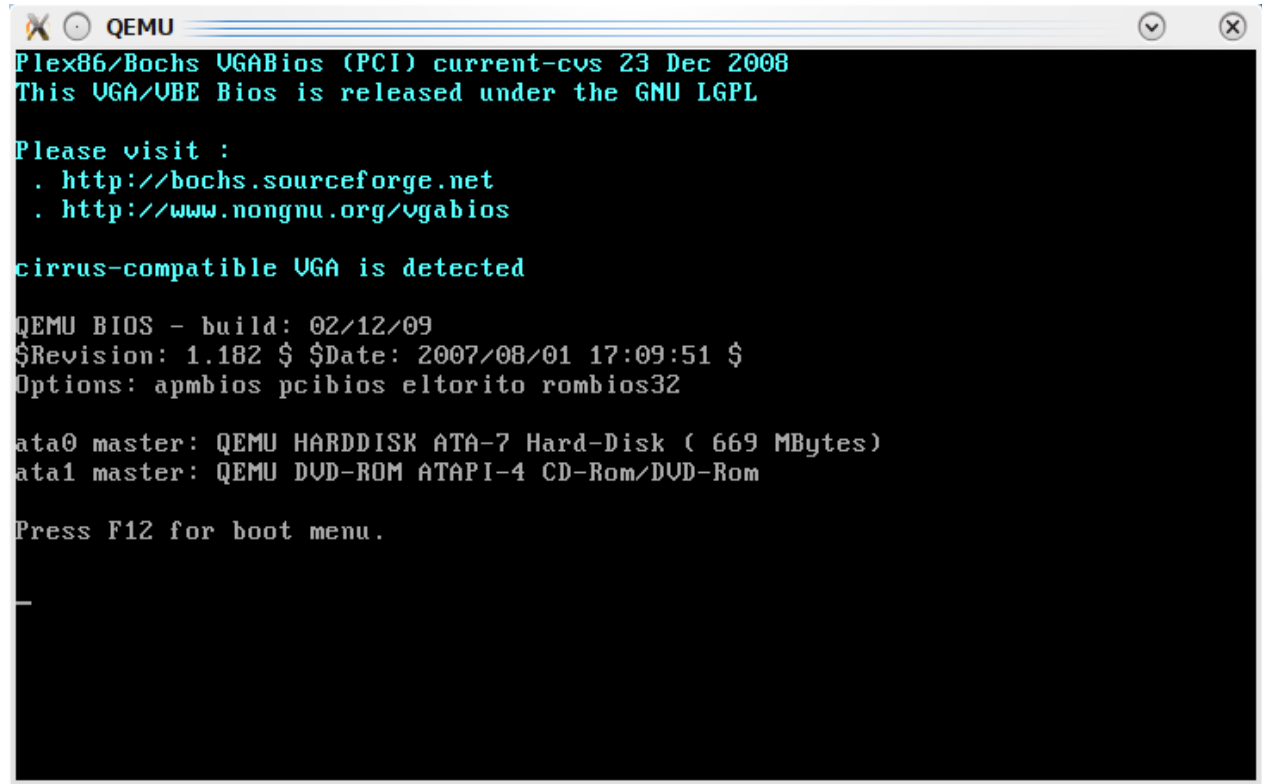


Mixed simulation kernels for high performance virtual platforms

M. Montón, J. Carrabina, M. Burton

- QEMU
- SystemC Bridge
- Wrapped QEMU
- Experiments
- Results
- Conclusions

- Generic machine emulator
- Very good performance
- Virtualization (KVM)
- C based code
- Open source
- Emulates
 - x86
 - ARM
 - MIPS
 - SPARC
 - ...



```
QEMU
Plex86/Bochs VGABios (PCI) current-cvs 23 Dec 2008
This UGA/UBE Bios is released under the GNU LGPL

Please visit :
. http://bochs.sourceforge.net
. http://www.nongnu.org/vgabios

cirrus-compatible UGA is detected

QEMU BIOS - build: 02/12/09
$Revision: 1.182 $ $Date: 2007/08/01 17:09:51 $
Options: apmbios pcibios eltorito rombios32

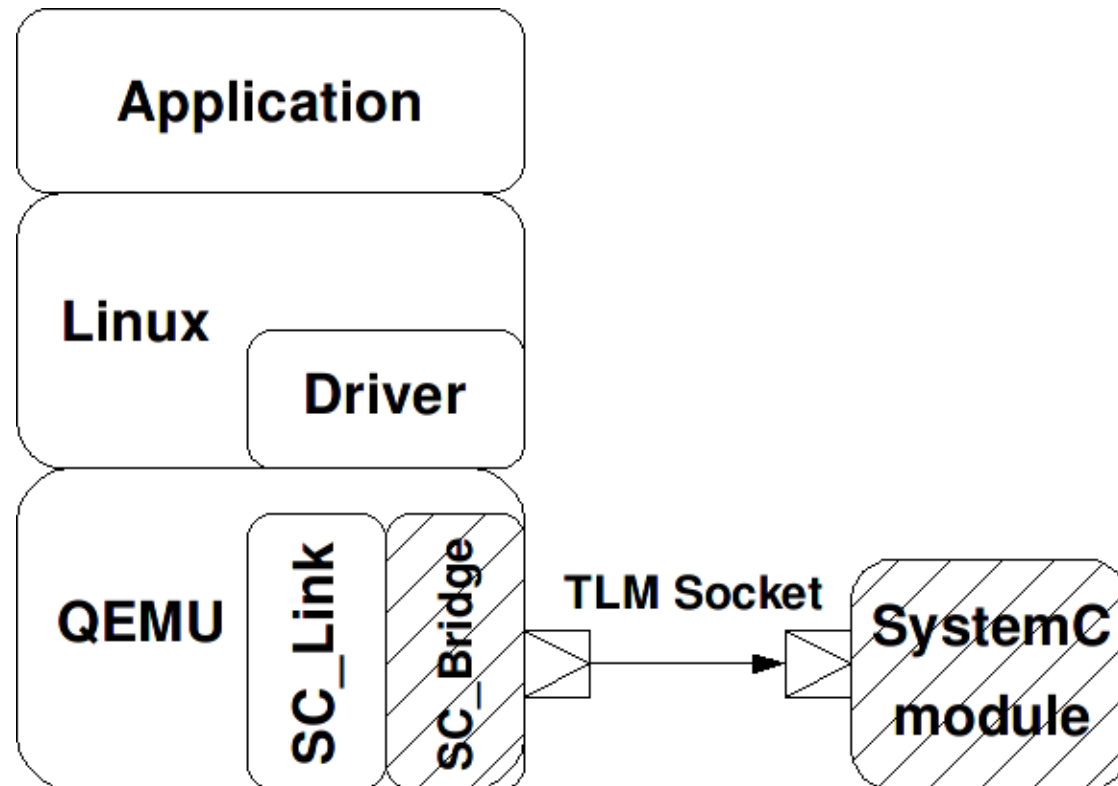
ata0 master: QEMU HARDDISK ATA-7 Hard-Disk ( 669 MBytes)
ata1 master: QEMU DVD-ROM ATAPI-4 CD-Rom/DVD-Rom

Press F12 for boot menu.
```

- System peripherals written in C:
 - UART, VGA, Ethernet, ...

- Why not use that simulator with SystemC devices?
 - Complete SoC Simulation
 - Debug New devices
 - O.S. Drivers

- QEMU Controls the SystemC simulation
- Ghost device (SC_Link) acts as TLM-2.0 Initiator
- Ghost device appears as a standard virtual platform device
- SystemC device is a TLM-2.0 Target



Synchronization

- Two simulators (QEMU & SystemC)
- How to synchronize them?
 - Every clock cycle -> Performance!
 - Majority simulation time is in QEMU, not in SystemC
 - Every transaction -> Enough?
 - Pending events in SystemC?
 - External events?

