When Innovation is Hard

Does the Open Source Development Model Work in Hardware?

ilkka.tuomi

meaningprocessing.com

Theses

- 1. Software and hardware are epistemologically different artifacts
  - This leads to different innovation and conflict models in software and hardware projects
  - The success of the open distributed approach in ”close-to-hardware” software development (e.g. Linux kernel) is partly explained by this epistemological difference
- 2. Close-to-hardware software, social software, and hardware are different
  - Close-to-hardware software development is easy to distribute because the developer community consists of a homogenous network of people and tools
  - There exists a ”phenomenological bottleneck:” Real hardware can only be approximated using digital representations; distributed hardware projects can therefore be really hard
  - There also exists a ”political bottleneck:” Social software can be hard as there are multiple interpretations and systems of meaning

Different Types of Animals Require Different Styles of Breeding

- Close-to-hardware SW
  - In a given HW environment:

SW is both the description and the implementation

Mapping between software and the functionality of the system is one-to-one

Technical functionality can be empirically tested

- HW
  - The context is the open world:

Unmodeled features matter

Systems wear, tear and break down

Although dominant voices are loudest, evaluation criteria have to be negotiated from multiple perspectives

- A homogenous developer community, rooted in its own ”objective reality.”

- A homogenous developer community, rooted in its own ”professional reality.”

  or:

- A network of interacting communities, each with their own stocks of knowledge.

The Epistemic Difference

- Close-to-hardware software
  - Approximates well the positivist and empirist assumptions

In the domain of SW, only those things exist that are explicitly described

Meta-level questions (e.g. progress) can be answered by empirical tests

- Hardware
  - Operates under the constraints of phenomenological ontology and epistemology
  - Approximates well the social constructivist assumptions

The Study Method

- Study open hardware development projects that are very similar to open source software development
  - but where the object of development is a physical product
  - Baseline: Linux kernel

Software that is closely integrated with the underlying processor hardware

  - Reference pools:

SourceForge.net software projects in the ”Linux,” ”Hardware,” and ”Finance” categories

  - Physical products: semiconductor cores at OpenCores.org

Semiconductor cores: pre-designed building blocks (processors, DSPs, bus-controllers, ...) that can be combined and re-used to build integrated circuits, including systems-on-chip.

The Case: OpenCores.org

- ”SourceForge” style portal that focuses on sharing logic designs (semiconductor ”intellectual property” cores / virtual components)
  - Developers can use their preferred license but are encouraged to use GNU or BSD -style licensing
  - OpenCores.org provides a CVS version control system that enables developer collaboration
- Founded in 1999 by Damjan Lampret, Slovenia
- 308 projects in CVS (Sept. 2008)

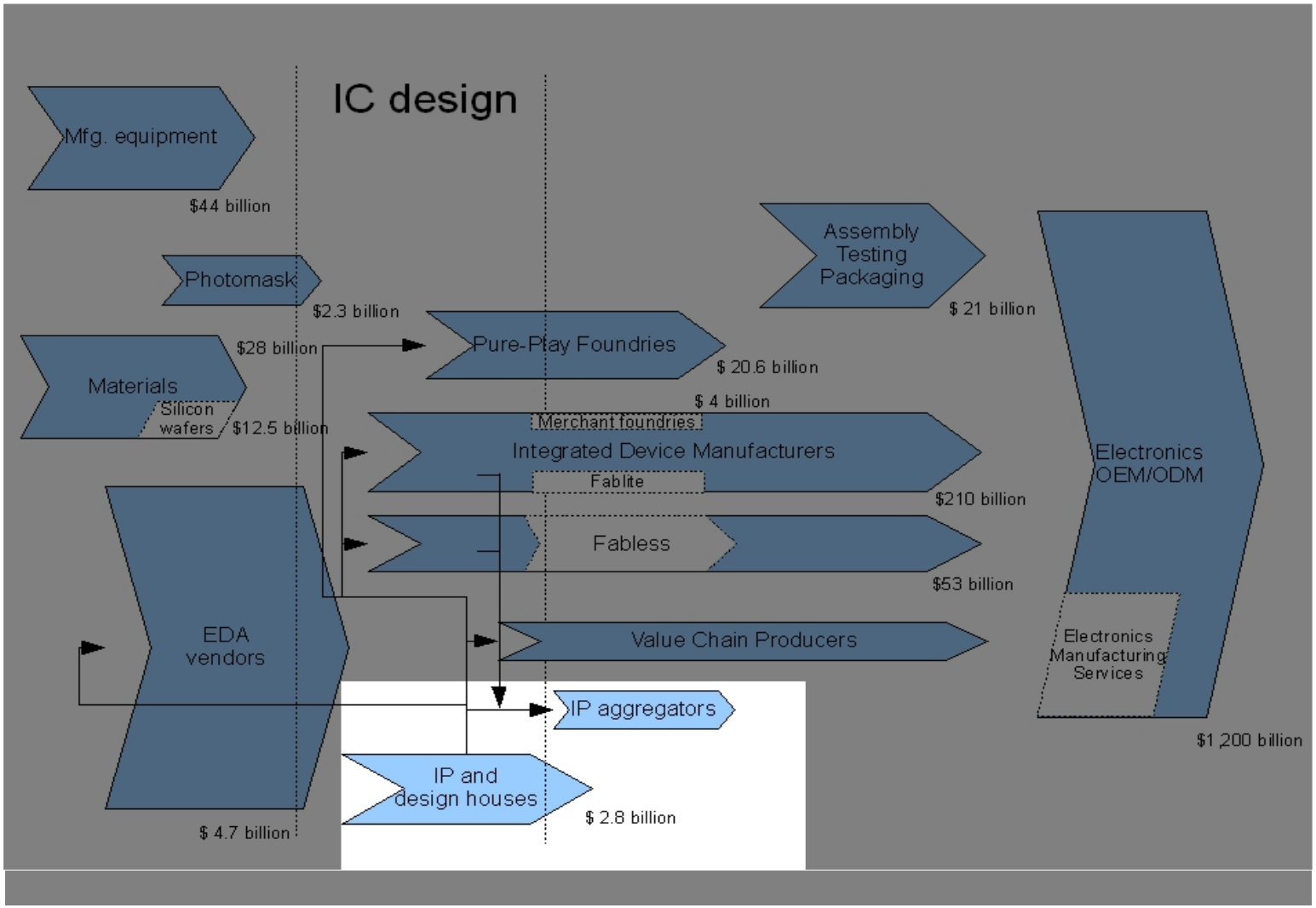
```
entity OR_ent is
port( x: in std_logic;
        y: in std_logic;
        F: out std_logic
);
end OR_ent;

architecture OR_arch of OR_ent is
begin
  process(x, y)
  begin
        if ((x='0')
```

