MaSoCist Evaluation primer

Martin Strubel

November 20, 2016

Revision:

1 Overview

The MaSoCist distribution enables you to quickly design, maintain, document and automatically create a family of Soft core featured System on Chip solutions on various FPGA architectures. It is relying heavily on the Linux kernel config utility and the section5 device description XML language.

The full MaSoCist development environment is supported for Unix operating systems only

This evaluation version is featured by the simple System On Chip design 'agathe'. It implements a microcontroller with the following peripherals:

- 2x16 GPIO
- IRQ controller
- SPI interface
- Up to 8 PWMs/Timers
- UART

See separate SoC documentation (`soc-agathe.pdf`) for details.
The MaSoCist environment is available as:

- Virtual machine (VirtualBox VM)
- LXC linux container
- Tar file for 64 bit x86 architecture (*)

1.1 Evaluation version specifics

The evaluation/opensource version lacks a few options. Therefore, a few configuration options can not be altered or have no effect. This behaviour is currently undocumented. The source is provided 'AS IS'.
Restrictions are present on:

1. SoC design functionality
2. Co-Simulation features, virtual hardware implementations
3. Configureable, pipelined CPU cores
4. Debugging (ICE) functionality

Module	Description
ghdlex, netpp	free Co-Simulation extensions (partially OpenSource)
gensoc	commercial SoC generator utility
stdtap/uniemu	JTAG debugger (GDB debug agent)

Table 1.1: Optional (proprietary) modules

1.2 Licensing

The licensing of the MaSoCist distribution depends on its distribution's package tag, e.g. for a tar file: `masocist-$(DIST_TAG)-$(VERSION).tgz`. By default, `DIST_TAG=''opensource''`.
When it comes to hardware designs, licenses such as the GPL (Gnu Public License) are hardly applicaple, in particular because it would complicate development under a dual licensing scheme. Since, by experience, complicated license schemes are seldom respected by hardware developers, an attempt is made to phrase a few rules as follows.

NONCOMMERCIAL USAGE

- You can use it for educational purposes or non-commercial home projects
- You will not get much free support, but you may of course feed back bugs
- You are encouraged to publish your changes, but noone will force you to it. Just respect the principle of fair use.

COMMERCIAL USAGE

- If you use the code in a commercial environment, you are free to do so, but you are **required** to publish changes made to the MaSoCist code base, including your own code that depends on MaSoCist functionality.
- If you are making a product that you are re-licensing or re-selling to others, you will need to acquire a license for the IP you use

The opensource variant also has the `CONFIG_OPENSOURCE` variable defined. Code under an exclusive OpenSource license will then throw a warning on compilation into a simulation. This is used to mark source released under the implicated OpenSource agreement. Removal of the CONFIG_OPENSOURCE checks are considered a violation of this OpenSource license.
If you wish to have a maintained custom package, you will have to sign up for a custom license agreement. This entitles you for a distribution tag. In this case, you are completely free to keep further development proprietary, EXCEPT changes made to the ghdlex package.
In any case, contributions made by third parties under an open source license agreement will always remain in the open source.
Note that you also have to follow the GHDL license agreements for distribution of simulation executables.

1.3 Target audience

The MaSoCist is made for hardware/software developers who mainly wish to play with configureable IP cores and OpenSource HDL CPU cores. The focus lies mostly in rapid prototyping and automatization. It is assumed, that:

