Design Sharing for HEP IC Development

Carl Grace

Lawrence Berkeley National Laboratory

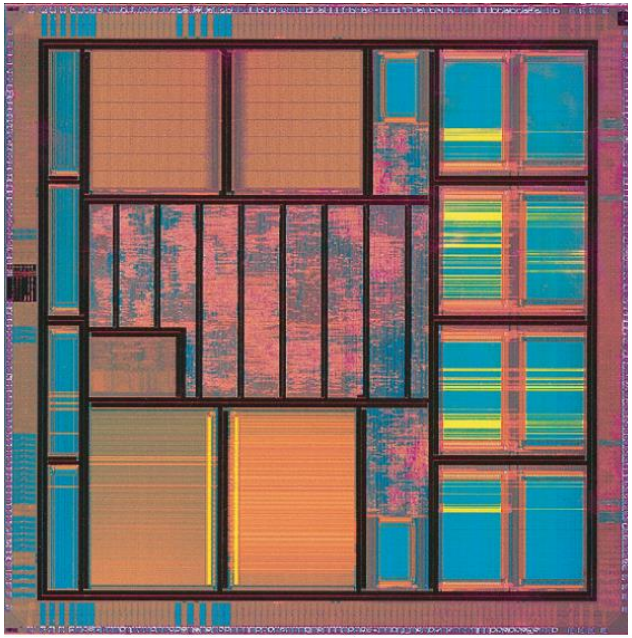
October 6, 2017

A brief comment on definitions…

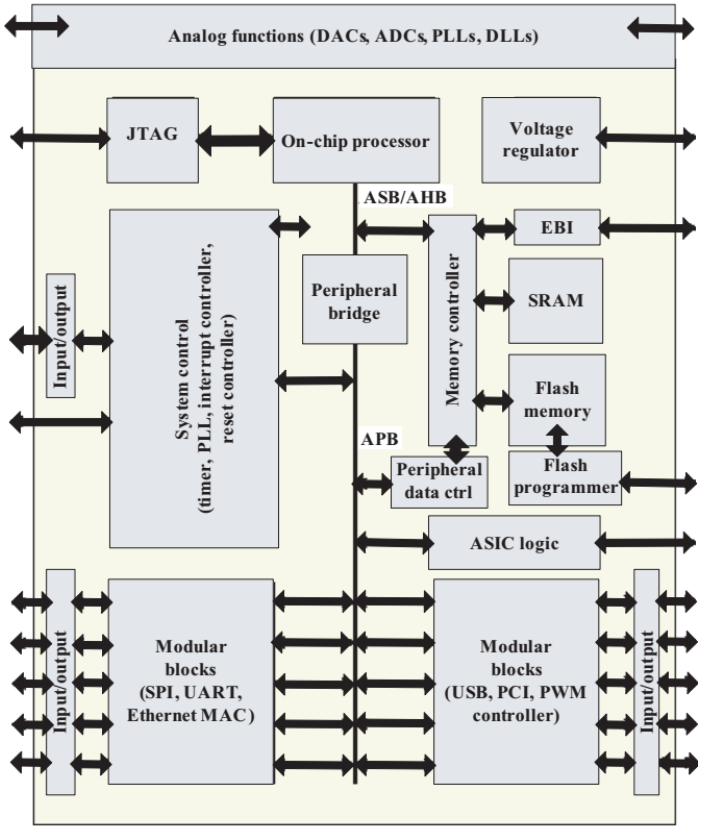
- Intellectual Property (IP) is an unfortunate name for "semiconductor design data"
- The term IP has specific meanings to legal folks that may or may not apply to the design data under consideration in a specific case
- IP is the standard term in the industry, so when I use it I mean "design data"
- I have struggled with this personally, trying to get our tech transfer office to understand exactly what it was I wanted to license from ARM…

The IP Approach to SoC Design

The complexity of modern ICs (especially SoCs) has driven the semiconductor industry more to an Intellectual Property model, where a complex system chip is largely an aggregation of third-party blocks, and the differentiation is primarily in the software (or sometimes the front end).



Typical SoC



Problems with SoC approach

1. Complexity (notice the similarity to why IC Design is getting harder…)

   Industrial SoCs are incredibly complex, and require integration of hard and soft IP for memories, embedded processors, bus logic, analog, power management, clocking, etc., etc.

2. Cost

   Dealing with such complexity requires a lot of staff and a lot of money. Each in the institution HEP community cannot possibly independently develop all the needed IP, and integration would cause verification problems that are probably not solvable with our resources.

