GreenSocs Purveyor of fine Open Source Virtual Platform Technology and Services
Since 2005

GreenSocs Virtual Platforms let you imagine, design, develop and test your embedded application as a whole; size your hardware to reflect the needs of your software; build your software and hardware together; debug your software and verify your hardware efficiently.

Feb 2016
Mark Burton
GreenSocs® is the industrial leader in integrating different Virtual Platform solutions

Dr Mark Burton is the founder of GreenSocs. Mark has worked for ARM managing their modeling group. He was the chair of the OSCI TLM WG and the OCP-IP SLD WG.

GreenSocs provided technology behind the TLM-2.0 standard, and the CCI standard. We continue to be at the heart of SystemC development.

GreenSocs is a contributor to QEMU, providing technology to support multi-thread and reverse execution.

GreenSocs has been in business (incorporated in UK and France), and profitable, since 2005, including within its client base large multi-nationals.
Tool Integration

Enabling System Level Design

- g5
  - gem5
  - QEMU
  - Core IP models

- GreenSocs
  - SystemC modeling and integration libraries

- IP Generation Tools
  - e.g. Verilog to SystemC
  - VeriPool

- eclipse
  - Debug Environments
    - Including eclipse.

- Profiling and Code Coverage
  - kcachegrind
  - lcov
Tool Integration

Enabling System Level Design

- **IP Generation Tools**
  - e.g. Verilog to SystemC
  - Verilator

- **Debug Environments**
  - Including eclipse.

- **Profiling and Code Coverage**
  - kcache
  - gcov

- **The Digital ‘plugin’ to COSEDA**

- **GreenSocs**

- **QEMU**
  - Core IP models

- **veriPool**
  - IP Generation Tools
  - e.g. Verilog to SystemC
  - Verilator

- **acellera**

- **eclipse**
  - SystemC modeling and integration libraries
COSIDE: Cortex M3 Toplevel

Enabling System Level Design
- COSIDE provides symbols for the GreenSOCs library models
- Allows graphical system assembly
- Automated toplevel netlist generation
COSIDE: Interface modules

- Own Interface module to communicate between your normal COSIDE module and the GreenSocs TLM eco system

![Diagram of OWN_MOD](image)

- Corresponding Wrapper of the whole Cortex M3 processor exposing the interface Signals

![Diagram of i_toplevel?](image)

- System configuration via LUA file provide through toplevel parameter
COSIDE: Hard- and Software debugging

- Hardware brake points within the model as well as within the ARM
- Software Debugging through remote session into the Cortex M3 core
QBox

• Wraps up Qemu in a TLM2-0 API such that it can be used in standard SystemC

• QEMU is a generic and open source virtualizer – it covers almost all CPU architectures and achieves extremely high performance.
Qemu: Our Preferred source of CPU models

- Qemu is the defacto standard Virtualizer.
- Free and Open Source.
- It is over 10 years old

<table>
<thead>
<tr>
<th>Architectures</th>
<th>CPU’s</th>
<th>Commits</th>
<th>Contributors</th>
<th>Lines of code</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>1100</td>
<td>43000</td>
<td>1000</td>
<td>989,863</td>
</tr>
</tbody>
</table>

- GreenSocs is a key contributor:
  Reverse execution and Multi-Core TCG Kernel.

- Regular committers from many organizations
### Existing Model database overview:

### CPU Family coverage:

<table>
<thead>
<tr>
<th></th>
<th>X86</th>
<th>ARM</th>
<th>MIPS</th>
<th>Alpha</th>
<th>PowerPC</th>
<th>SPARC</th>
<th>Microblaze</th>
<th>Coldfire</th>
<th>Cris</th>
<th>SH4</th>
<th>Xtensa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast SW dev model (LT)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Cycle Accurate HW dev model (AT)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

Full list (of several hundred) available on GreenSocs.com
Extending Qemu for EDA virtualization

- MULTI Thread Qemu
  - A massive speed improvement for Qemu to take advantage of multi-core hosts

- SystemC integration
  - The ability to mix SystemC models with Qemu.

- Reverse Execution
  - The ability to find a bug, and step backwards (in time) to find the source of the bug.
  - GreenSocs has a fast implementation which is compatible with SystemC.

- Instruction counting and analysis features
  - Enhanced counting mechanisms for memory accesses and instruction counting
Quick Benchmark Results for MTTCG

Enabling System Level Design
SystemC TLM-2.0 IEEE 1666 is:
The Virtual Platform Standard

• Open Source Simulator available for download from Accellera.org

• GreenSocs technology at the heart of TLM-2.0 standard.
• All GreenSocs interfaces use TLM-2.0

• GreenSocs helping Accellera forge a new Model to tool standard.
  • Preview available in GreenConfig.

