Hardware Open Source

Cramium Labs

Agenda

- 1. Background – IPR
  - what Intellectual Property Rights apply to hardware
- 2. ASIC design flow
  - substantial IP differences versus SW design flow
- 3. Copyleft
  - what is copyleft, and why it is difficult to apply to hardware
- 4. Practical considerations
  - open source HW – does it achieve an intended purpose?
- 5. License Approaches
  - background of some efforts to create open source HW license
- 6. Cramium Approach
  - description of Cramium's delivery approach for the Daric chip

- Two lists at the end:
  - **[R#]** are **References**, papers, association links, etc.
  - **[L#]** are **Licenses**.

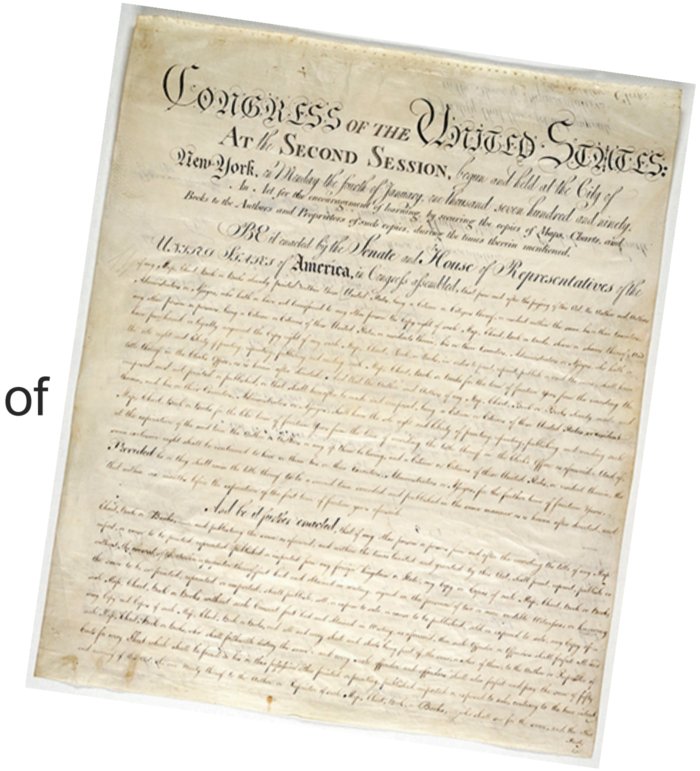
acronym definitions are pasted throughout the slides, at the first usage, in green text like this

$\downarrow$

ASIC – Application Specific Integrated Circuit, commonly used to mean any chip with complex logic.

1. Background – IPR

- Unfortunately, the issues here are so fundamental that we have to start with a basic background of copyrights.
- US Copyright law started with the 1790 copyright act, protecting only “maps, charts and...books”
- The world got more complicated, and this was revised many times to includes many other types of expression.
- Guidelines developed about when to use Copyright versus other types of rights [R1]
  - Utilitarian Articles
  - Idea vs. Expression dichotomy
  - Merger doctrine
  - Scènes à Faire doctrine



Congreſs of the United States:
At the Second Session, begun and held at the City of New-York, on Monday the fourth of January, one thousand seven hundred and ninety.
An Act for the encouragement of learning, by securing the copies of Maps, Charts, and Books to the Authors and Proprietors of such copies, during the times therein mentioned.
Be it enacted by the Senate and House of Representatives of the United States of America, in Congreſs assembled, [illegible]

[illegible]

Background (cont'd)

- Regarding Software, a seminal event was Apple vs. Franklin [R8]
  - SW source can be protected by copyright
  - This includes the binary/executable form
- Note that this was by no means straightforward.
  - District court rule in favor of Franklin, this was reversed by appellate court.
  - Especially the executable form was much debated.

Does this apply to board-level HW? Does it apply to ASIC design?

- References [R1, R3, R4, R6] all give in depth analysis of this question, and its implication.
  - However, they are often different in their conclusions.
- [R3] A chip design involves diverse design artifacts which are to varying degrees human readable, correspond in varying degrees to the final chip, and involve varying degrees of freedom of expression.
- Much debate about which type of intellectual property right is applicable to each of these artifacts.

Extending principles to hardware

- Many analogies between ASIC development and SW case law can be made (and are made in the references, especially [R3])
  - But opinions in articles are not case law.
- There are bits and pieces of law, for example:
  - [R23] holds that a schematic is copyrightable
    - But this is just one of many types of design artifact within an ASIC development.
  - [R13] There is also US SCPA (Semiconductor Chip Protection Act) of 1984
    - Creates a *sui generis* right of “maskwork”, neither patent nor copyright.
      - e.g. must be registered, unlike copyright
      - Note that only a small minority of chip tapeouts are actually registered in this way [R24].
        - » The registration process itself has some complexities (disclosure of 3rd party IP)
      - Also, the mask is only the final stage of a chip, many steps and processes downstream from the first “source”

So where does that leave us..?

- It is a long story, but here is one opinion for example (Jonathan Kuniholm [R16])
  - “Every single effort to tackle the problem of open hardware licensing has failed to acknowledge that it is unclear what we are licensing (TAPR, CERN, OHANDA, OSHW, you name it), and if any license will withstand a legal challenge. Open source software has a legal basis in the copyright of source code AND the executable--perhaps most importantly in the copy of the executable made in RAM at startup. Without a legal basis for ownership rights, there is nothing to license, and it is pointless to discuss the fine points of a particular license.”
- It is beyond the scope of this presentation to really cover the controversy.
  - Your takeaway should be that controversy exists.
  - The References cover it more completely.
    - [R1, R3, R4] are most cited;
    - [R2, R5, R6] are instructive to read also.

