Chapter 4

The Open Source Design Tools

4.1 Chip Design Flow

As long as the technology and processes involved in circuit design remain within the current boundaries, the design variables will remain the same: Circuit area, execution speed, and power consumption. If a new and completely different technology and materials arrive, this could change. Meantime, the design teams will dependently develop their areas, iteratively delivering new versions of them until the team achieves a final functional design within the time-to-Market frame.

The trade-offs. The design team keeps in mind that when you modify the area of a circuit it also impacts the power consumption and the execution time. Although it is not possible to gain in those three variables: you gain in one, then loose in the other two. That is why in these projects the design team needs to focus on the final goal, rather than its own specifications, and still, optimize the individual parameters to the maximum achievable.

Time to market. The circuit design field is a fast moving one, with new circuits being released every day. Professional design teams always work on a very tight time schedule, in order to complete their design, containing what will be a novelty in the market, only if it arrives on time to it. At this point is important to note that maximum optimization always goes as far as time schedule allows, it means that you will see that if you would have more time to work on your design it can be smaller, or faster, or may consume less energy. But there is no more time for that, or the market will no longer find your design useful. So you need to declare your design done, in order to get into the market on time. Of course this applies when you are designing for a market, but if you design as a hobby or as part of an open cores community, you can have more time.

4.2 Open Source Design Tools

Using open cores in designing independent integrated circuits is a growing trend between electronic engineers, and there are large communities focusing on open source development, intended for electronic hardware. Designing hardware cores by programming them starting from scratch, is not easy, you need to know about electronics, programming, and design tools. For instance, many steps need to be done to ensure a design can be synthesized and translated to an FPGA or a Silicon wafer, through several verification steps. Knowledge on FPGAs and standard libraries is needed, and you need to be good at HDL programming.

4.3 Open Source EDA Tools

There are plenty of good EDA tools that are available as open source. The use of such tools makes it easier for you to take advantage of the resources and open cores available in related sites and forums. The larger and most used site for this purpose is Opencores.org. You can access there IP cores and scripts for an open source HDL simulator.

Here is a description of the most used terms and tools you will need to know. Of course, as any practical tool, there is no other way to be a master than using it and practicing.

Icarus Verilog Simulator. Icarus Verilog is a Verilog simulation and synthesis tool. It works like a compiler: when you compile source code written in Verilog, you can deliver different formats. For synthesis (the process of generating a circuit design from a description language), the compiler generates net lists. These,

and other compilers, elaborate design descriptions according to IEEE standards. You can surf the internet to download the Icarus Verilog simulator.

Verilator. Verilator is a free Verilog HDL simulator. It compiles synthesizable Verilog into an executable format and wraps it into a SystemC model. Keep in mind that the resulting circuit after compilation greatly depends on how you programmed it, so, the execution speed of the resulting model can widely vary. Since Verilator has been used to simulate many very large multi-million gate designs with thousands of independent modules, it is often chose as part of several verification environments. You can also surf the web to find the site for Verilator.

GUI-based design tools. For those not used to code by lines, there are GUI tools (Graphic User Interface). Of course, the more easy and graphic the tool is, the less control you have on the final representation of your design. A sample of a GIU design tool is Fizzim, but there are several more. The advantages of using a GUI tool are that they run in Windows or Apple, or anything with Java.

4.4 Open Cores Library

There are several internet sites where you can find circuit cores developed by experienced designers. One of the most popular sites is www.OpenCores.org, where you can find from the simple circuits, as adders and multipliers, to complex designs as processors and memories. Even more, you can find complete systems of hardware IP cores that you can download and use as open source. You can find them in several Hardware description languages, such as VHDL, Verilog, Verilog, SystemC, Bluespec, and C/C++. The developing stage or status of each core is indicated, so you know how trust worthy or reliable a core is; the stages or

versions of these projects can be Planning, Mature, Alpha, Beta and Stable. Within the open source community, there are different licenses that apply to each product; the licenses you will find are GPL, LGPL, BSD, among others.

A list of example core and projects is presented here, but it is very dynamic so you can check recent cores on the website. The cores are grouped by purpose, for instance: Arithmetic, Processors, Memories, Systems-on-a-Chip, and so on.

a) Arithmetic cores:

- Anti- Logarithm (square-root), base-2, single-cycle
- Discrete Cosine Transform core
- Elliptic Curve Group
- Floating Point Adder and Multiplier
- Gaussian Noise Generator
- Random number generator
- Maximum/Minimum binary tree finder
- Signed integer divider
- Sine and Cosine Table
- Trigonometric functions (degrees) in double fpu

b) Processors:

- ARM-compatible cores
- R6502 Processor
- Educational 16-bit MIPS Processor
- FORTH processor with Java compiler

- HIVE- 32 bit, 8 thread, 4 register
- Leros: A Tiny Microcontroller for FPGAs
- 8051 compatible CPU
- MCPU- A minimal CPU
- Wishbone High Performance Z80
- ZPU- the world's smallest 32 bit CPU with GCC toolchain

c) Memory cores:

- 8/16/32 bit SDRAM Controller
- Functional simulation models for commercially available RAMs
- High Performance Dynamic Memory Controller
- High Speed SDRAM Controller with Adaptive Bank Management and Command Pipeline
- Parametrized FIFO based on SRL16E
- Single Port ASRAM
- Synchronous reset fifo memory
- Wishbone Flash Interface for Parallel FLASH

d) Communication controllers:

- 10, 100, 1000 Mbps Ethernet MAC
- 8b10b Encoder/Decoder
- A VHDL CAN Protocol Controller
- Ethernet MAC 10/100 Mbps

- Ethernet Switch on Configurable Logic
- USB Device Core
- Wishbone SD Card Controller
- Space Wire Light
- Serial Port Interface Flash controller

e) System-on-Chip:

- Embedded FPGA Core
- Arm core
- RFID Transponder
- I2C Controller
- Real-time image processing unit
- OR1200 SoC
- Opens ARC-based SoC
- Soft Multiprocessor on FPGA
- MP3 Decoder
- NoC Network on chip

f) Other cores:

- 16x2 LCD controller
- 8254 Timer
- Alternative Oscilloscope
- Adjustable Frequency Divider

- Keypad scanner
- DDS Signal generator
- Date time
- FM Receiver
- General purpose IO
- Sound Encoder
- PWM controller
- Multiple Switch Debouncer
- OpenRisc 1200 Graphic Configuration Tool
- Programmable Interval Timer

Licenses. Although the open source code is free software, there are differences among the different license agreements that you accept when downloading and using it. Here is a brief explanation of several licenses, and you should check extensively the kind of license you are agreeing to. According to the Open Source Initiative, an Open source license "shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale. The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is

not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed".

LGPL. The GNU Lesser General Public License (LGPL) is a free software license published by the Free Software Foundation (FSF) that allows developers and companies to use and integrate LGPL software into their own software without being required by the terms of a strong license to release the source code of their own software-parts. For proprietary software, LGPL-parts are in the form of a shared library so that there is a clear separation between the proprietary and LGPL parts. The LGPL is primarily used for software libraries.

GPL. The GNU General Public License is a free license mostly used for software and it is intended to guarantee your freedom to share and change all versions of a program, to make sure it remains free software for all its users. So, you need to check carefully the version you are using, since it can come from a developer that has made a lot of changes to the original version.

BSD. The BSD license is a simple and liberal license for software. The restrictions to users are that if they redistribute such software in any form, with or without modifying it, they must include the original copyright notice, a list of restrictions, and a disclaimer of liability. The restrictions are: one should not claim that they wrote the software and should not sue the developer if the software does not function as expected.