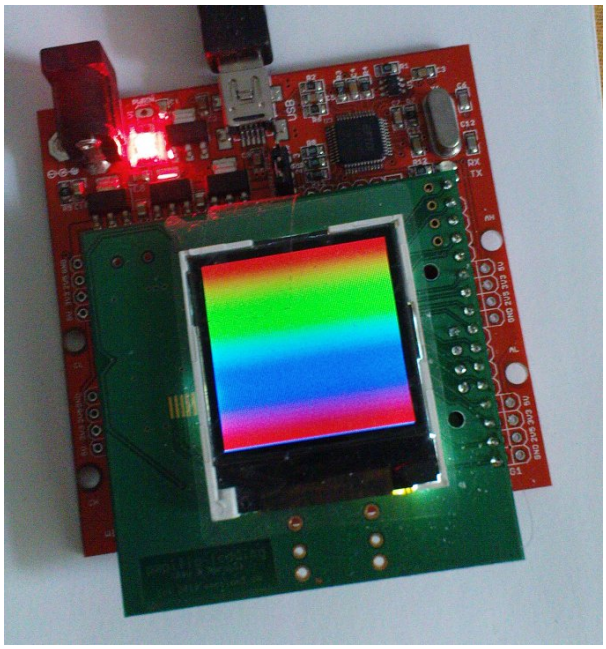

Figure 1.1: Papilio with TFT wing

- The developer is experienced with linux command line tools
- Independence of vendor specific tools is a strong desire
- No support is needed, and the "read the source, luke" principle is acceptable.

1.4 Board supply packages

Currently, two popular third party evaluation platforms are actively supported by the MaSoCist environment. They can be configured and programmed "as is", i.e. not needing any external adapters.

1.4.1 Papilio One

The Papilio platform is a Xilinx Spartan3 based development platform with a USB JTAG solution on board. This board has gained some popularity similar to the Arduino boards. It can be extended using pluggable **Wings**, developed and supported mostly by a large independent developer community. Fig. 1.1 shows a Papilio with a TFT wing.
The Papilio can be purchased from various online shops, see http://papilio.cc for more information.

1.4.2 MachXO2 Breakout board

The MACHXO2 breakout board (Fig. 1.2) is a Lattice MACHXO2 based development platform, also featured by an on-board JTAG controller.
The MachXO2 breakout board is available at various distributors such as Mouser, Digikey, etc.

Make sure to get a recent variant of this board, populated with a MachXO2 7000 FPGA. Older boards with the 1200 variant do not have sufficient resources.

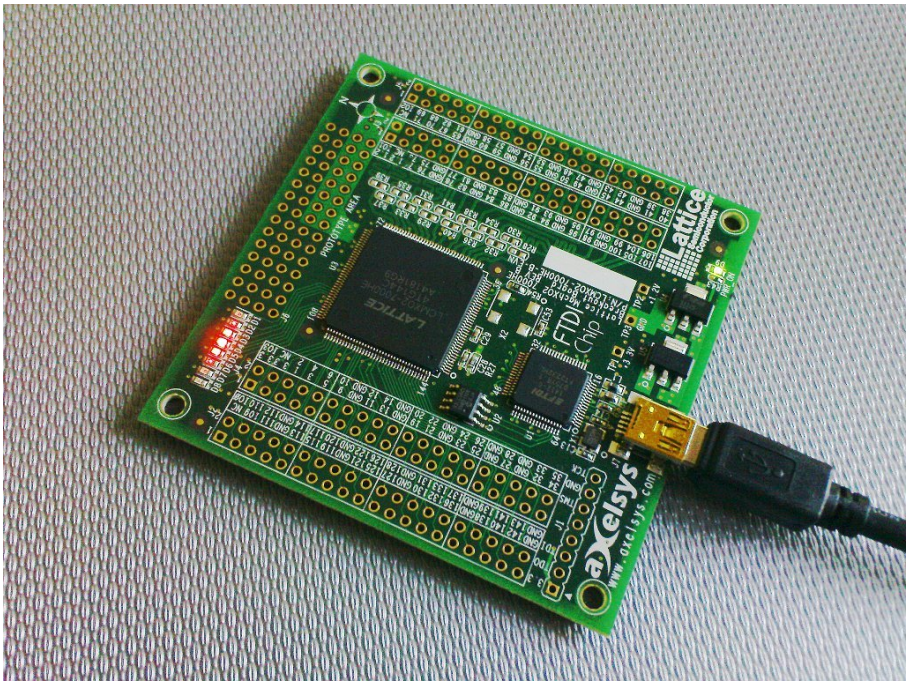

Figure 1.2: MACHXO2 breakout board

2

Quickstart

2.1 Prerequisites

You need a development environment. If you have signed up for a board supply package agreement, you may have received a virtual machine image (Linux container or VirtualBox image file). In this case you do not need to worry about the prerequisites, because everything has been installed "ready to go". If you have obtained the tar file distribution, the whole package may not be complete. In this case you have to make sure that the following tools are installed on your system in order to run the simulation or generate code. The tools are typically found as a package for Debian and other systems with their corresponding name.