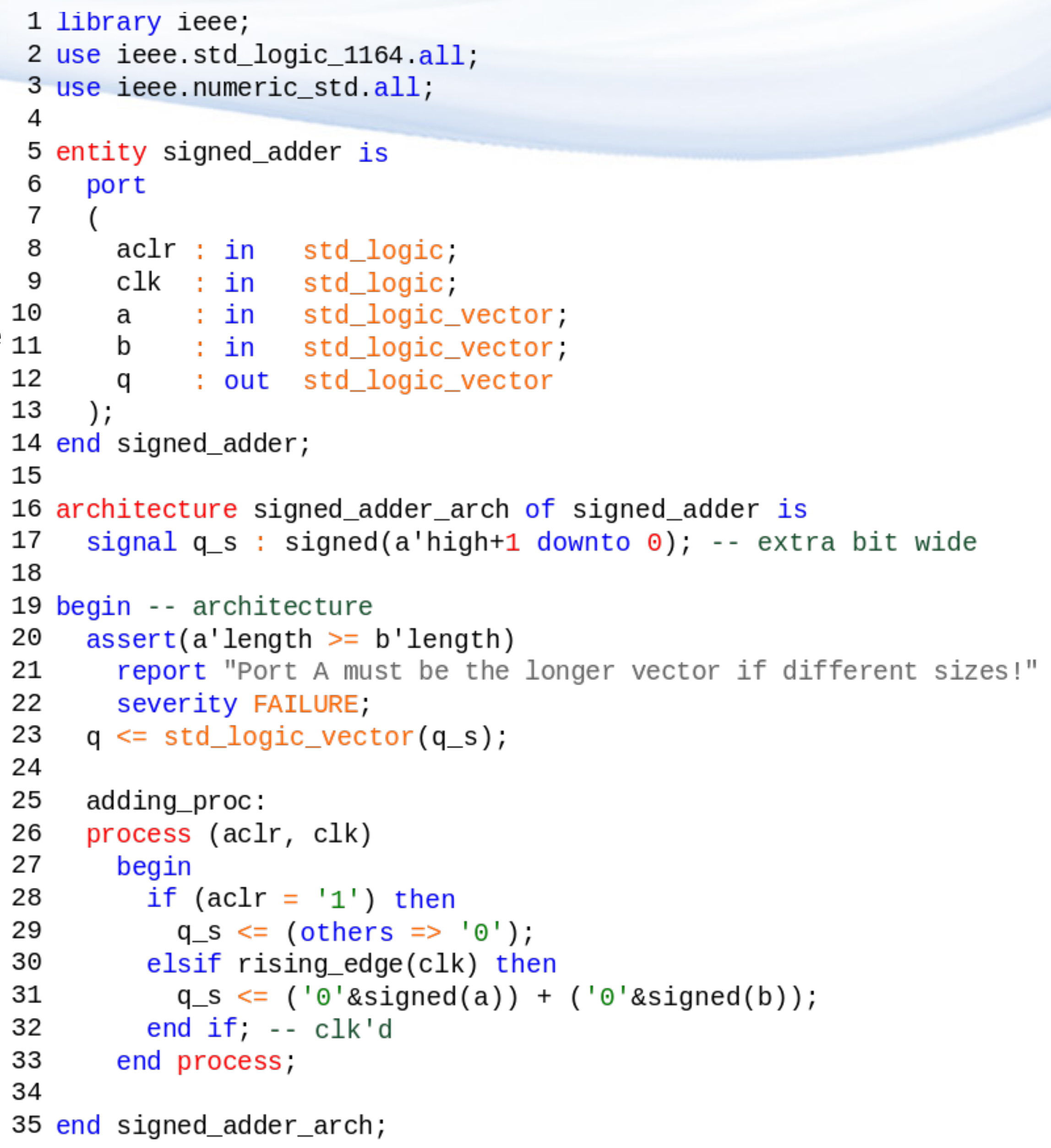
2. ASIC design flow

- This is a brief aside of a typical ASIC design flow, so you can understand what types of IP might come in, and from what sources.
  - This simplified view skips steps that are tangent to the subject of licensing.
  - This just covers ASIC flow, but FPGA could be considered a subset of this.
    - e.g. in FGPA, synthesis maps logic onto already existing elements in the FPGA.
- RTL is like a concurrent programming language
  - Commonly VHDL, Verilog and variants of these.
  - Like software, it is a human-readable text file.
  - Can include not only flow but timing constraints.

FPGA– Field Programmable Gate Array, a finished semiconductor product that can be programmed to connect its gates in arbitrary patterns.

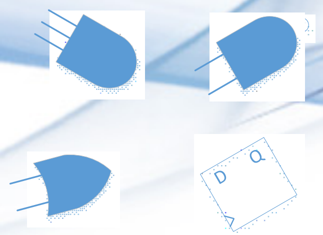
RTL – Register Transfer Level

VHDL – acronym is not instructive, just think of it as a language name.



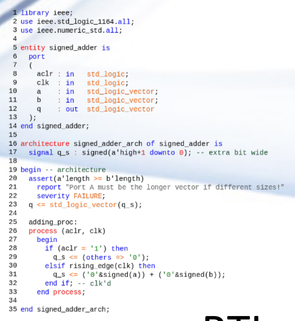
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity signed_adder is
  port
  (
    aclr : in   std_logic;
    clk  : in   std_logic;
    a    : in   std_logic_vector;
    b    : in   std_logic_vector;
    q    : out  std_logic_vector
  );
end signed_adder;

architecture signed_adder_arch of signed_adder is
  signal q_s : signed(a'high+1 downto 0); -- extra bit wide

begin -- architecture
  assert(a'length >= b'length)
    report "Port A must be the longer vector if different sizes!"
    severity FAILURE;
  q <= std_logic_vector(q_s);

  adding_proc:
  process (aclr, clk)
    begin
      if (aclr = '1') then
        q_s <= (others => '0');
      elsif rising_edge(clk) then
        q_s <= ('0'&signed(a)) + ('0'&signed(b));
      end if; -- clk'd
    end process;

end signed_adder_arch;
```

Synthesis



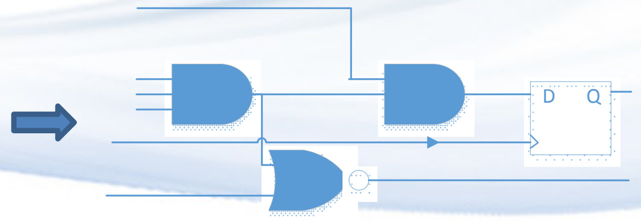
Library