3. Fitness-to-Purpose

   In HEP the value is typically added in the front end and we usually don't have the same pressures to integrate lots of different peripheral blocks (sometimes we do though).

“IP-lite approach”

- Often the same kinds of smaller blocks show up in our chips in HEP (and in other DOE-related applications)
  - Bandgaps, ADCs, DACs, serializers, PLLs, configuration RTL blocks, memories, etc
- These can be taken from old projects on an ad-hoc basis
  - Cross-project sharing is mostly confined to single institutions
- What are the barriers to cross institution sharing?
- Can we come up with something useful that will extend our reach?
- Can’t really use design sharing to differentiate

Soft IP sharing

- Easiest target is synthesizable RTL
- Could share silicon-proven small RTL blocks (e.g. I2C or SPI interfaces) or larger subsystems (e.g. JESD204B TX protocol) in git repository
- Potentially useful because RTL is process-agnostic
- Would the effort to make a block reusable (documentation/generalization) be worth it?
  - And who exactly would pay for it?
- Are there licensing issues here (especially with the national labs)?

Analog Design Sharing

- Why is Analog Design sharing so hard?
- Analog designs are tightly tied to a given process (even at same node)
  - Last year I ported an LVDS receiver from one 180 nm process to another and had to redesign the common-mode feedback because it oscillated in the new process! Astounding!
- We in HEP primarily get our value from analog performance and the power/performance tradeoff can be brutal (almost impossible to reuse a charge amp, for instance)
- In industry, SoCs typically have relaxed specs for analog and companies care mostly about cost & software (not us!)

Analog Design Sharing

- Analog design sharing can work well in two cases:

  1. Noncritical functionality that is silicon proven

  2. Functionality is so critical that ASIC is “built-around” the IP (e.g. many of LBNL’s imagers use the same pixel and change the periphery circuits for differentiation)

- Designing around core functionality leads to Platform-Based Design which is another important trend in industry

- With Platforms, we focus only one what adds value (usually the low-noise front end or pixel circuit), and plug in everything else we need to turn it into a system

IC Development Infrastructure (just add money!)

Software Infrastructure

- Design entry (schematic and physical layout design)
- Simulation (analog, digital, mixed-mode)
- Synthesis
- Automatic Place-and-Route
- Static Timing Analysis / Formal verification
- DRC/LVS verification
- FPGA firmware development environment
- Board development suite
- Test framework, instrument control
- Design-space exploration (MATLAB or similar)

Required Team Competencies

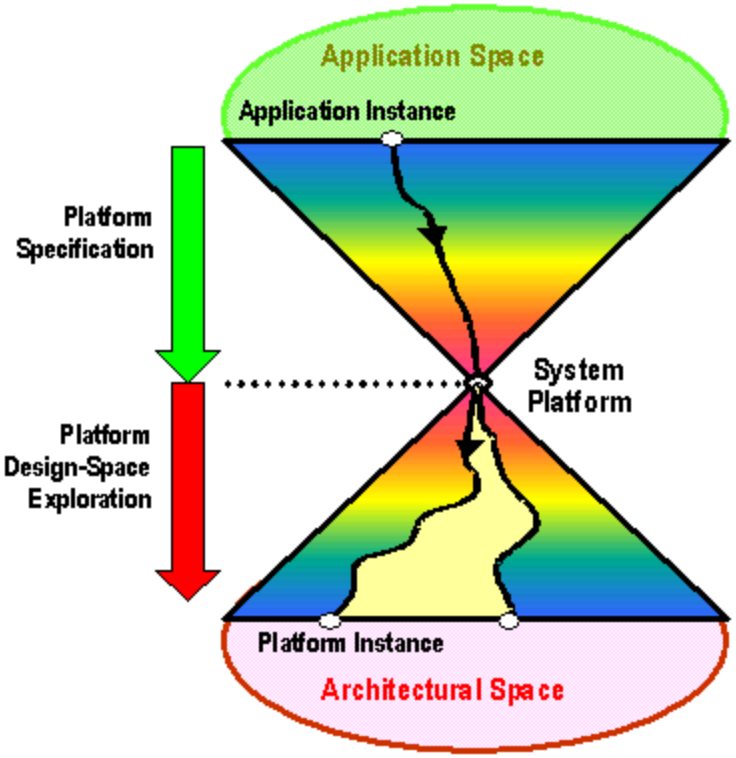
- Transistor-level analog and mixed-signal design
- Digital RTL development
- Physical Design and Verification
- System-level Validation
- Analog/Digital co-simulation
- Behavioral Modeling
- Project management
- Board-level circuit design
- FPGA firmware development
- Teststand software development
- Advanced test execution and debug

Only the largest organizations have the resources to do a system-level IC end-to-end alone.