Semiconductor IP cores are called ”intellectual property” cores because they are sold as intellectual property. Revenues come from royalties, license fees + related services.

The market leader is ARM. In 2007, about 3 billion chips were shipped with ARM processors. Every fourth electronic product manufactured in 2007 included a chip with semiconductor IP from ARM.

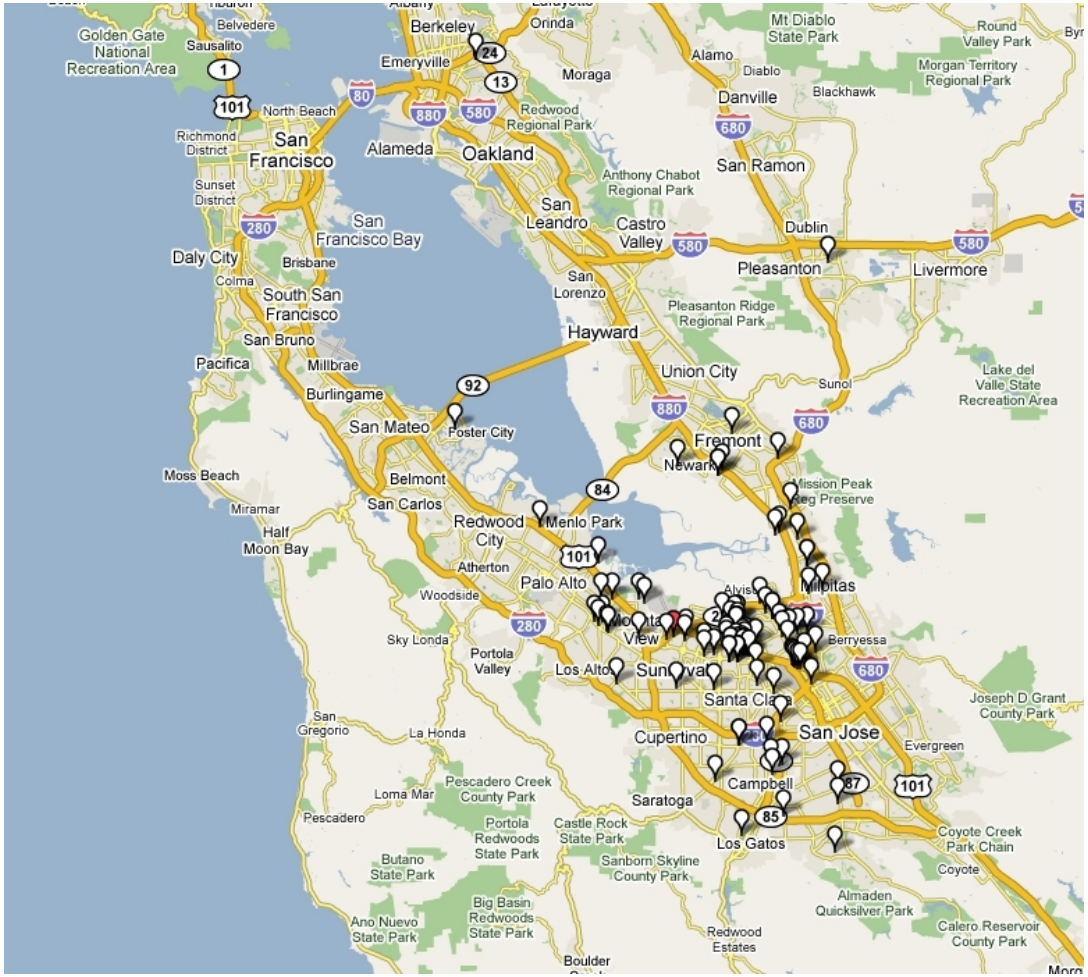
The Study Domain:
”Chipless” Semiconductor Producers