+



```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity signed_adder is
  port
  (
    aclr : in   std_logic;
    clk  : in   std_logic;
    a    : in   std_logic_vector;
    b    : in   std_logic_vector;
    q    : out  std_logic_vector
  );
end signed_adder;

architecture signed_adder_arch of signed_adder is
  signal q_s : signed(a'high+1 downto 0); -- extra bit wide

begin -- architecture
  assert(a'length >= b'length)
    report "Port A must be the longer vector if different sizes!"
    severity FAILURE;
  q <= std_logic_vector(q_s);

  adding_proc:
  process (aclr, clk)
    begin
      if (aclr = '1') then
        q_s <= (others => '0');
      elsif rising_edge(clk) then
        q_s <= ('0'&signed(a)) + ('0'&signed(b));
      end if; -- clk'd
    end process;

end signed_adder_arch;
```

RTL

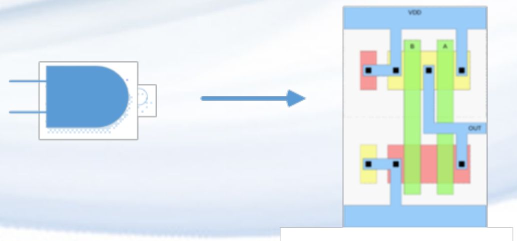


Netlist

- Synthesis maps this abstract form into a specific library
  - Library is a logical view of small blocks that have been designed and characterized, so that that in a specific foundry they have known size, setup time, hold time, drive strength, etc.
- This results in a Netlist, or “sea of gates”
  - Not very human readable, and limited room for “expression” in synthesis
- Some authors in References make analogy to “compilation” of SW into an instruction set.
  - As in SW compilation, there can be directives to prioritize size or speed (or other parameters)
  - Note use of word “library” is completely different from the meaning in the field of software; “Library” in software may be more analogous to “core” in ASIC development (*)
- A Library already has :
  - 1. Some information about the foundry process
  - 2. All the individual library cells, laboriously designed, often by a 3rd party.

- (*) NOTES:
  - Unfortunately “core” can mean both : (1) specifically a CPU; or (2) more generally any block of functionality.
  - Also popular term is “IP”, although [R6] explains why this is poor nomenclature for a block of ASIC function.

P&R and following steps



- Place & Route attempts to place this in a physical layout of cells, and routing of multiple layers of metal for connection.
- Extraction tools can extra R, C, L, etc. from the layout and back-annotate the netlist to more accurately check timing. Again, this involves key information from the foundry process.
- However, this is just digital part. ASIC generally requires much more:
  - SRAM – these are highly optimized having much better size/performance/power than synthesized logic. Generally this IP comes from another source (not the chip designer)
    - Memory compiler takes high level directives and spits out complete layout of the SRAM.
  - ROM – similarly, tool-generated
    - But in this case contains contents also
  - I/Os, ESD structures – generally foundry or 3rd party IP.
  - Very commonly other 3rd party IP: PLLs, A/D or other mixed signal, power regulators, processor core, NVM/fuse, TRNG, drivers for common I/Os such as USB, specialized logic functions, etc.
  - These have various business arrangements, and may be plugged in at various stages of design
    - RTL
    - hard macro instantiated by fabless company
    - frame view, with actual layout plugged into final GDSII by the foundry.

PLL – Phase Locked Loop, typically used to generate ASIC clocks

SRAM – Static RAM

ESD – Electro-Static Discharge

NVM – Non-Volatile Memory

TRNG – True Random Number Generator

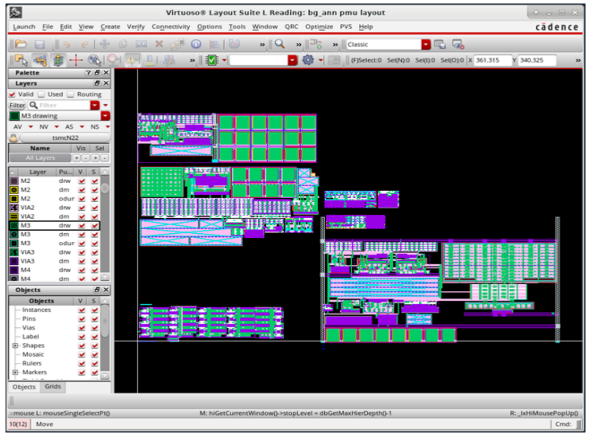
GDSII – a file format used to convey final chip mask design.

..and more steps

STA – Static Timing Analysis.

PDK – Physical Design Kit, large collection of information from foundry to indicate how to make designs for that foundry.

- Timing check involves laborious development of scripts for the STA tool, e.g. discarding false paths.
- Testbenches are more integral to instantiation than in the case of SW, due to high costs.
  - Customer of a block will expect vendor to include a means of verifying it at the chip level.
