Platform-Based Design: An Emerging Reality

Bob Altizer
BASYS Consulting
VSIA Platform-Based Design Development Working Group
bob.altizer@basysconsulting.com

ABSTRACT: Defining a reusable hardware-software platform as the basis for embedded system-on-chip (SoC) products is an idea with considerable value. The VSIA's Platform-Based Design (PBD) Development Working Group has been studying the key issues underlying PBD, and recently released for member review the first industry-standard definitions of an SoC platform and the platform-based design process, and a fundamental taxonomy of platforms in the embedded space. This session examines some of the key definitions and issues for PBD practitioners.

Why worry about PBD? We think it can have a big impact for several reasons: Platforms are the reuse paradigm for the next level of issues the embedded systems industry is facing, with a big impact on IP interoperability and time-to-market. They're also the catalyst for systematic reuse and producing families of derivative products. Effective use of platforms throughout the supply and design chain based on fundamental agreements on foundations and definitions enables better interaction between suppliers and customers. PBD is also an integration thing: using platforms means system specification and the ability to integrate reusable IP have a greater impact on project success than traditional design skills. Finally, VSIA members have told us they have more to gain by collaborating on PBD, than by keeping all their work proprietary.

But perhaps the biggest impact of PBD will be economic. A *product family* based on a reusable platform will pay off in may ways, most importantly with greater net economic return across the family than you'd see from traditional, one-at-a-time product development. Using a PBD-based product development process helps you better target products to markets, and your investments are focused on developing reusable core assets from which you can derive long-term value.

Within the PBD working group, we started by considering a number of our members' points of view, including one called "SoC Platform Postulates." The first postulate is pretty straightforward: You can, in fact, construct a platform that serves as the basis for multiple derivative products. Second, successful PBD depends on several constituents: the platforms themselves, the market-specific IP to be integrated with them; a design flow that supports that integration; and an array of tool, application, and systems support. It's not necessary to have all these fully formed before starting a PBD program, but the more of them you have, the better off you'll be. Third, PBD will contribute to economic advantage in a number of ways. Even if you benefit from only one or two of these,

you're likely to be much better off. And finally, we presume that the economic advantages will be enough to justify the investment in PBD, which can be large.

One of our first tasks as the PBD Working Group was to come up with an agreed-upon set of terms. Fortunately, there are plenty of opinions about platforms to choose from in the SoC industry, including abstractions mapping layers to one another, a reuse mix-and-match environment for an application domain, a collection of assets to leverage reuse and development of new products, and an architectural framework for rapid integration. Important books by Henry Chang, et al. [1] and Grant Martin and Henry Chang [2], as well as numerous other references are good sources for this discussion.