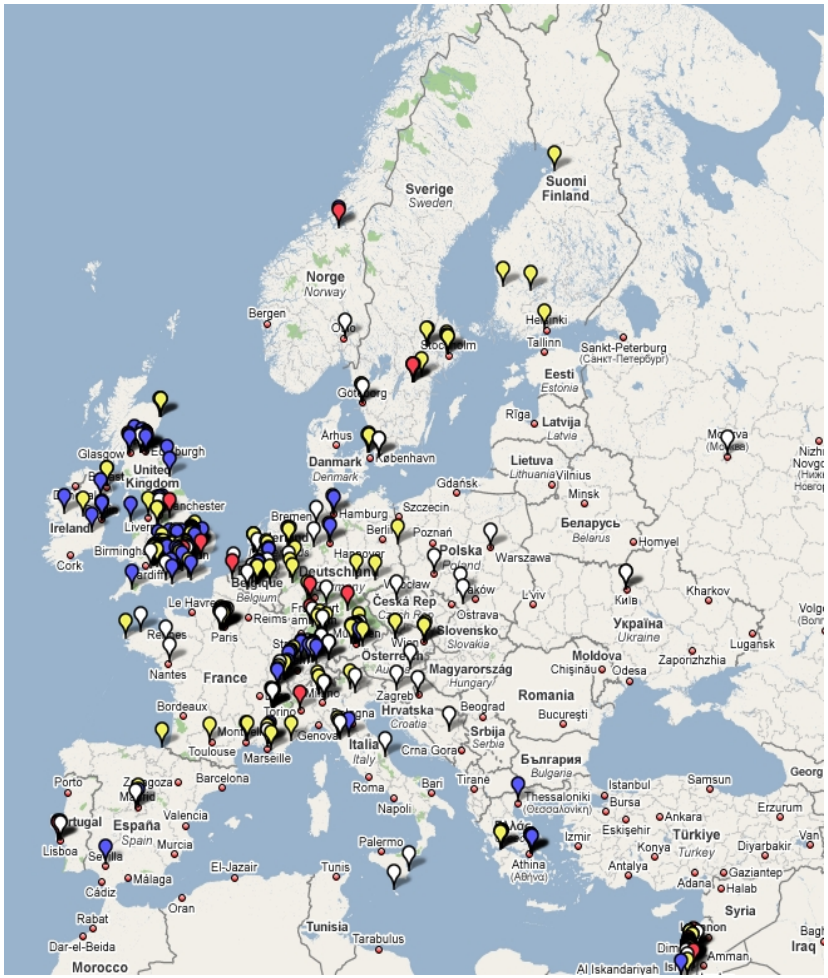


Top IP Core Vendors

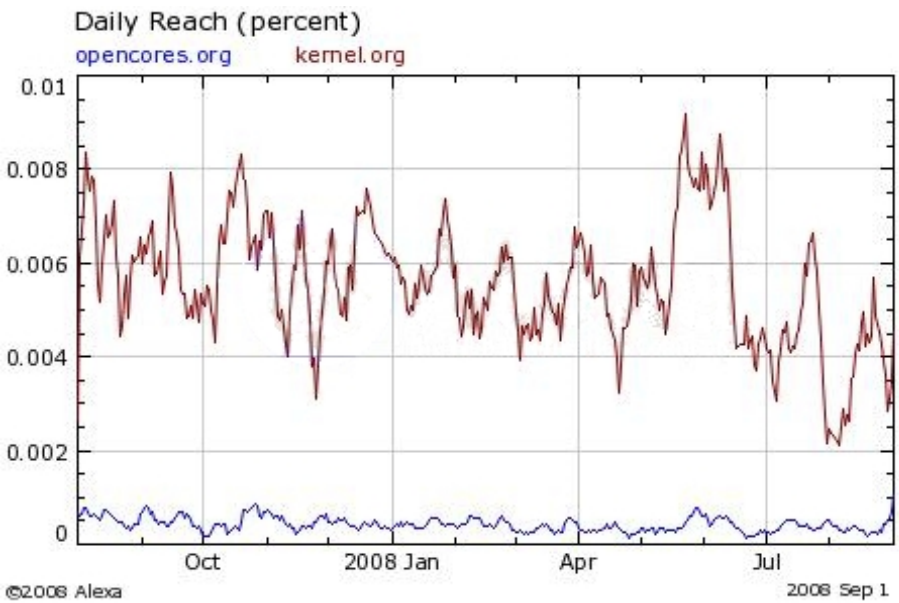
Rank		Employees	IP Revenue ($M)	Market Share	
2007	**Company**	**2007**	**2007**	**2007**	**Cumulative**
1	ARM	1728	516	29.6%	29.6%
2	Rambus	430	180	10.3%	39.9%
3	Synopsys	5196	97	5.6%	45.4%
4	Motorola-TTPcom	*286*	*87*	5.0%	50.4%
5	MIPS	196	83	4.8%	55.2%
6	Mosaid	112	57	3.3%	58.5%
7	Silicon Image	635	51	2.9%	61.4%
8	Virage Logic	417	47	2.7%	64.1%
9	Imagination Technologies	366	43	2.5%	66.5%
10	SST	715	40	2.3%	68.8%
11	CEVA	192	33	1.9%	70.7%
12	Chipidea	310	33	1.9%	72.6%
13	ARC	196	29	1.7%	74.3%
14	Mentor Graphics	4358	25	1.4%	75.7%
15	Wipro-Newlogic	350	*21*	1.2%	76.9%
16	Dolphin Integration	164	17	1.0%	77.9%