- Production test insertion adds yet more IP sources, e.g.
  - Mbist controller for memories
  - Scan insertion for logic
- Design will usually also include analog blocks.
  - In this case schematic is drawn incorporating active and passive devices
  - Uses sensitive information from foundry PDK for design and verification.
  - From this, a more manual layout flow is done (as opposed to digital P&R)
- Many quasi-manual tweaks to layout
  - About 6 weeks from final synthesis to tape out, as opposed to ~minutes for SW.
    - DRC / LVS fixes
    - Dummy metal
    - ECOs are manual fixes (tool-assisted)
- Final stream out is in GDSII form, which is converted to photographic plates, which are directly used in manufacturing



DRC – Design Rule Check

LVS – Layout Versus Schematic

ECO – Engineering Change Order (not an illuminating acronym, see conext in slide)

Summary of ASIC design flow

- The design process has many stages, each of which involve varying 3rd party IP, varying levels of “expression”
- Also prolific use of tools to massage the database and even insert content
  - vendor policy on IPR may vary (and may take an interesting position, such as inserting a copyright notice into the output file)
- Clearly this involves more manual labor steps than running a makefile for SW
  - And many more types of design artifact (not just source->object->binary)
  - And many more contributors of intellectual property
- All of this raises many questions
  - Is each artifact “expression of idea,” or useful item, or scènes à faire ..?
  - What is the appropriate protection for each design artifact (patent, copyright, maskwork...)
  - What exactly is “source”?
    - And particularly, what is “Corresponding Source” (e.g. as defined in GPLv3)

3. Copyleft

- Copyleft was coined as a term to describe licenses designed to *promote / require* proliferation, rather than *prohibit / restrict* proliferation.
- Entity taking an inbound copyleft license will receive a license to use the code, but with requirements to publish and license (usually under the same terms) any derivative works.
  - Usually triggered by "distribution" of the derivative work.
- Among the key issues* are
  - What is a derivative work that is subject to this requirement;
    - in other words, what is the copyleft boundary.
  - What counts a distribution that triggers the requirement.
- Licenses are described as:
  - Permissive – i.e., non-copyleft
    - e.g., Apache, MIT, BSD...
  - Weakly reciprocal – sets narrower rules for what is included in the copyleft boundary.
    - e.g. LGPL
  - Strongly reciprocal – casts a wider net for what is included in the copyleft boundary.
    - e.g. GPL

(*) Another key issue is whether this license creates a contract, and if so between what parties [R26, R27].

Looking at SW licenses...

- In software, guidelines can be categorized loosely as :
  - 1. There is a small amount of broad text in licenses directly;
  - 2. There is a **vast** corpus of literature, opinion, de facto practice, interpreting what is subject to copyleft, what counts as distribution, etc.;
  - 3. There is a small but growing body of case law [R10]
- Covering this in any detail would take a long time – there are entire careers just in this space.
- However the salient point is, you will find, over and over, (in license, opinions and case law), terms such as: program, compiled; object file; dynamically linked; statically linked; system library; command line interface; pipe; socket; remote procedure call, etc.
  - Even in SW world, where it is quasi-clear what the individual terms mean, interpreting their relation to the license text is a field of very active controversy.
  - When discussing ASICs, the situation is exacerbated (perhaps to the point of futility) considering that many software terms have no clear analog in an ASIC

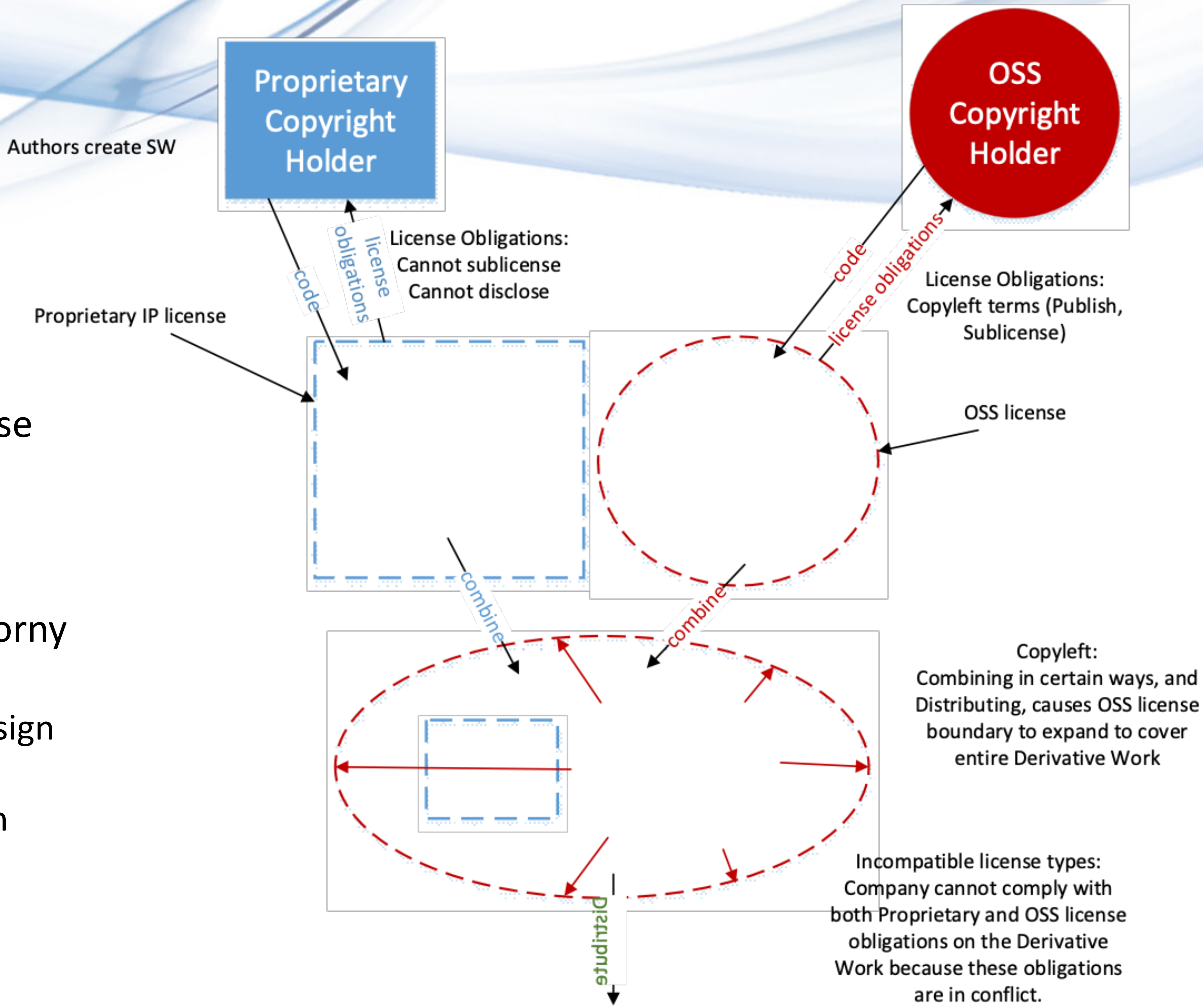
- For example, excerpt from GPL v3.0, setting boundary of Corresponding Source (i.e. that which must be included and further licensed).

"Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work. "

(See also [R29] for further discussion of what does not trigger copyleft – also filled with software terms)

Incompatibility

- User of inbound copyleft license risks **license incompatibility**
- Diagram illustrates one such example, in the case of SW.
- This example is particularly thorny in ASIC development.
    - As previously noted, ASIC design flow is complex, with many contributors of many types in each type of design artifact


Authors create SW
Proprietary Copyright Holder
OSS Copyright Holder
code
license obligations
License Obligations:
Cannot sublicense
Cannot disclose
Proprietary IP license
code
license obligations
License Obligations:
Copyleft terms (Publish, Sublicense)
OSS license
combine
combine
Copyleft:
Combining in certain ways, and Distributing, causes OSS license boundary to expand to cover entire Derivative Work
Distribute
Incompatible license types:
Company cannot comply with both Proprietary and OSS license obligations on the Derivative Work because these obligations are in conflict.

Risks and consequences

- Enforcement actions can come about
  - for ideological reasons
  - for cynical financial reasons (trolling / extortion)
  - for some combination of the two
- Copyright has statutory damages [R1]
  - Unlike a contract breach, this does not require establishing actual damages
- A software provider might quickly and cheaply modify code and redistribute.
- An ASIC vendor has no such recourse. Modifying an ASIC takes a lot of both time and money.
  - These facts may give a troll more leverage.

4. Practical Considerations

Let’s step back and consider what is the purpose of Open Source in the first place.

- Consider for example the “Four Freedoms” expressed by Gnu.org:
  - *The freedom to run the program as you wish, for any purpose (freedom 0).*
  - *The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.*
  - *The freedom to redistribute copies so you can help others (freedom 2).*
  - *The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.*

- Even if we can define “source”, does providing it enable these freedoms?
  - Parts can be emulated in FPGA, but this is slower and incomplete.
  - Complete mixed-signal simulation is impractically slow.
    - Also the best practice, with most expensive tools, and unlimited compute resource, still cannot precisely predict things like sidechannel signatures, and countermeasure effectiveness.
      - These must be tried in silicon, and to be more precise, across many units and conditions.
  - Actually modifying and making a derivative work could cost millions of dollars.
  - All of the above may involve foundry IP and other third party IP.

ASIC design for the masses...?

- There are low-NRE design flows, which have a place in the ecosystem,
  - But they are not anywhere near the same class as high-NRE design flow in terms of unit cost, performance, size, power, speed.
- We all know advancing nodes have led to exponential performance improvement
  - But not surprisingly, this is accompanied by exponential development cost.
- Consider that a180nm mask set is about $100K; 5nm is $10-15M

- [R1] sums up the difference between SW vs HW thusly: “electrons are free, atoms are expensive.”

Inspectability

- Key difference between ASIC and SW is -- unless you own a 10 billion dollar foundry (and mask lab, and test house, etc.) -- you will necessarily transfer the design and procure through vendors.
  - In contrast, the average engineer already personally owns a several machines capable of building and running SW.
- So how do you know that the ASIC you get back matches what you sent?
  - And unlike SW, each is a physical object, so just because one is inspected doesn't mean all have been inspected.
- There is no simple "hash" like there is a for a flat binary file.
  - Scan helps, but may be disabled for security reason.