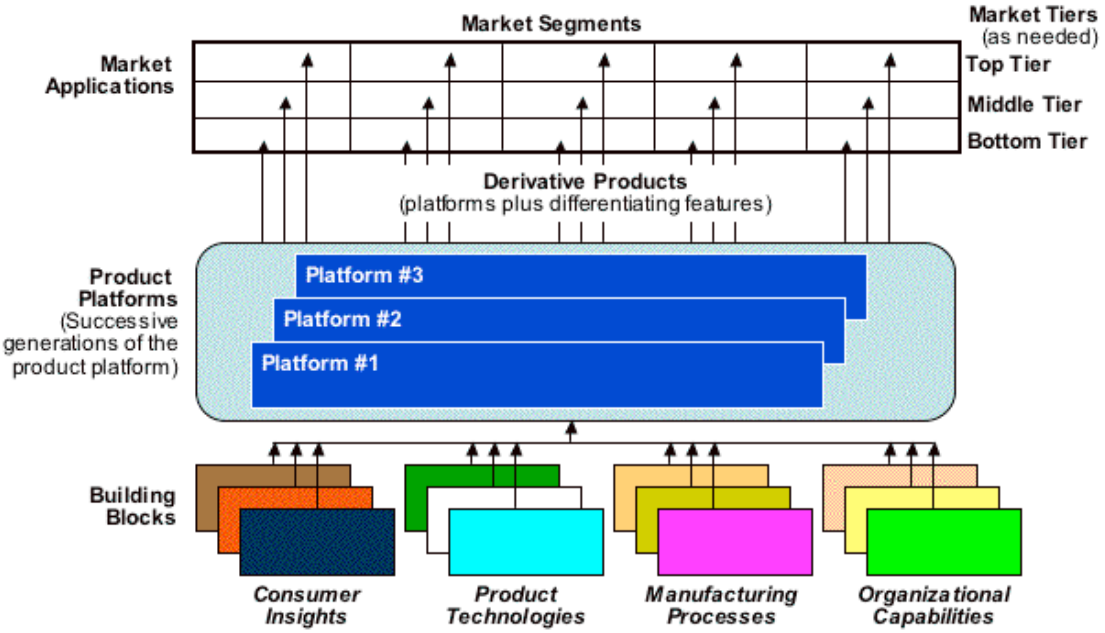

Figure 1: The Platform "Power Tower"

Business schools also study PBD. Figure 1 shows the "Platform Power Tower" as presented by Marc Meyer and Alvin Lehnerd of Northeastern University [3]. To start, you aim to put products into certain market segments, the columns in the array; you may define top-mid-and-bottom tiers as well, the rows. Your product family building blocks cover your whole business: consumer insights – their needs and requirements – as well as your technology, manufacturing, and organizational capabilities. Based on these building blocks, you can create one or more platforms to serve the common needs of multiple market segments and/or tiers. Finally, you create sets of derivative products from the platforms. You may not try to hit every cell in the market array at once. How you fill it, segment-by-segment, tier-by-tier, or whatever, depends on your customers' needs and your business goals. When all the elements of the "Power Tower" are in place, you have a foundation for wide and deep market penetration in your business, with platforms capable of supporting several generations to meet evolving needs.

In the PBD Definition and Taxonomy document we just released for VSIA member review [4], we concentrated on resolving the many notions, points of view, and personal definitions in the world of platform-based design at the SoC boundary and below.

First**,** we have standardized definitions of platforms and PBD. A *platform* is an integrated and managed set of common features used for building a set of derivative products. This may not satisfy some who wanted to see a down-to-the-cell specification of "the" SoC platform, but it's a very useful definition. For one thing, if what you're building is not intended as the basis for a set of derivatives, it's not a platform! *Platform based design* is an integration oriented design approach emphasizing systematic reuse. Here it's important to recognize that integration and reuse take higher priority in the design flow than new, special, custom, or even "optimized" design, a new notion to many designers and design managers.

We also defined two basic taxonomies, or classification systems. First, the *Platform Specification Approach* is the way the platform development team develops the platform architecture they need, based on their business needs and technical capabilities. We identified three basic approaches to platform specification that product family planners and platform architects can use to specify their desired platform and its integration environment, within the context of their business goals, overall product family strategy, technology capability, and other constraints. The second taxonomy, *Platform Object Complexity,* describes the contents of a deliverable platform at a particular integration level as the sum of the constituents mentioned earlier. The product of the specification process is the deliverable platform object at an appropriate level of complexity.

Application-driven platform specification is a top-down approach, using system-level design methods and focusing on the functional requirements of family of products (set of derivatives) to be built on the platform. At this level there should be a product family-driven process for creating a platform and the set of derivative products based upon it. Application-driven platforms are agnostic with respect to the fundamental software and semiconductor technologies used to build the SoC, though the underlying technology must offer suitable application "performance plateaus," that is, enough performance to meet application requirements. Platform developers don't have to rely just on existing components; they can create new ones – cores, interfaces, peripherals, and so on – as needed.

Architecture-driven platform specification is a middle-out approach; a system-level approach with restrictions derived from the relevant family of existing cores, core support packages, memory structures, and on-chip communications standards. They have loose coupling to both applications and technologies, a middle out methodology in that a pre-defined kernel containing the basic architecture already exists, and is used along with pre-verified component IP to create a derivative.