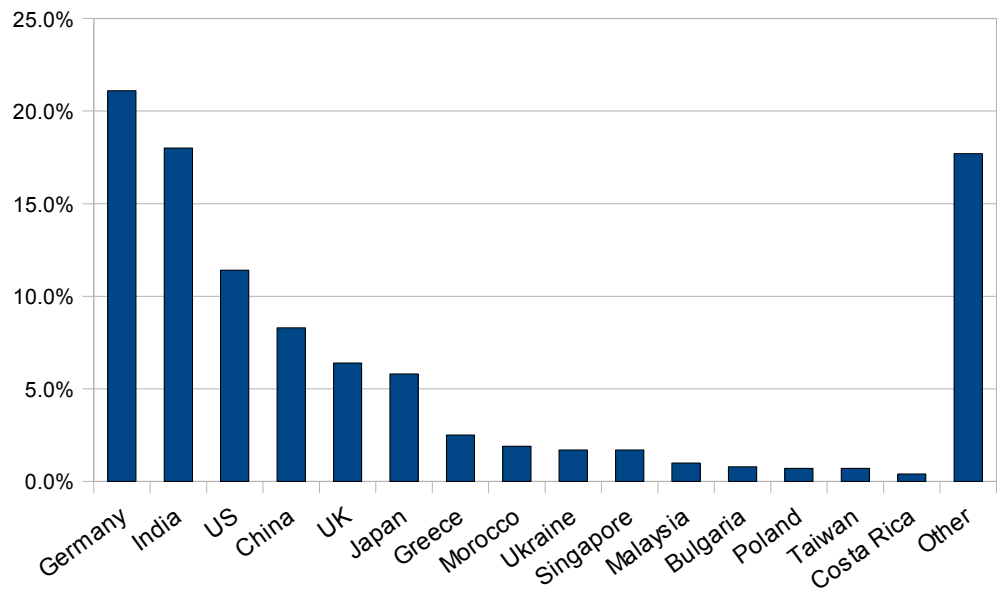
Many Small Firms





OpenCores.org
According to Alexa





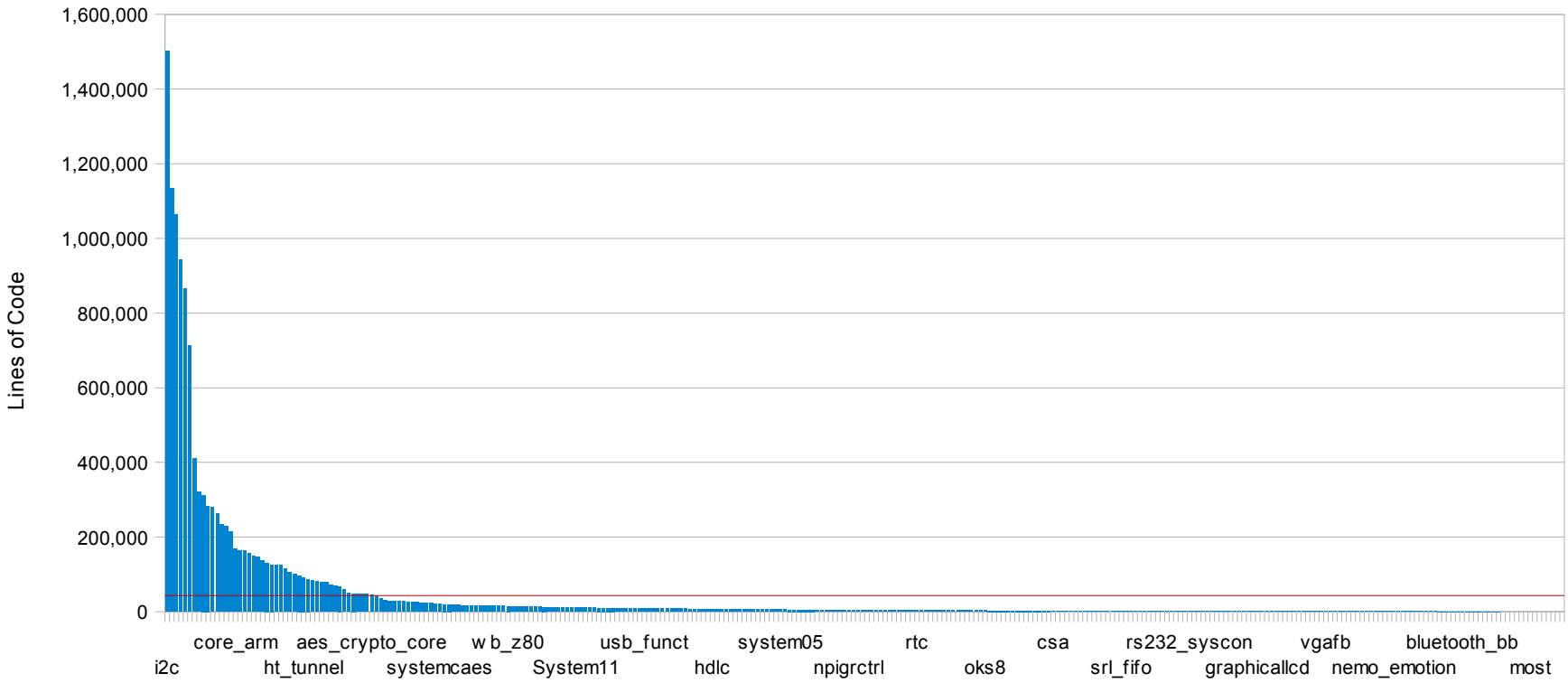
Visitor Countries

OpenCores.org CVS