1. Linux kernel config 'kconfig'
2. GNU full native toolchain with make, gcc, cpp, etc.
3. Target toolchain, zpu-elf-gcc, mips-elf-gcc, etc.
4. python 2.x, IntelHex module
5. xsltproc

To compile and run the simulation, the following packages are required:

1. GHDL v0.30 or greater, GHDLex v0.051 or greater
2. netpp (/usr/lib/libslave.so)
3. gtkwave

2.2 Quick board selection

The supported board supply packages, 'out of the box':

- Papilio One platform
- MachXO2-7000 Breakout board

To configure for one of these boards, change your working directory to the MaSoCist top level dir, run
make papilio_config
or
make breakout_config
Then you are ready to rebuild the source code and HDL files.
For available custom or third party configurations, see vendor/$(VENDOR)/defconfig_* or check your custom SoC documentation.

2.3 Simulation

The entire SoC is simulated by GHDL with a few extensions to support virtual interfaces. Not all boards can be simulated by default, see below.
To build the simulation, GHDL and the GHDLex library must be present. Then you execute
make sim
in the MaSoCist directory. Possibly you will have to call **make -C sim clean** after a reconfiguration.

2.3.1 Running the simulation

Once the simulation was built correctly, the resulting tb_<platform> will be created in the `sim/` directory. Some configurations are based on a virtual UART instance, see `sim/virtualuart.vhdl`. This is started using the init-pty.sh script. Then you can connect to the running simulation using a terminal program, such as minicom:

```
minicom -o -D /tmp/virtualcom
```

When running inside the linux container (LXC), no manual setup using init-pty.sh is required. The virtual COM port is set up by the LXC.

Note that this UART is a full UART emulation, so the baud rate defined by the divider value passed to virtualuart.vhdl must match the UART peripherals baud rate, unlike the virtual console (`CONFIG_VIRTUAL_CONSOLE`), which does not depend on a baud rate.
Finally, the simulation is run by starting the tb_<platform> executable from the command line. For interactive waveform tracing using GTKwave, there is a script `run.sh`, taking the platform name as argument, e.g.

```
./run.sh papilio
```

2.3.2 Advanced interactive simulation

Enter the sim/ directory and type "make run". You should see the GTKwave window popping up, showing a slowly progressing wave display due to the enabled throttle. To disable the throttle, run the following command inside the virtual machine (or on your host)
netpp localhost TapThrottle 0
If you run netpp from your host to access the Virtual Machine or LXC (Linux Container), you need to determine the IP address of its virtual bridge interface (or what you have configured your VM with). For example, **ifconfig** on Linux may display:

```
vboxnet0  Link encap:Ethernet  HWaddr 0a:00:27:00:00:00
          inet addr:192.168.56.1  Bcast:192.168.56.255  Mask:255.255.255.0
```

Then you simply access the simulation via
netpp 192.168.56.1 TapThrottle 0

This option only applies when the simulation is configured with a virtual TAP, like `CONFIG_VTAP`

2.3.3 Vendor specific simulation issues

Simulation of some boards may require libraries that are not included in the MaSoCist, because they are vendor specific. There may be several solutions:

- Obtain necessary files from your local FPGA tool installation and create a GHDL library. Use the -P option to GHDL to specify the search path to the GHDL config file.
- Try the `CONFIG_EMULATE_PLATFORM_IP` option
- Use a virtual board config that is fully vendor IP independent

For example, when you receive an error message like this:
`../hdl/plat/breakout_top.vhdl:16:9:  cannot find resource library ''machxo2''`
You have to create a file lattice/machxo2-obj93.cf somewhere using the rule:

```
MACHXO2_VHDL = $(wildcard $(LATTICE_SIM)/machxo2/src/*.vhd)
lattice/machxo2-obj93.cf: $(MACHXO2_VHDL)
    [ -e lattice ] || mkdir lattice
    ghdl -i --workdir=lattice --work=machxo2 $(MACHXO2_VHDL)
```

where LATTICE_SIM is the directory of your simulation VHDL files, like
`/usr/local/diamond/3.1_x64/cae_library/simulation/vhdl/`
Then set the LIBGHDL variable in `vendor/default/local_config.mk` to the directory where you created `lattice/machxo2-obj93.cf`, like:

```
LIBGHDL = $(HOME)/src/vhdl/lib/ghdl
```