  - Signature helps, but only to a point
- Optical scan helps but is limited (encapsulation, geometry)
- The aforementioned 3rd party IP issue also limits allowed disclosure for inspection

Business considerations

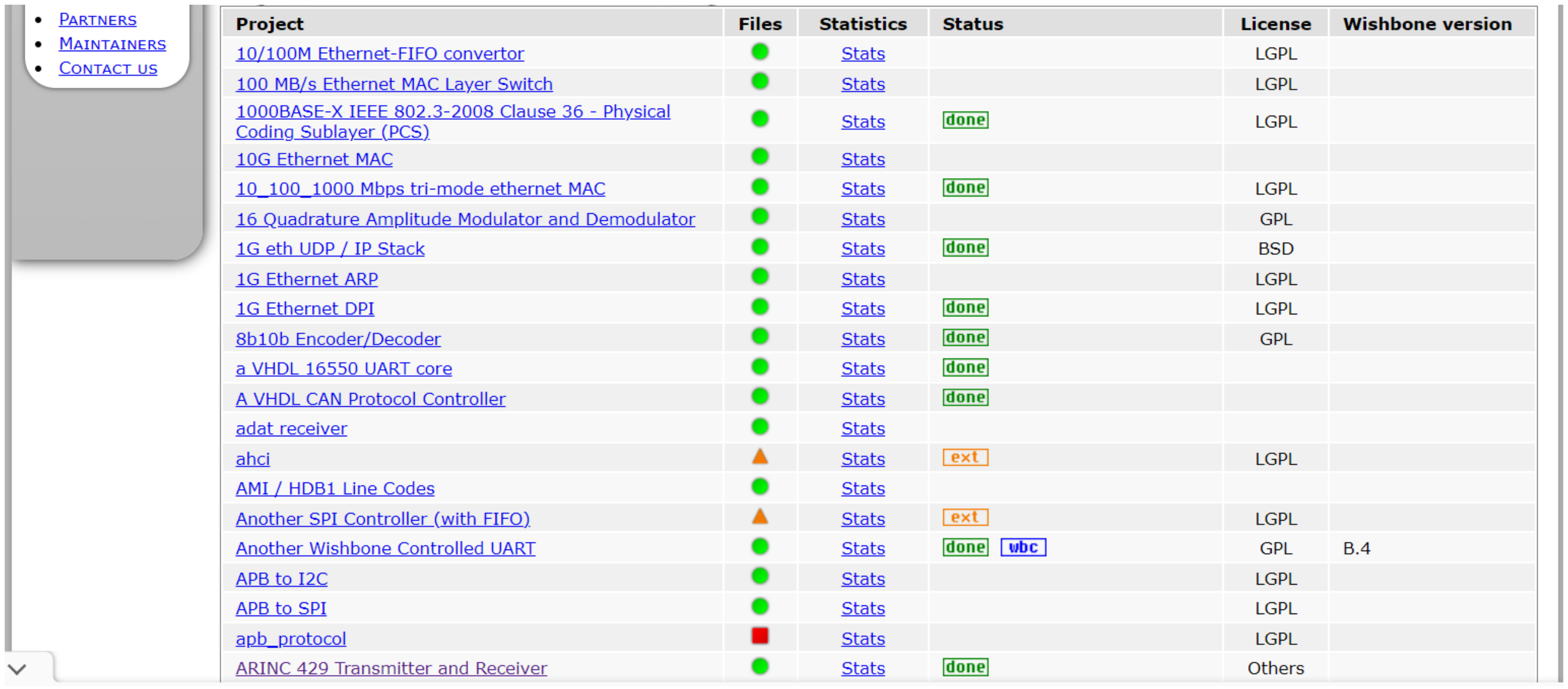
- The business considerations are also different

- In SW, primary costs are often:
  - Marketing cost, cost to acquire customer
  - Operational costs
  - Customer support cost
- In chips, primary costs are:
  - Non-recurring costs (engineering design and tooling)
  - Recurring costs (manufacturing cost-of-goods: wafer, test, package, etc.)

- Note that these cost barriers apply not only to the open source licensee, but to the licensor.
  - Licensee has barriers to “the four freedoms”
  - But similarly, even the licensor cannot immediately or easily benefit from improvements to the source materials in any derivative works

5. License Approaches

- Perhaps the most common approach is to simply use a software license.
  - Addresses all the aforementioned problems via the ostrich method.
- For example, open any tab at opencores.org



- Partners
- Maintainers
- Contact us

Project	Files	Statistics	Status	License	Wishbone version
10/100M Ethernet-FIFO convertor	●	Stats		LGPL	
100 MB/s Ethernet MAC Layer Switch	●	Stats		LGPL	
1000BASE-X IEEE 802.3-2008 Clause 36 - Physical Coding Sublayer (PCS)	●	Stats	done	LGPL	
10G Ethernet MAC	●	Stats			
10_100_1000 Mbps tri-mode ethernet MAC	●	Stats	done	LGPL	
16 Quadrature Amplitude Modulator and Demodulator	●	Stats		GPL	
1G eth UDP / IP Stack	●	Stats	done	BSD	
1G Ethernet ARP	●	Stats		LGPL	
1G Ethernet DPI	●	Stats	done	LGPL	
8b10b Encoder/Decoder	●	Stats	done	GPL	
a VHDL 16550 UART core	●	Stats	done		
A VHDL CAN Protocol Controller	●	Stats	done		
adat receiver	●	Stats			
ahci	▲	Stats	ext	LGPL	
AMI / HDB1 Line Codes	●	Stats			
Another SPI Controller (with FIFO)	▲	Stats	ext	LGPL	
Another Wishbone Controlled UART	●	Stats	done wbc	GPL	B.4
APB to I2C	●	Stats		LGPL	
APB to SPI	●	Stats		LGPL	
apb_protocol	■	Stats		LGPL	
ARINC 429 Transmitter and Receiver	●	Stats	done	Others	

Hardware-specific licenses

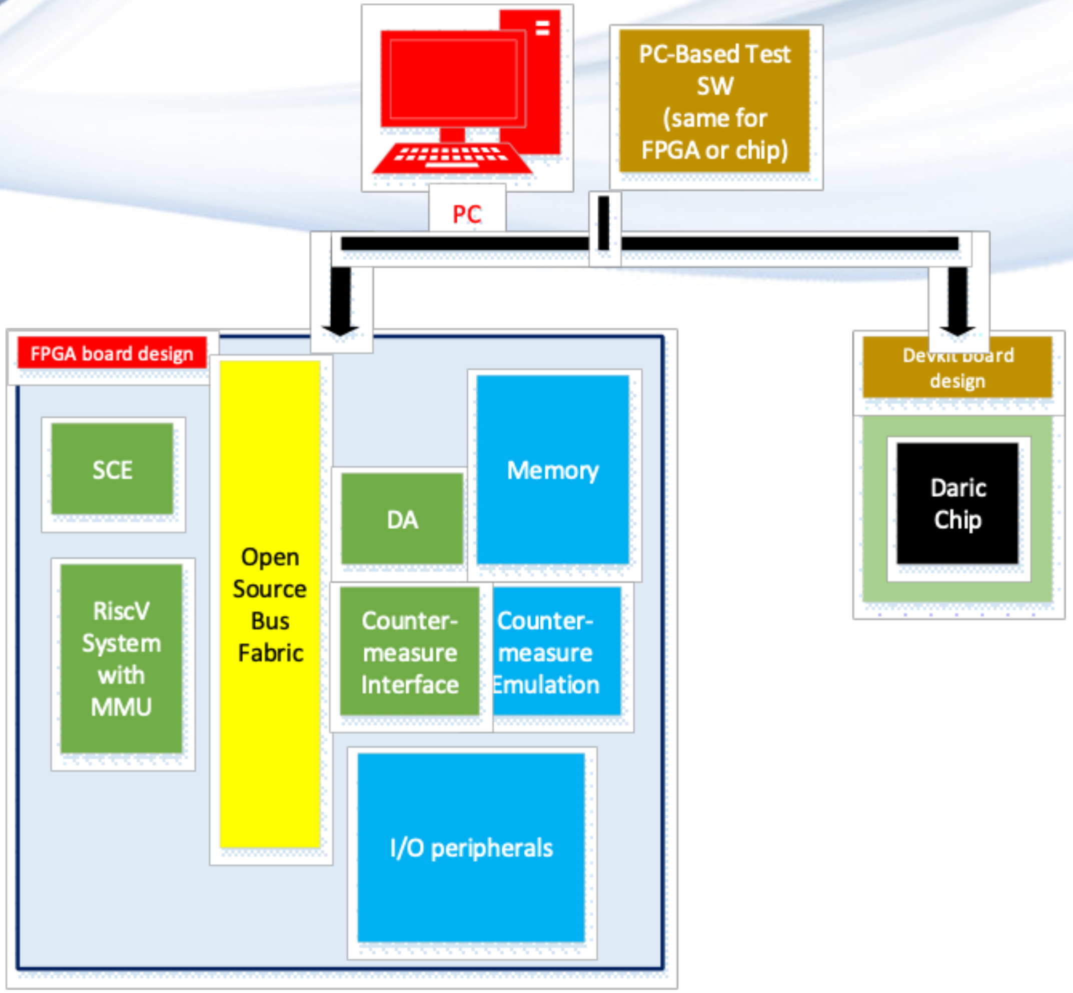
- First attempts to recognize differences between SW and HW focused mostly on board-level hardware. [L1, L2, L6]
- Two that find use in practice are Solderpad and TAPR
  - Solderpad [L1] is a modified Apache 2.0,
    - non-copyleft, which avoids some difficulties.
  - TAPR [L2]
    - Deal primarily with patents, not copyrights, for reasons explained in [R1]
    - Is a copyleft license
- Later efforts incorporate some ASIC thinking
  - CERN [L3] started as board-level, but especially with v2 puts more effort into incorporating ASIC thinking [R2]
  - Offers permissive (non-copyleft), and weakly/strongly copyleft versions (respectively, -P, -W, -S)
  - Designed from the ground up with copyleft language for HW; concept of “Available Component.”
- IPIL [L5] is primarily about ASIC, and [R6] serves as the “rationale” whitepaper.
  - But I am unware of any usage in industry
- The attached License list is the most complete I know of (aggregated from the References)
- References also link to groups with resources on OS hardware,
  - For example [R19, R16, R17]
    - OSHWA [R19] focus so far is mostly board-level HW and 3D printing.
    - OSHWA just had annual summit

6. Cramium Approach

- **Current plan, subject to change**
- The core components of the logical security are the RiscV CPU, the Secure Crytpo Engine (SCE), and the Data Access controller (DA)
- The SCE, DA open-sourced under the CERN-OHL-W license.
- RiscV based on VexRisc [R30]
- The PC test driver is also OSS (license TBD).
    - The PC test system can operate equivalently with the FPGA or Devkit
    - (Requires driver work, plan TBD)

- In this way, anybody can construct, inspect, or test an FPGA system that performs the same logical security operations as the chip.
    - And post-silicon, can run any test on the devkit to verify that the chip in fact behaves the same as the disclosed OS logic.