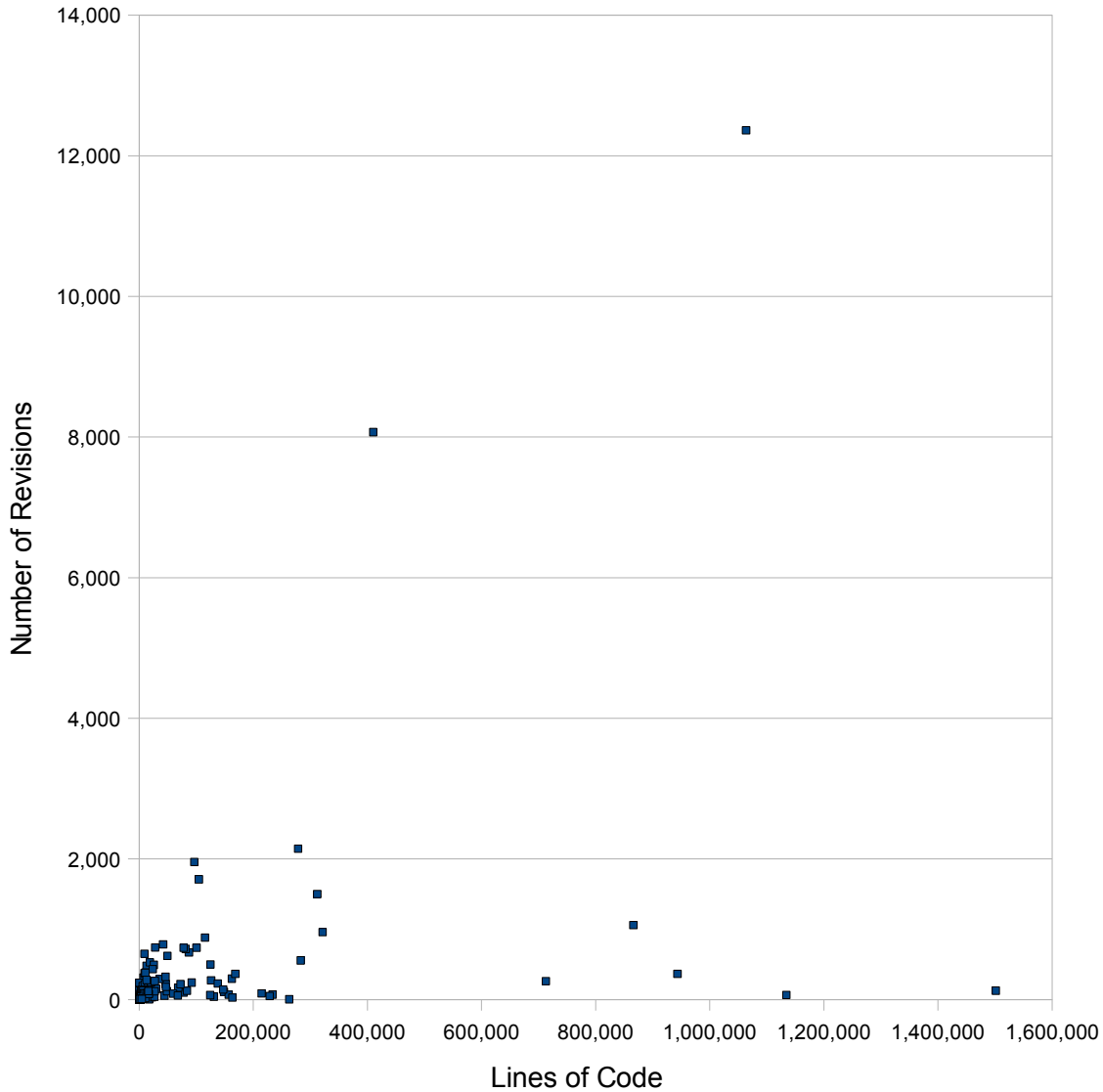
	# developers	lines of code	lifetime (months)	# commits	# revisions
mean	1.4	43,211	9.6	28.4	182.5
max	37	1,501,619	160	1712	12363
min	1	0	0	1	1
median	1	4,273	1	4	29