Can smaller organizations specialize to provide a joint, cross-design group capability?

Platform-Based Design

- Most readout systems look broadly similar
- A platform can embody these commonalities
- Individual readout ICs are instances of the common platform
- Dramatic improvements in design productivity and tractability
- Enables small teams to complete projects that would be impossible using an ad-hoc approach



Sow once → Reap many times

Each new chip is a platform instance instead of a scratch design

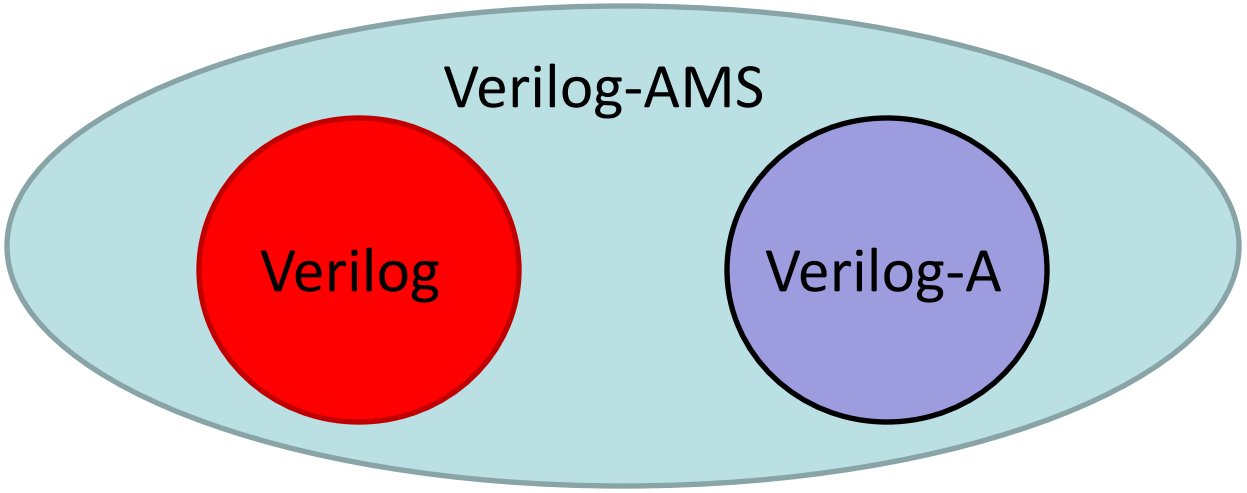
A. Sangiovanni Vincentelli, UC Berkeley

Leads directly to improved top-down design methodologies

Platform-Based Design

- Platform is an integrated system designed for modification and extensibility
- Choose flexible macros for reuse
- Process, block interfaces, and characteristics standardized
  - e.g. 65nm CMOS, pitch matching, electrical interfaces, biasing requirements
- Platform includes set of pin-accurate functional models in Verilog-AMS
  - Models allow rapid development of platform instances

HARD to agree on what the platform should be!



Verilog-AMS allows full system simulation (analog + digital)

Enables digital-centric design approach → lower cost and higher performance

Successes with Platform-Based Design

- SLAC has been very successful with their ePix/tPix/etc platforms. We can (and should) all learn from this.
- FNAL's test platform development great speeds development and saves a lot of money
- Similarly, LBNL's imager platform shortens design time and simplifies evaluation and camera development, but our platform isn't as developed as SLAC's
- Other platforms in the community go a long way to helping us punch above our weight (e.g. pixel chips, etc.)

Key Issues with Platform-Based Design

- Can we identify a platform that would be useful?

    1. Is there enough commonality in our designs?

- How do we fund platform development?
    1. May be natural to take an existing part as first iteration, but then how do we “compensate” the lead institution?
- Would the handcuffs of Platforms hurt more than help?

Summary

- With chips getting more complex, an IP approach is attractive
- We probably shouldn't follow industry too far down the SoC rabbit hole
- We have had success in HEP sharing within a project/collab
- There are significant barriers (e.g. licensing, tool usage, effort) to doing even the simplest cross-project sharing
- There has been success with platform-based design (which is a kind of sharing). We should be doing this as much as makes sense.

Other Perspectives