For full 'model in the loop' simulation, it will be necessary to acquire an additional, non-free Simulation package by section5

2.4 Prepare synthesis

If you have configured something using the menuconfig utility, run either "make syn" from the top level or "make" inside the `syn/` directory to update your files. Then run your synthesis tool and open up the corresponding project file in syn/<FPGA_VENDOR>/<PLATFORM>, for example:

breakout
syn/lattice/breakout/breakout-opensource.ldf

papilio
syn/xilinx/papilio/zpu/zpu-opensource.xise

If the project files are not present for your platform, see below on how to quickly import the files required for synthesis.

2.4.1 Porting to new platform

When synthesizing for a new FPGA platform, you will have to create a new project first. The MaSoCist environment helps you with importing all the necessary files by creating TCL scripts for the supported synthesis tools (Xilinx ISE 13.4, Lattice Diamond 3.2 at this time).

1. Create the new project in syn/<fpga_vendor>/<platform_name>/<project_name>
2. To import the project files, open a TCL shell inside your IDE and run source ../../proj_<platform>.tcl
3. Run Synthesis to check if all files referenced are imported

FPGA platform specific IP cores may have to be manually imported into the project

2.5 Configure and build

For usage of the kconfig environment, see SoC specific documentation.
Basically, you execute the kconfig menu by running
make menuconfig
from the top level MaSoCist directory.
Before synthesis or simulation, it is required to rebuild the corresponding system files:
make syn

In the evaluation version (no gensoc included), the hardware peripherals can not be configured. You can only choose among supplied board configurations in `CONFIG_SOCDESC`.

2.6 Target download

2.6.1 Papilio

First, you have to set the `XILINX_ISE_DIR` variable in order to assemble the full firmware image for the SPI flash.

- Define XILINX_ISE_DIR in vendor/opensource/local_config.mk
- Export XILINX_ISE_DIR in .bashrc or alternative shell startup file

Example:

```
XILINX_ISE_DIR = /media/sandbox/Xilinx/13.4/ISE_DS
```

The Xilinx tools allow to recompile the software without the full ROM synthetization. See `syn/xilinx/papilio/Makefile` rules:

$(PLATFORM)_fw.bit:
Build rule to merge program data with existing firmware bit file

download:
Downloads bare SRAM image into target

flash:
Merges extended memory (cacheable) space into overlay SPI flash image and programs it into the target

For download to the target, the **papilio-prog** application is required. See Papilio homepage for details.
The full firmware download is run by **make flash**. If no SPI code cache and program overlay is used, you can run **make download** instead. Otherwise, you have to make sure that all addresses are in sync.

2.6.2 MachXO2 Breakout

The MachXO2 variant requires a full synthetization of the generated HDL when the boot rom was altered. For download to the target, please use the Lattice Diamond Programmer.
Project files for target download exist:

breakout.xcf
Download into SRAM (volatile)

flash.xcf
Flash permanently onto the target

2.6.3 Other boards

All other boards such as proprietary / custom development can be programmed using an ICEbearPlus adapter, if the JTAG pins are accessible. Some boards may have embedded USB JTAG controllers. See Table 2.1 for details.

Board name	Programming	JTAG adapter
HDR60	Diamond Programmer	Integrated USB JTAG
gözcü	xc3sprog	ICEbearPlus
EFM01	Proprietary Cesys tool	ICEbearPlus (debug only)
Digilent Spartan-3 Board	xc3sprog	ICEbearPlus
denver2/3	xc3sprog	Embedded ICEbear

Table 2.1: Programming method

The Impact Cableserver for ICEbear is no longer supported.

3

Troubleshooting

Typical errors that may appear on your console:

/usr/bin/ld: cannot find -lslave
: You need to install a netpp slave package or the full netpp source and set the NETPP environment variable to it

virtualuart.vhdl:42:24:@0ms:(assertion failure): Failed to open PTY pipe
: The virtual UART is not running. Start sim/init-pty.sh first, see Section 2.3.1.

../hdl/plat/breakout_top.vhdl:16:9: cannot find resource library "machxo2"
: Vendor specific library not installed, see Section 2.3.3.