Industrial use of open-source IP cores

Jiri Gaisler

Gaisler Research

jiri@gaisler.com

Presentation overview

- Status of open-source IP development
  - Who develops? Why?
  - Quality and usefullness
  - Percieved benefits for industry
- Some experience of using open-source IP
- Experience of developing open-source IP
- Dependability issues
- The way forward ?

Why so little open-source hardware models?

- Why is there so much O-S software, but so little hardware?
  - Tool costs
  - Boards cost
  - Complexity of the problem
  - Usefullness
  - Short exposure to the problem

Who designs open-source IP and why

- Students (universities)
  - Education and research
- Enthusiasts (opencores, fmf, free-ip)
  - Joy of creation
- Component manufacturers (memory, i/f models, fpga libs)
  - Promote component sales
- Government and non-profit organistaions
  - Standardizing libraries and interfaces
- A few companies (Flextronics, Gaisler, Sun ...)
  - Market penetration, marketing/advertising

Quality of open-source IP

- Education projects: varying, rarely completed
- Joy of creation: varying, but rarely complete/optimised
- To promote components sales : high, but often not complete
- Standardisation: high but focused on types and functions
- Market penetration: relatively high but uneven

The percieved usefullness of O-S IP for industry

- Avoiding high license costs
- No royalties
- Able to modify core at will
- Long-term supply and maintenance
- Portability
- Simplify prototyping
- Dependability?

Possible problems

- Functional bugs
- Incomplete documentation
- Incomplete functionality
- Low performance or high area
- No support for target technology
- No technical support or bug-fixing
- No long-term maintenance

Example of O-S IP resuse (1)

- Opencores ethernet IP core reuse in LEON processor
- WB/AHB bridge: 1 man-month (reused)
- Linux/RTEMS driver development: 2 man-months
- Operation at 10 Mbit OK, failed at 100 Mbit
- Updated to new release, still failure
- 1 man-month debugging
  - Reset operation wrongly documented
  - FIFO synchronisation requires 2x sysclk/txclk ratio
- 4 man-months total effort = $25K = commercial core cost
- Benefits: unlimited use of IP to the cost of a single instantiation

Example of O-S IP resuse (2)

- Opencores PCI bridge reuse in LEON processor
- WB/AHB bridge: 2 man-months
- Operation OK in simulation, failed on hardware
- Debugging: 1 man-months
  - FIFO synchronisation problem(PCI clock < CPU clock)
  - Parity generation problem
- Switched to new 'fixed' version: 1 man-month
  - Problems persisted
- Developed own PCI target: 1 man-month
- Results: 4 man-months waisted, need solved by simpler function

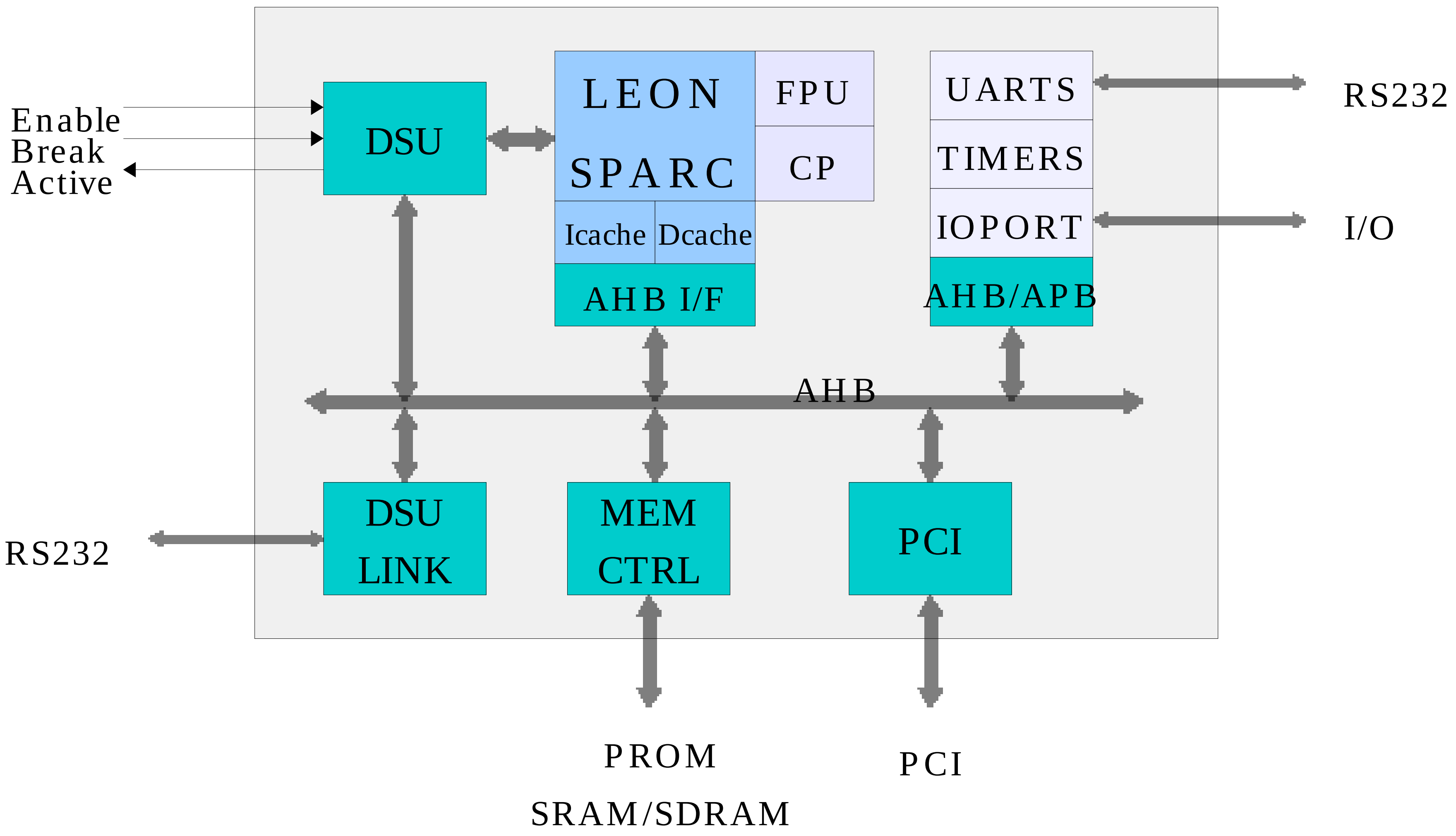
Can O-S IP be used in industrial projects?