N= 308

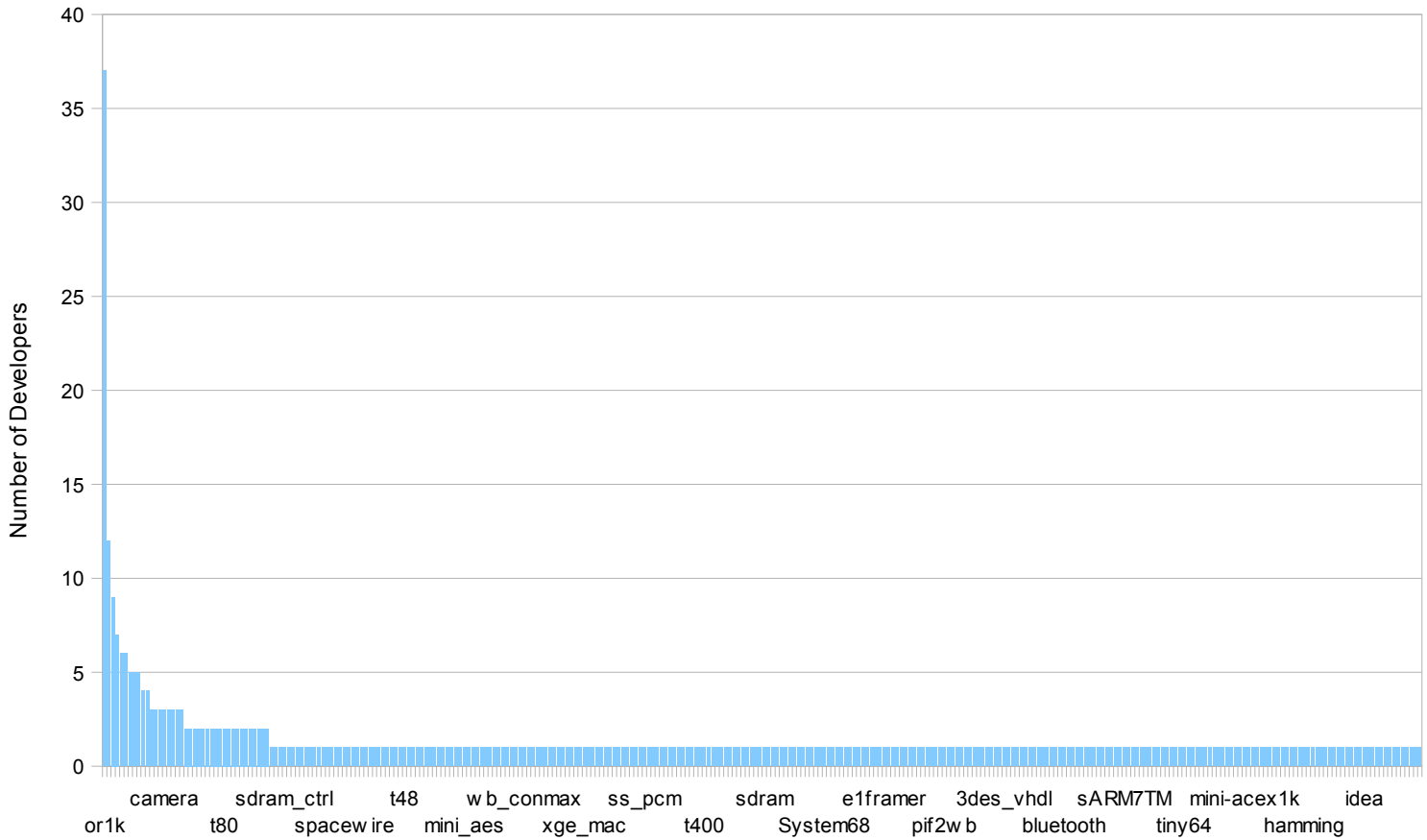
Projects by Code Size



Revisions

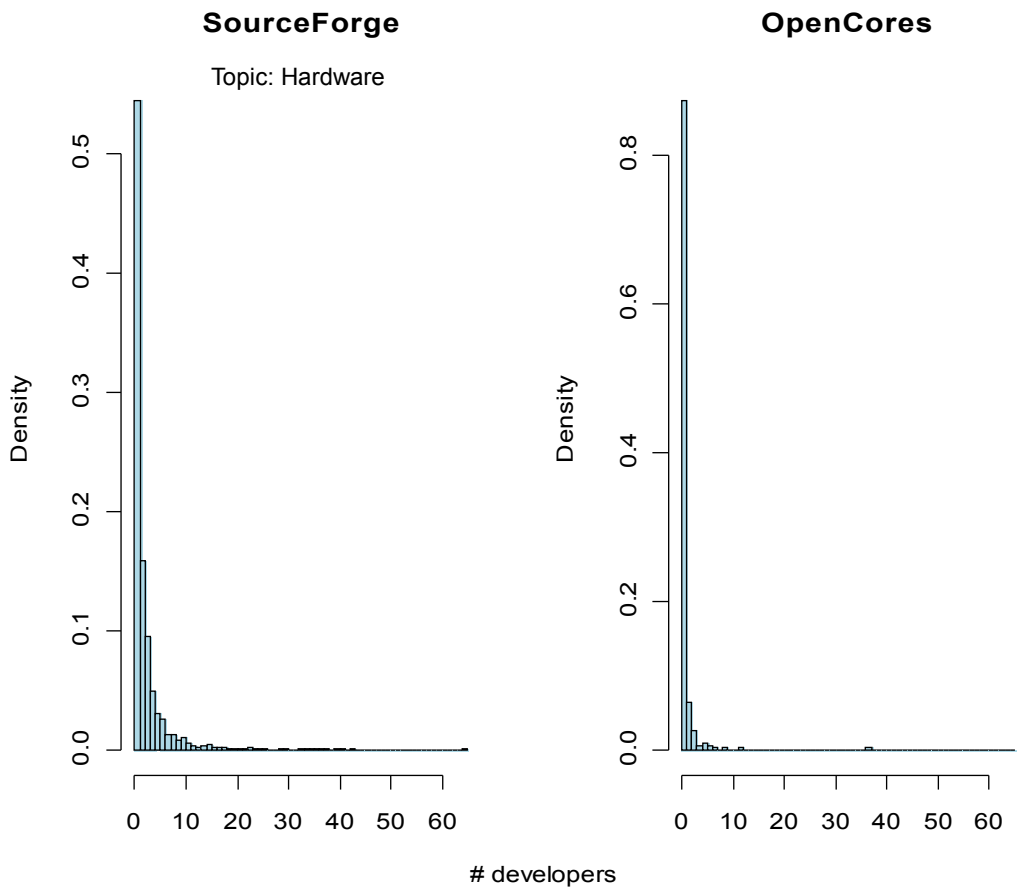

14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
Number of Revisions
0
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000
1,600,000
Lines of Code