- NOTES:
    - Bus fabric is ARM AXI, so cannot be Open Source, but in the FPGA model is replaced with an Open Source equivalent.
    - Some generic items such as memory, I/Os are mimicked by similar-function items in the FPGA
    - Countermeasures are largely analog and layout, and so only the interfaces can be included in FPGA; the analog blocks are stubbed out and replaced by false stimuli for test purposes.



🟩	Corresponds exactly to chip at RTL level; same RTL is synthesized into FGPA and chip.
🟨	Implementation very closely corresponds to chip but cannot literally be the same RTL (due to 3rd party IP)
🟦	Generic function, Implementation may loosely correspond to chip (equivalent function)
🟫	Other open source. Ancillary material that does not appear inside IC, but is open-source.
🟥	Off-the-shelf, can be bought

Inbound IP

- In our case, there are not any copyleft inbound licenses
  - There are some permissive-license inbound RTL (mainly peripherals)
  - There are 3rd party proprietary inbound licenses
- A reasonable question might be, would our distribution comply with the CERN-OHL-W if our crypto engine design came in through CERN-OHL-W?
  - We are taking the approach that this is a desirable goal
    - However as explained in this presentation, definitive answers are hard to come by.
  - Note that as copyright owner, we do not rely on an inbound CERN-OHL-W license, so this is a hypothetical question

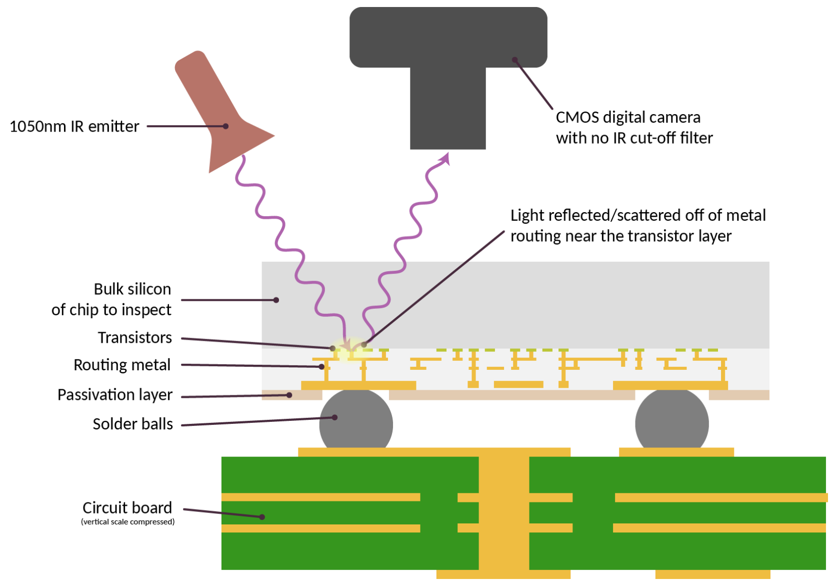
Inspectability

- Several customer projects, with varying approaches.
- Many options for supply chain validation
  - Each chip can have unique ID and certificate
  - Database of valid chips can be checked by OEM
  - Can show some end-customer indication also, for example signature on code locked during the OEM lifecycle
  - Many architectures possible, beyond the scope of this presentation.

OEM – Original Equipment Manufacturer, typically said of chip customer who assembles a system.

WLCSP – Wafer Level Chip Scale Package, a package technique of placing bumps on top of the die and mounting is upside-down on the PCB.

- At least one project plans to use a WLCSP package
  - backside of die exposed, even in-situ
  - scannable at about 1um resolution using tools that are not prohibitively expensive.
  - Bunnie's blog [R28] provides a lot of detail on various scanning options


1050nm IR emitter
CMOS digital camera with no IR cut-off filter
Light reflected/scattered off of metal routing near the transistor layer
Bulk silicon of chip to inspect
Transistors
Routing metal
Passivation layer
Solder balls
Circuit board
(vertical scale compressed)

Summary

- Open Source software is a developing field with many controversies.
  - Open Source board-level hardware is a nascent field in comparison
    - and Open Source ASIC is even less mature/defined than that.
- Even the fundamentals are unclear, including:
  - legal foundation
  - community goals
    - Would welcome feedback in particular on “community goals”.
- Even if fundamentals were clear, there is considerable complexity in implementation, given ASIC design flow.
- Nevertheless, we try to take some tangible step to at least advance the state of the field.

REFERENCES (1)

[R1] Ackermann article (effectively, the TAPR rationale) https://www.jrackermann.com/docs/Ackermann_Open_Source_Hardware_Article_2009.pdf

[R2] CERN rationale https://ohwr.org/project/cernohl/wikis/uploads/0be6f561d2b4a686c5765c74be32daf9/CERN_OHL_rationale.pdf

[R3] Eli Grenbaum/Harvard law http://jolt.law.harvard.edu/articles/pdf/v25/25HarvJLTech131.pdf

[R4] Andrew Katz article (effectively , the Solderpad rationale, but discussed other licenses in detail also) https://www.jolts.world/index.php/jolts/article/view/69/131

[R5] Monton et al, article from Autonomous University of Barcelona https://arxiv.org/pdf/2010.09039.pdf

[R6] Timothy Murphy, University of Idaho Law article (contains IPIL license) https://digitalcommons.law.uidaho.edu/cgi/viewcontent.cgi?article=1412&context=faculty_scholarship

