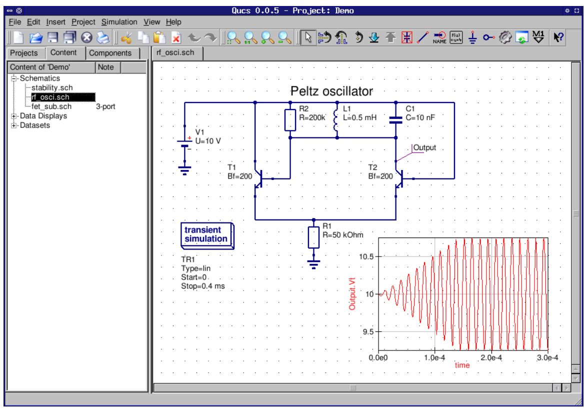


Figure 2. Screenshot of Qucs, taken from [16]

B. *Digital simulation*

Three projects gave us enough confidence in this section: *Tkgate* [17], *LogiSim* [18] and *Qucs*. *Tkgate* is an event driven digital circuit simulator with a tcl/tk-based graphical editor. It is a mature project that started at Carnegie Mellon University in 1987. *LogiSim* is written in Java and intends to be an educational tool for designing and simulating digital logic circuits. *Qucs* can not only perform analog simulation, as explained in the previous section, but also digital simulation. It is in fact based on the VHDL engine FreeHDL [19], but the user can simulate directly from a schematic entry without entering any VHDL code.

C. *Schematic capture*

Four programs have passed our rapid test. Three of them have already been reported in previous sections (*gschem, Kicad, Qucs*). The fourth one is *TinyCAD* [20]. It is a simple program that can print your diagrams or transform them into PNG images. It can be also used to output netlist for Spice or PCB programs. However, Spice output requires the introduction of many parameters by the user, which makes this process painful. In addition, only commercial PCB formats are possible as outputs, except for *FreePCB* (see next section).

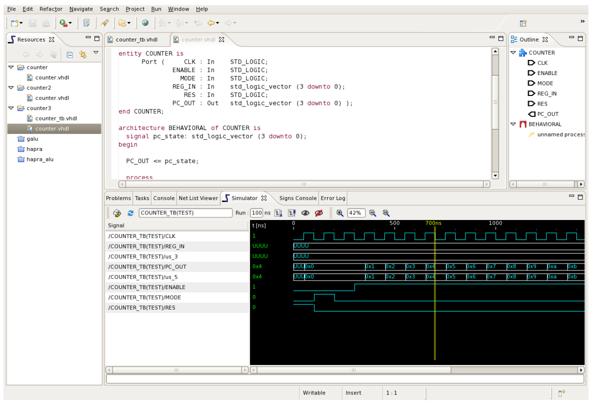
D. *PCB design*

This is the category in which we have found fewer programs and as a consequence, fewer programs passed our rapid tests. This is probably due to the higher difficulty of the user interaction programming. The three programs are *PCB* [13], *FreePCB* [21] and *Kicad* [15]. *PCB* is now associated with the *gEDA* tool suite. Therefore, a schematic entry with *gschem* can be the first step of a printed circuit board design. The schematic file can be converted to a netlist compatible with *PCB* using the command *gnetlist*. *PCB* is a twenty year old application, which has been enhanced with Gerber generation, autorouter and GTK port. *FreePCB* was designed to be easy to use and ease to learn. It has only Windows version. In principle, netlist has to be entered semi-manually with a text editor. Since 2008, a utility written by a user allows converting *TinyCad* netlist into PADS-PCB netlists suitable for *FreePCB*. *Kicad* also allows the user to design a printed circuit board, starting from a schematic capture. All the steps of the design flow are integrated into a single graphical interface, much as the manner in which some commercial programs do.

E. *HDL compilation, simulation and data visualization*

We have found five compiler and simulator tools for HDLs. *Icarus* [13] is a command line verilog compiler, which is now associated with gEDA. In fact, there are two main commands: *iverilog*, which compiles the code, and *vvp*, which is the simulation run time engine. The ouptut of the simulation is a VCD file, but other waveform dumps are available. Typically, the user has to include a test bench within its program. *Ghdl* [22] acts in a similar manner, but with VHDL code. It is a command line compiler that can analyze or execute a design (test benches) depending on the flags used. In the latter case, VCD file format output is possible. *Veriwell* [23] is another command line verilog compiler which outputs VCD files. While last version is recent, documentation is worse than that of *ghdl* or *icarus*. *Signs* [24] can be used either as a command line or as an Eclipse plugin [25] for hardware design and simulation. At the moment, VHDL is the main language. Eclipse defines itself as an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. It was hard to make it run the examples of the *Signs* tutorial, but this is probably due to our inexperience with the Eclipse environment. On the other hand, *Signs* has an integrated interface, Fig. 3, which is closer to commercial tools. The last compiler we have found is *freeHDL* [19]. Although the home web page has some characteristics of an abandoned project, such as empty pages and old documentation, it is the backend simulator used by *Qucs*, which is a quite active project. *Qucs* now accepts user written VHDL code in digital simulations. The results of a simulation are embedded as in Fig. 2.