Technology-driven platform specification is a bottom-up approach, based on traditional design methods (standard cell, full custom, library-oriented, etc.) with the latest and potentially highest performance (and costliest) semiconductor technology processes. They are agnostic with respect to applications, though applications with higher performance and/or integration needs will likely be the first users of newest technology-driven platforms.

Determining which specification approach to use is where the details of the specification taxonomy come into play. In large part it will be dictated by your overall business philosophy: how much you focus on technology, how much you focus on end-products, and so on. We've identified a number of attributes of each specification approach, and in the body of our just-published Definitions & Taxonomy document, we've described how each attribute applies to each of the specification approaches. In principle, you should be able to look at how the attributes align with your business approach, and find the one platform specification approach best for you. We know it won't always be that easy in practice, but we do expect you'll get a good indication of which approach to use.

Now let's look at the *platform objects* themselves, and elaborate on the multiple constituents of a complete platform-based design system, which are important to a successful PBD program:

- *Components and features* are the traditional – and to some the only – elements, the bare minimum from an engineering point of view. Traditional hardware and system level components appear here, but also required are operating and application software, software APIs, and programming models.
- *Business and economics* includes the business plans and economic models to support a platform-based product family, including economic scoping, roadmaps, and full life cycle plans, as well as IP licensing and royalty plans.
- *Development and integration tools* support the integration-based design flow, and include both hardware and software tools, reference designs, and expert-level application and system engineering support.
- *Overall platform and product line support* includes program management; good relations between platform developer, IP developers, and the integrator-customer; change and fix management; promotion and marketing of platform capabilities and success stories; and more.

If you don't have all these constituents 100% in place, platform-based efforts won't necessarily fail. But the more of them you have, the more likely it is you'll succeed.

We define platform object complexity as a set of *complexity levels*, covering stages of integration between simple IP blocks used in block-based design, and complete, physical SoC devices ready for delivery and integration into a customer's higher level product. Each complexity level describes deliverables in each of the major platform constituents.

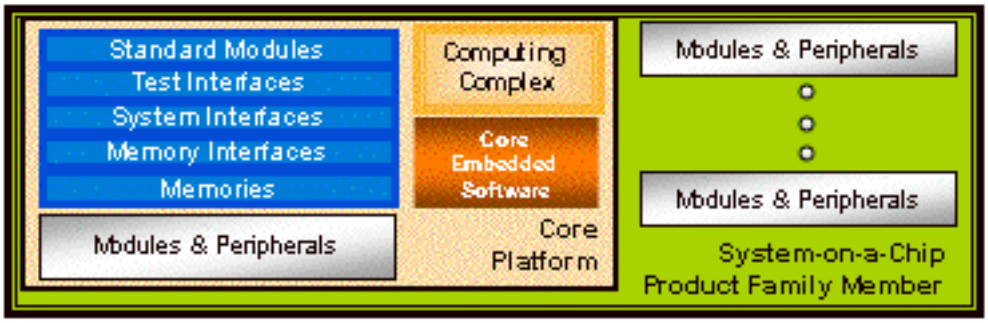

Figure 2: SoC Platform Objects

Figure 2 illustrates various levels of platform objects:

- The lowest complexity level, *Set of Blocks*, is not really a platform at all, but the result of traditional block-based design, where individual blocks or pieces of integrateable IP are linked to create a usable device.
- The next level, *Core Platform*, is a true platform at the sub-SoC level, integrating computing capability, some peripherals, and some software, not necessarily tailored for a particular application domain. Core platforms are used where the platform specification process identifies an economic value at this lower level of integration, either as a component of a higher level platform, or of a customer's product. Core platforms may exist as a set of integrateable soft models without any physical implementation.
- The highest complexity level, *SoC Platform*, is a complete, fully integrated and physically deliverable device, which may be in the form of a traditional single package, a flip-chip, multiple modules, or even a hard macro. SoC platforms are defined when the product specification process identifies economic value at this level of integration. They may be derivatives of core platforms, by integrating domain specific functionality, design tool, and support with a lower level core platform.