- Simple answer: no
- Elaborated answer: no, not unless :
  - There is a support structure in place
  - The IP is written for reuse
  - The end-user has enough skills (to develop the IP himself)
  - The IP is well written and mostly bug-free
  - The IP complexity is not too high
- Note: this has nothing to do with the fact that the IP is open-source, rather with the way the IP is developed and maintained!

Example of O-S IP development

- LEON SPARC V8 fault-tolerant processor, funded by ESA
- Development effort: ~4 man-years
- Released as open-source for two reasons:
  - Improve test coverage
  - Promote SPARC architecture
- Both objectives reached:
  - Several bugs reported and corrected
  - Several ports of compilers/kernels/techs carried out
- Heavy use in research and education (+40 papers in 2002/3)
- 3 FT designs, +30 commercial design underway

LEON2 architecture



Why did LEON succeed (as O-S IP) ?

- Continuously funded, supported and maintained
- Written for portability (tools and technology)
- Highly configurable
- Modular
- Infrastructure of supporting tools (compilers, kernels, simulator)
- Implements a complex (=expensive) function not easily aquired
- Uses standards (SPARC, AMBA, PCI, ethernet)

The way forward ?

- To be successful, open-source development needs to adopt methodologies from the world of open-source software, by defining a common approach to:
  - Interfaces and functions (cores, buses, data types)
  - Portability
    - Hardware abstraction layer (HAL)
    - Tool independent coding style
  - Packaging and reuse
  - Version control and maintenance
  - Documentation

The Gaisler way forward

- The opencores attempt is promising, but lacks a common approach to most development issues
- LEON addresses many development issues, but is centered around one function (processor)
- Other IP libraries are either not open-source, or technology specific
- With experience gained from the LEON development and open-source core resue, Gaisler Research has started the development of a library of reusable, open-source IP cores: GRLIB

GRLIB overview

- Open-source IP library centered around AMBA bus
- Hardware abstraction layer for tech macros (mem, pad, mul)
- High-level, tool-independent defined coding style
- Weakly coupled modules
  - Self-contained, can depend on base package and other mods
  - Must contain scripts for config, simulation and synthesis
  - Can support optional auto-probing/configuration
  - Can contain s/w drivers and test benches
- Can be coupled to SOC builder

GRLIB LEON3 example

```vhdl
library ieee;
use ieee.std_logic_1164.all;
library gaisler;
use gaisler.libamba.all;
use gaisler.libmctrl.all;
use gaisler.libleon3.all;
use gaisler.libtech.all;

architecture rtl of demo3 is
constant FABTECH : integer := virtex2;
constant MEMTECH : integer := virtex2;
begin
...
  cpu0 : leon3          -- LEON3 processor
  generic map (fabtech => FABTECH, memtech => MEMTECH)
  port map (clkm, rstn, ahbmi(0), ahbmo(0), ahbsi(0), leon3i, leon3o);

  sd0 : sdctrl          -- SDRAM controller
  generic map (pwron => 1, ahbconf => AHBCONF, apbndx => 2)
  port map (rstn, clkm, ahbsi(1), ahbso(1), apbi(2), apbo(2), sdi, sdo);
...
```

GRLIB LEON3 example

```
Vsim> run -all

# Top level design: leon3_demo
#
# foundry technology: UMC 0.13
# memory  technology: Virage mempack 1.0
#
# leon3 processor: version 1.1
#    icache: 4*8 kbyte, LRU
#    dcache: 4*8 kbyte, LRU
#    mmu: V8 reference mmu, 16-entry shared TLB
#
# AHB arbiter/decoder version 1.0
# 2-bank SDRAM controller version 1.0
# APB bridge version 1.4
#
.
.
```

O-S IP and dependability

- Multiple users/tools/technologies accelerate bug fixing
- Full source code allows inspection for malicous code
- Allows independent evaluation and result sharing
- Faster turn-around cycle to include new research technologies

Conclusions

- At present, O-S IP resuse is difficult and require expert skills
- Improved methodology is required to improve IP quality and lower the 'reuse barrier'
- Gaisler research believes that a bus-centric and selft-contained IP distribution approach is most suitable
- Dependability can quickly be tested and improved
- Tool cost still a significant problem