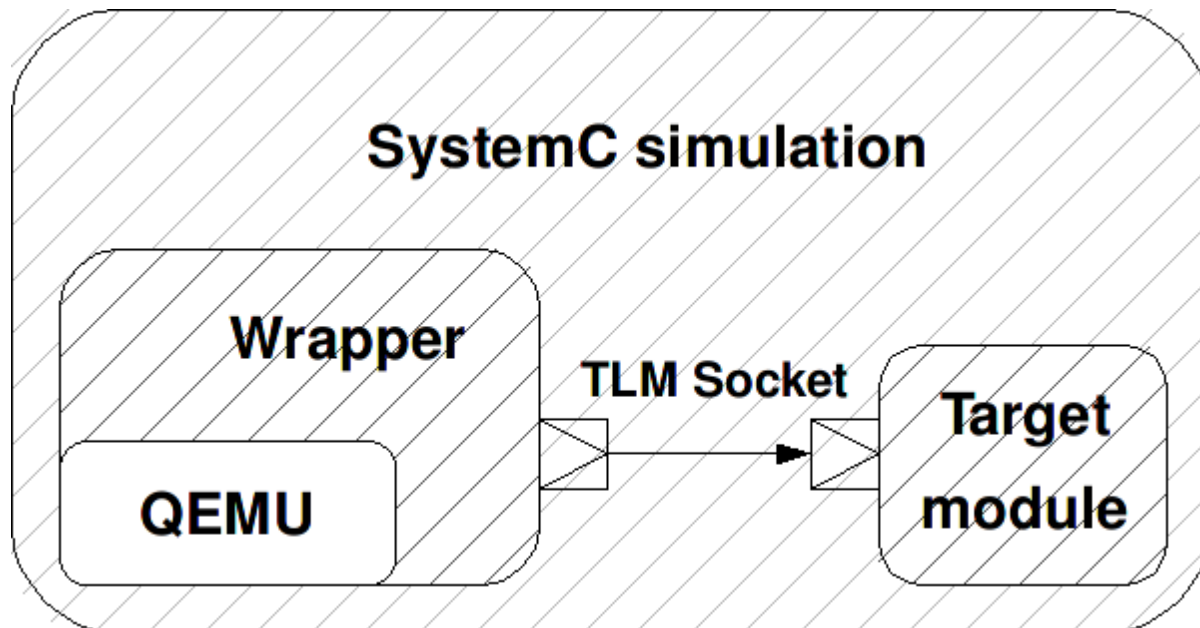
Lazy Synchronization

- SystemC models are TLM-2.0 Target
 - Access to register file can trigger internal event
 - I/O of device through the bridge
-
1. run() SystemC to equal time to QEMU time.
 2. start TLM-2.0 transaction.
 3. check when will occur a event in SystemC (if any).
 4. post a callback in QEMU in that future time to synchronize again.

Pros & Cons

- **Fast**: Only synchronizes when really needed
- SystemC devices can use **QEMU helper functions** for I/O
- **No changes** to QEMU code
- Same strategy **portable** to other simulators (Simics)
- **Ad-hoc** management of SystemC kernel
- Possibly **overhead** of task switching between QEMU and SystemC kernel
- Any change to SystemC devices needs a **full QEMU compilation**

- Better having a TLM-2.0 Initiator for QEMU
- Device connection using TLM-2.0 sockets
- Standard SystemC simulation setup
- Wrap QEMU into a SystemC Initiator

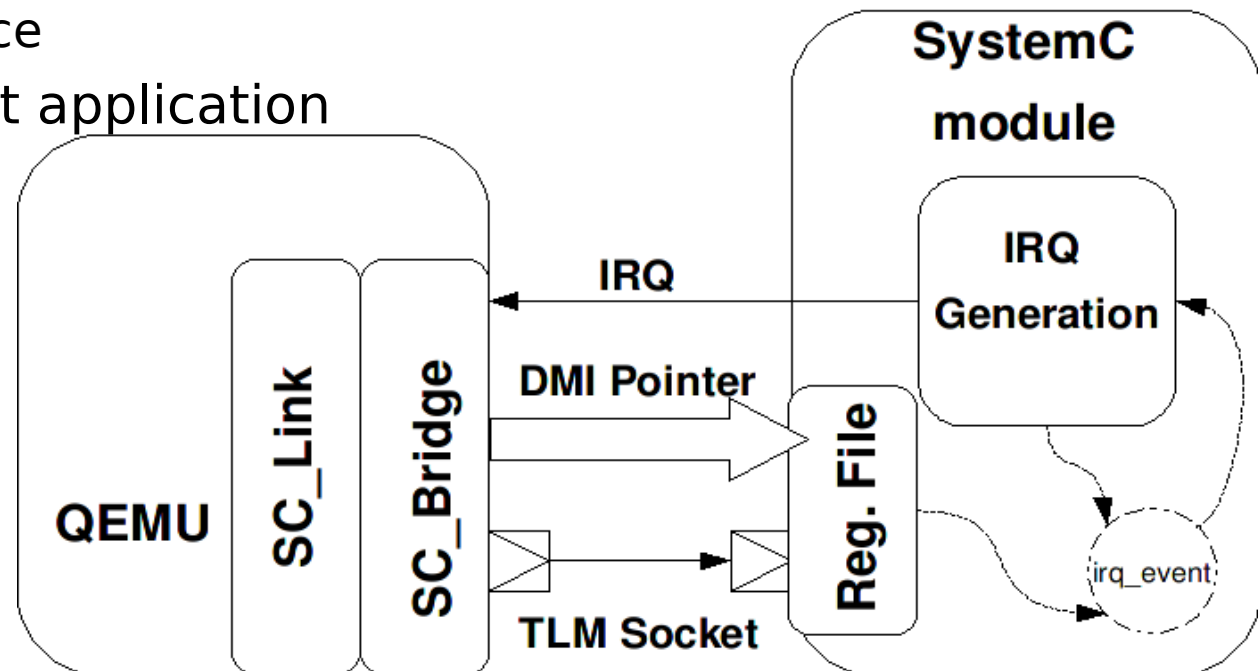


- QEMU main is a end-less loop
- We need to break that loop to call wait()
- We only call wait() if QEMU time and SystemC time are different.
- Each access to our SystemC device is transformed to a TLM-2.0 transaction.
- Device I/O is managed by device itself