• Our solutions are tool independent, and work with all vendors.
Model Based Virtual Platform Architecture

Virtual Platform Model

Component Library

Server Library

Or

Client Library

Interoperability Layer

Open source simulator, or SystemC standard Vendor tool

Enabling System Level Design
GreenSocs SystemC Infrastructure.

- Model Construction
  - Eases building register definitions, state machines etc
  - Scripting (Python)

- Model to Model communication
  - Busses and routers (e.g. AMBA, PCIe, OCP, etc)
  - Signals (interrupts etc)
  - Serial, Ethernet, Graphics etc....

- Model to Tool communication
  - Configuration, (inc Lua)
  - Control (Run time re-configuration)
  - Inspection (outputs and tracing).

- Model IP
  - Routers,
  - simple IP blocks,
  - libraries (Graphics, communication)

- Tools
  - Integration with Qemu, GEM5, Eclipse and other tools.
So, what does the code look like?

PL011::PL011(sc_module_name name):
gr_device(name, gs::reg::INDEXED_ADDRESS, 0x1000, NULL),
target_port("target_port", r),
irq_socket("irq_socket"),
serial_sock("serial_socket"),
irqNumber("irq_number", 0)
{
    serial_sock.register_b_transport(this, &PL011::serial_b_transport);
    gs::socket::config<gs_generic_signal::gs_generic_signal_protocol_types> cnf;
    cnf.use_mandatory_extension<IRQ_LINE_EXTENSION>();
    irq_socket.set_config(cnf);
    target_port.disable_delay();

    this->readPtr = 0;
    this->readWPtr = 0;
    this->readCnt = 0;

    pl011_r("UARTDR", "Data register", PL011_UARTDR, 0x0000, 0x0FFF);
    pl011_r("UARTRSR", "Receive status register", PL011_UARTRSR, 0x0000, 0x000F);
    pl011_r("UARTFR", "Flag", PL011_UARTFR, 0x90, 0);
    pl011_r("UARTILPR", "IrDA low power counter register", PL011_UARTILPR, 0x0000, 0x000F);
void PL011::end_of_elaboration()
{
    GR_FUNCTION_PARAMS(PL011, clearIRQ);
    GR_SENSITIVE(r[PL011_UARTICR].add_rule(gs::reg::POST_WRITE, "clearIRQ",
                                      gs::reg::NOTIFY));

... void PL011::clearIRQ(gs::reg::transaction_type *&tr, const sc_core::sc_time &delay)
{
    r[PL011_UARTRIS] = r[PL011_UARTRIS] & ~r[PL011_UARTICR];
    updateIRQ();
}

void PL011::updateIRQ(void)
{
    IRQ_ext_data data;
    gs_generic_signal::gs_generic_signal_payload payload;

    /* Update the IRQ line. */
    data.value = (r[PL011_UARTRIS] != 0);
    data.irq_line = irqNumber;
    payload.set_data_ptr((unsigned char *)&data);
    irq_socket.validate_extension<IRQ_LINE_EXTENSION>(payload);
    irq_socket->b_transport(payload, time);
}
GreenSocs provides its library, and IP under an end user license agreement to its customers.

You receive full source, and the rights to modify and distribute that source code internally to your company free of charge.

The license is a modified BSD.
Licensing Model for QBox

Qbox is provided as an TLM model from http://greensocs.com/

It is a free download, and neither you, nor any customer is required to pay any license.

It is licensed under the GPL. You may use it freely, if you distribute code based on it, you must do so in source form.

You should write your SystemC models to respect the TLM-2.0 API’s. You should allow your customers the choice of which CPU model to use. One (free) choice is a Qbox model.

Nobody (apart from GreenSocs) should be distributing Qbox.
Importance of Standards

Your product

Her tool choice

Is not his choice

Must not force your choice!

Is their component

Models must be tool independent
They must use the Standard interfaces
Consultancy and Services

The **Experts** in **Virtual platforms**: Creation, deployment, integration

Tool independent – vendor neutral.
Allow us to guide you to **success**

MODEL DEVELOPMENT

Virtual Platforms based integrated development environments, for CoTs or specialist devices, ready for your software engineers to be productive.
All models adhere to **STANDARDS**
**All model source provided.**

OPEN SOURCE DEVELOPMENT

Adding to the existing open source tools and models.
‘Upstreaming’ and dissemination