Two waveform viewers are specific to digital simulation: *GTKwave* [26] and *Dinotrace* [27]. Both of them accept VCD and other file formats. *Dinotrace* comes with an interface to Emacs which allows source code and log files to be annotated with the values of signals.



```
entity COUNTER is
    Port (     CLK : In    STD_LOGIC;
            ENABLE : In    STD_LOGIC;
              MODE : In    STD_LOGIC;
            REG_IN : In    std_logic_vector (3 downto 0);
               RES : In    STD_LOGIC;
            PC_OUT : Out   std_logic_vector (3 downto 0) );
end COUNTER;

architecture BEHAVIORAL of COUNTER is
  signal pc_state: std_logic_vector (3 downto 0);
begin

  PC_OUT <= pc_state;

  process
```

```
Signal                      t [ns]  0        500   700ns         1000
/COUNTER_TB(TEST)/CLK       1
/COUNTER_TB(TEST)/REG_IN    UUUU    UUUU
/COUNTER_TB(TEST)/us_3      UUUU    UUUU
/COUNTER_TB(TEST)/PC_OUT    0x4     UUU 0x0  0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xa 0xb
/COUNTER_TB(TEST)/us_5      0x4     UUU 0x0  0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xa 0xb
/COUNTER_TB(TEST)/ENABLE    1
/COUNTER_TB(TEST)/MODE      0
/COUNTER_TB(TEST)/RES       0
```

Figure 3. Screenshot of *Signs*, taken from [24]

VII. CONCLUSIONS AND FUTURE WORK

We have searched for free software tools that help in some tasks of the electronic design flow. Then, we have performed a rapid check and test of the programs in order to select which ones are worth continuing working with. We have found that about 39% of the projects are abandoned. On the positive side, we have found that there are several programs that cover each of the tasks considered in this study: analog simulation, digital simulation, schematic capture, PCB design and HDL design. Thus, we think that efforts should focus on improving existing projects, rather than on creating new ones, unless new projects include original approaches.

Generally speaking, documentation is a weak point. We have rejected about 9% of the projects for this reason. Even within the projects we have selected, we feel that user information should be improved. Sometimes we made the examples work with a mixture of documentation and intuition. New users should have a clear view of what software can do and where to obtain more information. However, we acknowledge that documentation is slowly improving in the projects we know for some years.

Another issue is the fact that people are not used to work with the GNU/Linux operating system and they are unwilling to change their operating system. One quick way to sort this out is by using virtualization. It is possible to use programs like *VirtualBox* [28] and *VMWare* [29] that allow users to work with different operating systems at the same time. This is a great way to try and use these free software programs that otherwise people would refuse because of the nuisance of installing and configuring GNU/Linux and the programs. It is also a great way to create an environment with the operating system and the installed programs that users can carry with them in portable media like USB memories.

In future, we plan to evaluate the selected programs in depth. For that purpose, we will use a quality model. We intend to apply a model based on the ISO/IEC 9126 standard [5] and a model specific to free software programs [7]. We want to derive from this study which is the suitability of the programs for real-world designs and where the efforts should be focused in order to improve them.

REFERENCES

[1] Free Software Foundation, http://www.fsf.org/ , accessed on August 2009.

[2] Eric S. Raymond, The Cathedral and the Bazaar, http://www.catb.org/~esr/writings/cathedral-bazaar/, accesed on October 2009.

[3] K. Crowston, H. Annabi, and J. Howison, Defining open source software project success. In Proceedings of 24th International Conference on Information Systems, Seattle, WA, USA, December 2003.

[4] Software repositories, http://sourceforge.net/, http://freshmeat.net/, accesed on August 2009.

[5] International Standard ISO/IEC 9126-1, Software engineering-Product quality-Part1: Quality model.

[6] J.C. Deprez, S. Alexandre, Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS, Lecture Notes in Computer Science, Vol. 5089, pp 189-203, 2009.

[7] Quality in Open Source Software, http://www.qualoss.org/ , accessed on October 2009.

[8] Kubuntu Linux distribution, http://www.kubuntu.org/, accessed on October 2009.

[9] Xcircuit web page, http://opencircuitdesign.com/xcircuit/, accessed on October 2009.

[10] Alliance web page, http://www-asim.lip6.fr/recherche/alliance/, accessed on October 2009.

[11] Ngspice web page, http://ngspice.sourceforge.net/history.html, accessed on October 2009.

[12] Gnucap web page, http://www.gnu.org/software/gnucap/, accessed on October 2009.

[13] gEDA web page, http://www.gpleda.org/index.html, accessed on October 2009.

[14] Kjwaves web page, http://sourceforge.net/projects/kjwaves/, accessed on October 2009.

[15] Kicad web page, http://www.lis.inpg.fr/realise_au_lis/kicad/, accessed on October 2009.

[16] Qucs web page, http://qucs.sourceforge.net/index.html, accessed on October 2009.

[17] Tkgate web page, http://www.tkgate.org/ , accessed on October 2009.

[18] Logisim web page, http://ozark.hendrix.edu/~burch/logisim/ , accessed on October 2009.

[19] FreeHDL web page, http://freehdl.seul.org/ , accessed on October 2009.

[20] TinyCAD web page, http://tinycad.sourceforge.net/, accessed on October 2009.

[21] FreePCB web page, http://www.freepcb.com/, accessed on October 2009.

[22] GHDL web page, http://ghdl.free.fr/ , accessed on October 2009.

[23] VeriWell web page, http://sourceforge.net/projects/veriwell/, accessed on October 2009.

[24] Signs web page, http://www.iti.uni-stuttgart.de/~bartscgr/signs/, accessed on October 2009.

[25] Eclipse web page, http://www.eclipse.org/org/, accessed on October 2009.

[26] GTKWave web page, http://gtkwave.sourceforge.net/, accessed on October 2009.

[27] Dinotrace, http://www.veripool.org/wiki/dinotrace, accessed on October 2009.

[28] VirtualBox, http://www.virtualbox.org/, accessed on October 2009.

[29] VMWare, http://www.vmware.com, accessed on October 2009.