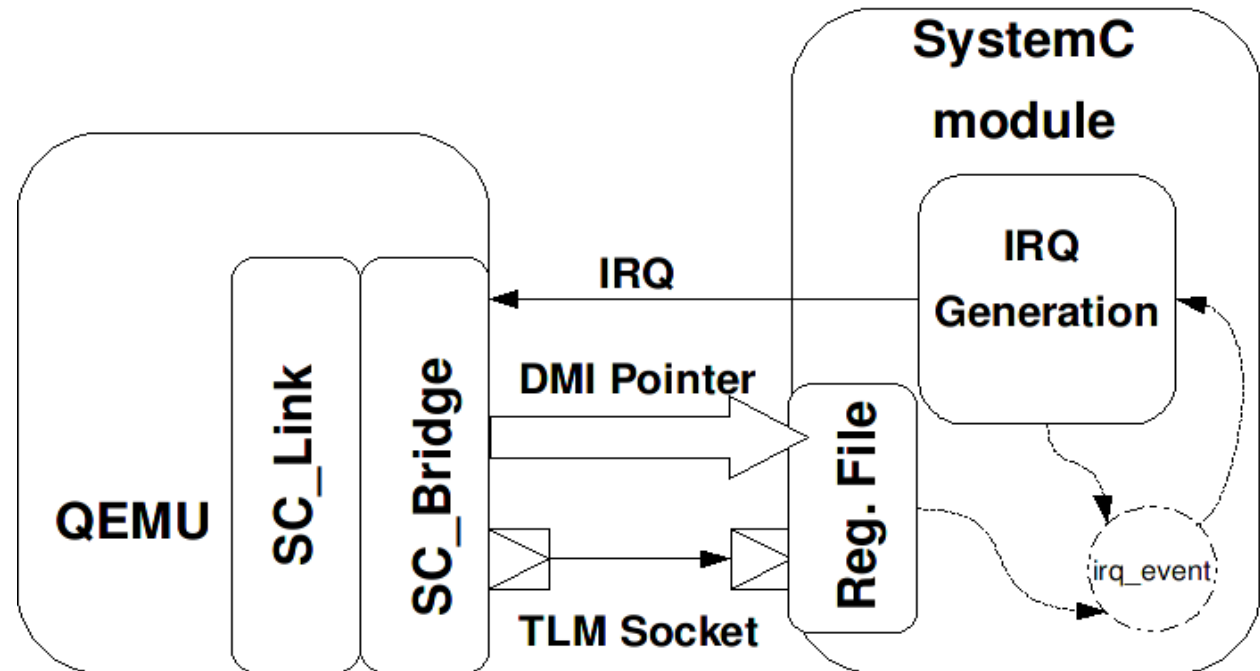
Pros & Cons

- **Fast:** Only synchronizes when really needed
- QEMU is a **TLM-2.0 Initiator**
- QEMU Initiator is a set of **libraries and header.**
- **Standard** SystemC simulation
- Possibly **overhead** of task switching between QEMU and SystemC kernel
- Devices can't easily **use QEMU helper functions**

- We "plug" a simple device to a x86 platform and ARM platform
 - Register file (128 Registers)
 - Event posting for IRQ generation
 - DMI to Register file
- GNU/Linux driver for the device
 - `/dev/sc_device`
- User space test application



- Testing
 - two solutions for synchronization (Wrapper and Bridge)
 - two different platforms (Intel and ARM)
 - two different buses (PCI and AMBA)
- No different performance observed



Simics

- Same strategy used with QEMU Bridge
- Simics **no source code is available**
- Added a new device managing SystemC kernel
- Set-up
 - PowerPC basic platform
 - SystemC 16550 UART connected to console
- **Penalty less than 5%** on performance between Simics UART and SystemC UART.
- **Dynamically loadable** into Simics
 - Recompile Bridge only when SystemC device changes

- Added SystemC capabilities to QEMU
- Methodology to join SystemC kernel to virtual platform simulators
 - QEMU
 - Simics
- TLM-2.0 based
 - Platform independent (x86, ARM, PCI, AMBA, ...)
 - No performance penalty (< 5%)
- We have a full virtual platform TLM-2.0 SystemC
 - Debug & Test new devices
 - Complete Virtual Platform in SystemC (Moblin)



All code available at <http://www.greensocs.com>

Thank you!