Number of Developers



More than 5 developers: 6 projects (1.9 %)
More than 1 developer: 39 projects (12.7 %)

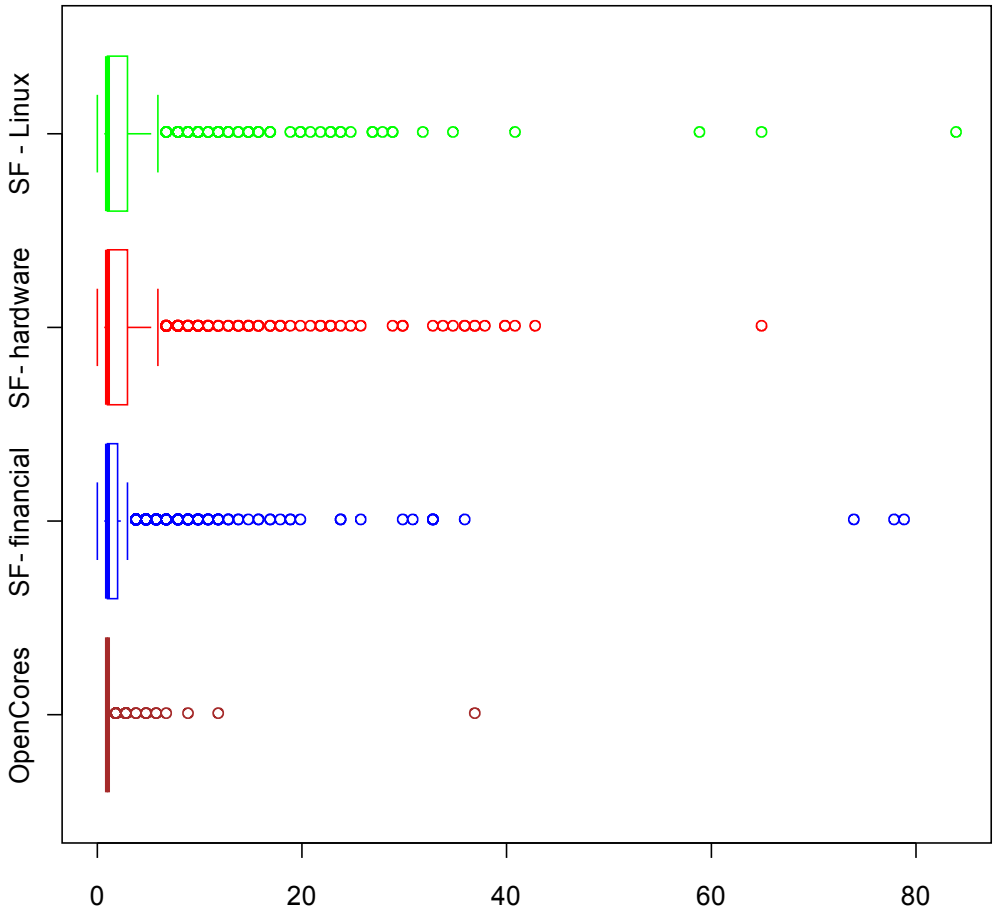
SourceForge vs OpenCores



Wilcoxon-Mann-Whitney, alternative: true location shift is not equal to 0
data: OpenCores and SF – hardware developers
W = 128623, p-value < 2.2e-16