Each complexity level limits the number of attributes that are required to transfer between a platform provider and its integrator-user. We should note that "Complexity Levels" are similar to "integration levels"; we use the new term to describe the specific sets of delivered elements. Furthermore, a platform at a given complexity level can be considered as a virtual component by a platform integrator working at a higher level.

So the PBD process in the SoC business as we see it is straightforward. Common features supporting multiple higher-level products are aggregated into a platform, then integrated with product, application, or domain-specific variable features to form one or more derivative products. A product family of multiple derivatives can be developed simultaneously, or derivatives can succeed each other over time. How the platform integrators define their product families depends on their market, product roadmap, and business philosophy.

We also have some stipulations to the PBD process:

- Each product is the derivative of the platform; not of other products;
- Once specified, a platform is a black box that may be configured but not changed by the user. If a given platform really needs changing, a new generation platform should replace it.
- The principles of PBD can be used at any level of abstraction or construction in building a system.

Recalling our initial ideas about platforms being the catalyst both for IP reuse and better economic return, here are some of the things we'd expect to see a "best performing" platform shops:

- They'll have a *product family definition* that's market or domain based, recognizing the different market segments and tiers to be served by derivative products. This is strongly tied to how the customer – the platform integrator – will use a platform to build his products.
- They'll do *economic scoping* to determine which common features are best included in the platform, and which differentiating features are best kept outside. This is strongly dependent on the design and manufacturing capacity of the platform provider.
- They'll have a formal process for *platform architecture specification*, so the platform is well described, and the rationale behind the design decisions are well known. In addition, the platform roadmap will plan the evolution and migration of the platform over the lifetime of the whole product family it support.
- And finally, they'll have a *platform-based development process* that emphasizes design both for and with reuse, and integration of reusable IP into desirable products. This may require restructuring their organization from one based on a traditional one-at-a-time product approach, but it will be worth it.

In conclusion, lets look at some of the key questions around PBD, and why we think it's an "emerging reality." First, we think we've answered the question "what are the basic platform types?", and given a set of definitions the industry can use to provide a common language in the ongoing discussion about PBD.

How much of the SoC Market will be platform-based? A significant fraction, but not all. PBD will never replace the need for optimal, high-performance design; custom design will always have a significant part of the market. In the near term, we'd expect to see a third or more of the market start to benefit from full-fledged PBD approach.

What methods and tools do we need for effective design and reuse of platforms? Traditional design methods will still predominate, as the most important contribution of PBD as we've shown it here will be in product line planning and platform specification. But some key work in describing IP for automated integration is under way, such as the SPIRIT consortium, many of whose members are part of our PBD working group.

What can the SoC industry learn from research in software intensive systems? Two words: “a lot!” Basic approaches for economic scoping, product family evolution, and manufacturing and organizational issues supporting PBD has been done. There’s plenty to learn outside the SoC technology realm as well.

How can industry come together? Obviously, we’d like all of you to join in our work in the VSIA PBD working group, as we begin a new phase focusing on using our definitions and taxonomies, and focus on deliverables from platform providers to user-integrators, and let us know your interests, needs, and concerns.

We hope this overview of our working group’s progress gives you confidence that we’re on track for success in making the technical and business success of PBD in SoC a reality.

REFERENCES

[1] Henry Chang, Larry Cooke, Merrill Hunt, Grant Martin, Andrew McNelly, and Lee Todd, **Surviving the SOC Revolution: A Guide To Platform-Based Design**, Boston, MA, Kluwer Academic Publishers, 1999

[2] Grant Martin and Henry Chang, editors, **Winning the SOC Revolution: Experiences in Real Design**, Norwell, MA, Kluwer Academic Publishers, 2003

[3] Marc H. Meyer and Alvin P. Lehnerd, **The Power of Product Platforms: Building Value and Cost Leadership**, Free Press, 1997

[4] VSIA Platform-Based Design DWG, **PBD Definitions and Taxonomy Document**, **Version 1.00** (PBD 1 1.0), September, 2003