[R7] Summary and analysis of Google v Oracle was about use of Java SE APIs; https://www.oyez.org/cases/2020/18-956

[R8] Apple v Franklin, Golden Gate Law Review article https://digitalcommons.law.ggu.edu/cgi/viewcontent.cgi?article=1344

[R9] The Free Software Foundation's view on copyleft
https://www.gnu.org/copyleft, or alternatively https://perma.cc/2BHV-LG4T

[R10] Listing and description of some prominent GPL case law (software)
https://wiki.fsfe.org/Migrated/GPL%20Enforcement%20Cases

[R11] GPLv3 case law in China, https://www.mclibre.org/descargar/docs/revistas/ifosslr/ifosslr-10-1-en-201812.pdf

[R12] Nimmer on copyright: MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT; describes several case law precedents for a building to be a derivative work of an architectural plan.

[R13] Article and analysis for United States Semiconductor Chip Protection Act of 1984 ("SCPA").
https://ir.lawnet.fordham.edu/cgi/viewcontent.cgi?article=1041&context=iplj

[R14] WTO Agreement on Trade Related aspects of Intellectual Property Rights (TRIPS) https://www.wto.org/english/docs_e/legal_e/27-trips_01_e.htm

[R15] Copyleft.org view on GPL https://copyleft.org/guide/monolithic/

REFERENCES (2)

[R16] Collection of resources and opinions on the topic of Open Hardward Licenses https://wiki.p2pfoundation.net/Open_Hardware_Licenses

[R17] https://opencores.org/ Example of a OSS hardware repository. Note that many use SW licenses, including copyleft licenses.

[R18] https://www.fossi-foundation.org/licensing - Free and Open Source Silicon Foundation, collection of open-source projects.

[R19] Open Source Hardware Association https://www.oshwa.org/

[R20] Open Source Initiative: https://opensource.org/

[R21] Free Software Foundation: https://www.fsf.org/

[R22] An open source VHDL simulator http://freehdl.seul.org/

[R23] Picker vs Imaging Equipment Services https://law.justia.com/cases/federal/district-courts/FSupp/931/18/2346839/

[R24] Sheppard Mullin blog on Maskwork https://www.intellectualpropertylawblog.com/archives/protecting-semiconductor-chip-design-under-the-semiconductor-chip-protection-act-of-1984-scpa-part-i-registration-and-inspection/

[R25] OHANDA An early organization in this field recently defunct, formerly http://www.ohanda.org/ see further detail at https://en.wikipedia.org/wiki/Open_Hardware_and_Design_Alliance

[R26] An analysis article on Artifex Software, inc. v Hancom, Inc., 2017 https://www.synopsys.com/blogs/software-security/breach-gpl-license-breach-contract/

[R27] SFC v. Vizio article, https://fossa.com/blog/massive-implications-software-freedom-conservancy-vs-vizio/

[R28] Article on chip inspection, https://www.bunniestudios.com/blog/?p=6712

[R29] Gnu.org discussion of what constitutes aggregation that does not trigger copyleft, https://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#MereAggregation

[R30] VexRiscv on github https://github.com/SpinalHDL/VexRiscv/blob/master/LICENSE

LICENSES

- [L0] This list does not attempt to list the many SW licenses that are often used for open source ASIC RTL (appropriately or not). A comprehensive list of SW licenses can be found in [R20]
- [L1] SolderPad http://solderpad.org/licenses/
- [L2] TAPR https://tapr.org/the-tapr-open-hardware-license/
- [L3] CERN-OHL-W, -L, and –S https://cern-ohl.web.cern.ch/
- [L4] Creative commons https://creativecommons.org/share-your-work/public-domain/cc0/
- [L5] IPIL – IP Instantiation License, this is contained in the paper by Timothy Murphy cited in Reference [6] above. https://digitalcommons.law.uidaho.edu/cgi/viewcontent.cgi?article=1412&context=faculty_scholarship
- [L6] Chumby developers agreement originally listed at lost site:
  - http://www.chumby.com/developers/agreement
  - But some discussion can be found at:
  - https://www.bunniestudios.com/blog/?p=118
- [L7] OHDL http://juliusbaxter.net/ohdl/

Thank You

info@cramiumlabs.com