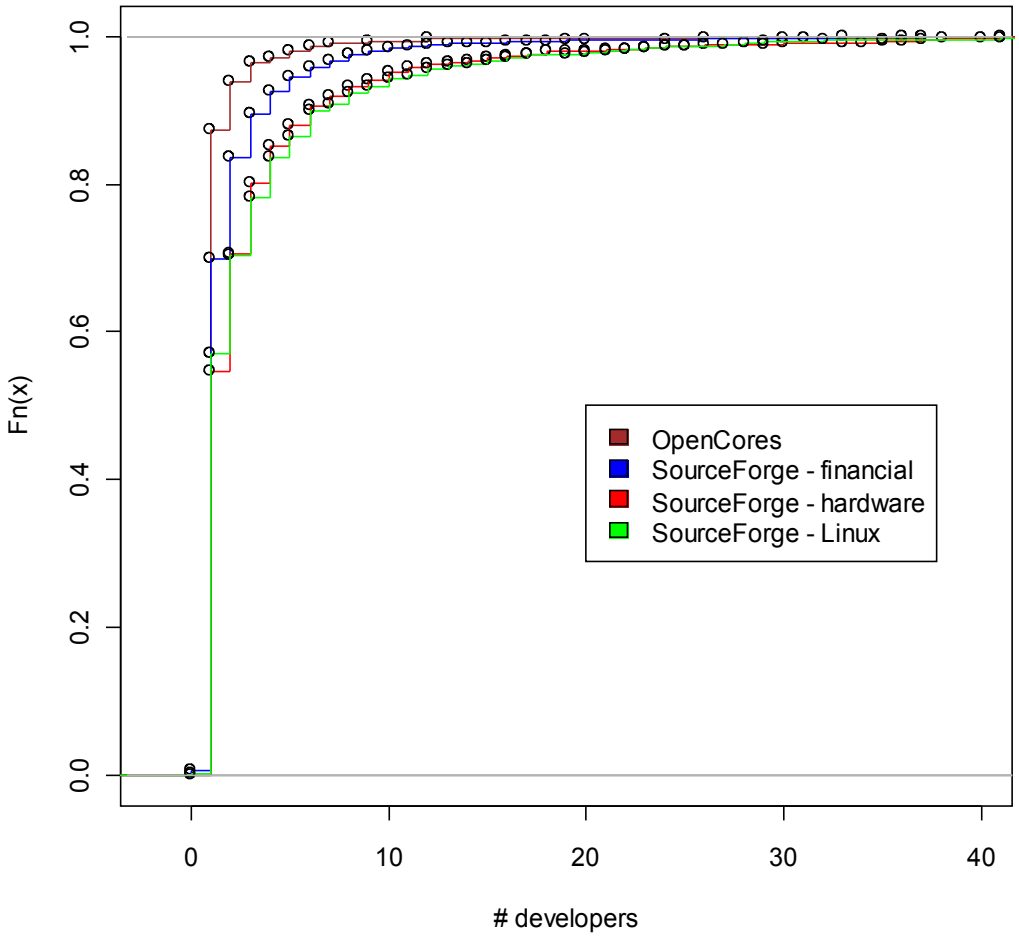
Project Developers

Close-to-Hardware, ”Social SW”, and Hardware



SF – financial excludes an outlier with 393 developers

Developer Distributions



	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	N
OpenCores	1	1	1	1.41	1	37	308
SF – financial	0	1	1	2.19	2	393	2435
SF – hardware	0	1	1	3.09	3	65	1263
SF – Linux	0	1	1	3.26	3	84	827
SF -financial (ex. 393)	0	1	1	2.02	2	79	2434

OpenCores are Different

Wilcoxon-Mann-Whitney, alternative: true location shift is not equal to 0

data: OpenCores and SF – hardware developers
W = 128623, p-value < 2.2e-16

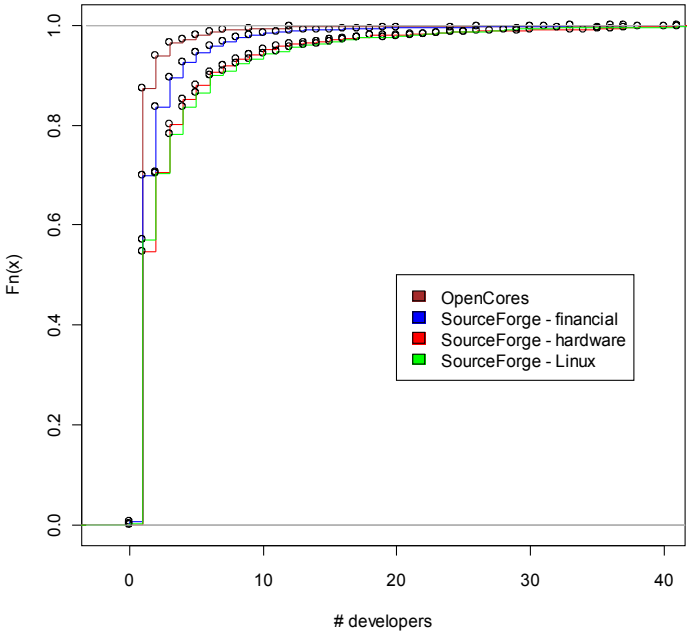
data: OpenCores and SF – Linux developers
W = 87228.5, p-value < 2.2e-16

data: OpenCores and SF – Finance developers
W = 311118.5, p-value = 9.675e-10

data: SF - Linux and SF – Finance developers
W = 1164896, p-value = 1.110e-15

data: SF - Linux and SF – Finance (ex. 393) developers
W = 1164896, p-value = 8.882e-16

data: SF - Linux and SF – Hardware developers
W = 515923, p-value = 0.606



Conclusions

- Most open source projects have a small number of developers
- SourceForge projects are different from OpenCores projects

OpenCores projects have significantly fewer developers

- The results are compatible with the thesis:

Close-to-hardware software is easier to develop in an open distributed development model than hardware

- Hardware-related software also seems to be easier to develop than ”social SW” (e.g. accounting, ERP, ...)

This is compatible with the thesis:

  - When there is no shared ontological basis, development becomes difficult

- When project outputs are not ”tightly-coupled functional systems,” different dynamics are possible

e.g., WikiMedia, representative democracy, ...

Discussion

- The size of developer communities (number of people committing code) is different for close-to-hardware, social software and physical hardware projects.
  - Does this reflect difficulties in coordination and collaboration? Or something else? (e.g., the number of potential competent developers)
- Is open source development *innovation* or *problem solving*?
  - Is this ”open innovation,” or ”distributed collaboration”?
- Assume the *downstream multi-focal innovation model (Tuomi, 2002)*:
  - ”Innovation happens when social practise changes.”
  - ”There are always several stakeholder communities, each with their own systems of meaning and measures of progress.”
  - ”An innovation artifact is often a 'carrier' of multiple innovations that have only loosely coupled evolutionary paths.”
  - ”Innovations are always retrospectively described, usually from the dominant point of view.”
  - ”Mainstream innovation histories fill the gaps and invent facts to make the story convincing.”
- Is the translation from abstract functional specification to a physical device a process that can destroy interpretative flexibility? (e.g., by sedimenting the innovation in a way that makes it difficult to reinvent and misuse it?)