The Secret to Building IP

John G. Maneatis, Ph.D.
President, True Circuits, Inc.
December 1, 2016

The secret to building IP is maximizing reuse, which affects every part of your development process, from architectural choices to circuit design to CAD tools and flows, and even in the personnel you hire. Customers want low risk designs customized to their specific needs while IP suppliers want to develop designs once and amortize their investment. Satisfying demanding customers requires building highly flexible and programmable IP that can be delivered in all common process variants and metal stacks. By using a high degree of automation, you can recover your costs and invest in new R&D. This paper will focus on how True Circuits has been able to successfully maximize our reuse at every level.

Maximizing reuse begins with designing standard and highly configurable IP. Standardization is key and needs to be applied to everything, from circuits and layout to simulations and testing to datasheets and documentation. But standardization itself is not enough. You need to be able to represent similar things through their differences, with an easy way to generate all versions. At True Circuits, we use a single, universal design database for everything. Similar things are only described through their differences from standard templates and only in one place. We use parameters to control these differences and custom CAD tools to build everything from these templates. This approach allows us to target any design version, in any process. All of the simulation and quality assurance that we do helps insure that all designs are robust. Designs are customized from standard templates by adding parameters that control the differences and create the final versions.

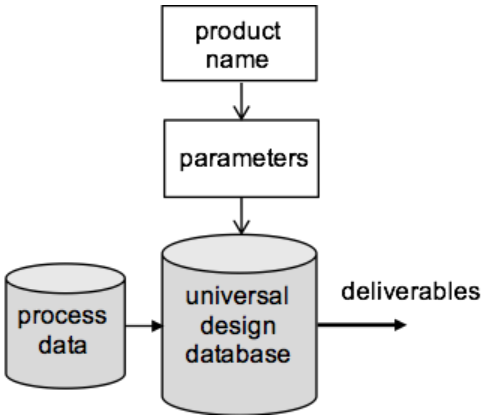
To make this standardization work, we start with robust and flexible circuits with automated tuning that are easily re-targeted to different processes and applications. Next, we employ a highly integrated CAD environment that automates all of the tasks that can be automated. It is important that this CAD environment is integrated across the entire range of tools. We have found that it is necessary to create our own tools and tool customizations in order to create a flexible, integrated environment. Industry-standard,

third-party tools tend to be point solutions, giving access to limited automation that typically stops short of solving a complete problem. At True Circuits, our simulation, layout, and lab characterization flows, among others, all use one, integrated CAD environment. Everything is driven by parameters that modify standard templates.

To make such an automated approach work, you need to hire versatile designers with strong skills in programming and algorithm development. All of the easy and repeated design tasks are automated using this approach, leaving behind the more difficult tasks. If you have the right people, you can get an order of magnitude increase in their productivity.

A custom CAD environment is also very important to achieving reuse. You have to have a passion for CAD and not be afraid of writing any tool that you need. It is easier than you think because you only need to support the features that you intend to use. If you want to support a lot of designs and processes, you need CAD tools that are customized to facilitate their creation. You want to get to the point of being able to create the design kit simply from the product name. Our custom CAD environment describes, specifies, models and simulates designs, and creates front and back-end views, package designs, documentation, lab tests and silicon reports. We use parameters to drive everything including all tools and flows, circuit specifications, simulation setup, lab characterization, models, documentation and design kit assembly.

The typical flow begins with a product name and generates all standard parameters that apply to a given design. These parameters get combined with process data and applied to the universal design database to produce the customer IP deliverables for that design. Because QA is automated and built into the flows, and the designs are all related, the IP is correct by construction.



The database descriptions and flows used to build designs must be tightly coupled. We embed the C Programming Language in all description languages to support parameterization, minimize descriptions for repetitive items and maximize reuse of elements. We use C because it is compiled and very fast. We can create compiled libraries and keep everything fast and efficient. Using our netlisting language, designs can be very succinctly described. All tools and flows can be run from the command line with parameters as needed. Some flows are built on top of other flows to allow complete designs to be created from just the product name.

Consider the circuit description for simple ring oscillator. Not only can sizing be parameterized, the structure can be parameterized as well. In this case the number of stages in the ring is programmable. The coding for this ring oscillator is as follows:

```
xr1 vo[num:1] vdd vss ring(size)

.subckt ring(s=1) vo[n:1] vdd vss
:int i, j;
:for (i = 1, j = n; i <= n; j = i++) {
     xi[i] vo[j] vo[i] vdd vss inv(s)
:}
.ends ring
```

The foundation from which IP is constructed must be robust. We use a rich library of process-insensitive circuits to achieve adaptive bandwidth, self-training and matched timing by construction. The IP is designed to operate over wide ranges and is pin programmable to achieve customer flexibility. This approach helps limit the number of versions of designs that are needed and allows customers to pass on this flexibility to their end customers.

Standardization is achieved with a "universal design database," but it is not necessarily one database, as different types of items may be represented in different databases or generation tools. The key is to avoid repetition in the items described. For example, the same generator may be used to create the specifications for a design, no matter where they may appear, including design kits, product collateral or web pages. The look and feel of these items may be different, but all are controlled by standard parameters. Also, the pin list that appears in the front-end models and in many views is specified in a single place. This approach makes adding features easy and avoids inconsistencies. All

information should be included so designers do not need to “remember” anything about the designs.

For specifying circuits, we use a universal design database that contains basic gates, programmable datapath blocks (adders, multipliers, shifters, etc.), programmable higher-level structures (fractional dividers, etc.) all of the way up to complete IP blocks. The database is process independent and different process versions can be created with all needed structural options simply by setting the process number parameter. Customization can usually be done with standard options or with reusable sub-blocks, ultimately all controlled by parameters. Making a copy of the database to implement a change is not allowed—the change must be incorporated into the universal design database. Using this approach, we can generate any version of any design ever shipped to a customer. The database also codes how to simulate the designs with specific support for the features selected. This approach of maintaining a single database for all design versions takes discipline, but it makes it easy to support a large number of designs, allowing improvements and added features to be shared by all designs.

Custom CAD tools are an essential part of the reuse strategy. The only externally-licensed tools that TCI uses are for back-end verification. Everything else is internally developed.

The front-end side starts with the circuit database previously discussed. This database is manipulated with a powerful netlist processor, which supports some logic synthesis. It can generate many views, including circuit logic, timing, characterization and layout placement. The characterization environment is parameter driven. It is composed of several layers for netlisting, running simulations, performing measurements and reducing data. Circuit libraries specify a parameter-driven characterization template that automatically creates all of the simulation files needed to characterize the design or design library. It is powerful enough to code any measurement that can be imagined. We utilize the large number of CPUs in our server farm to rapidly and fully characterize designs, reducing the resulting data down to key metrics for easy analysis. We also generate characterization reports for customers. The complete characterization is commonly started with a single command.

Inside the characterization environment are a few point tools for running simulations and performing timing verification. The simulation tool supports mixed mode simulations, delay modeling with back annotated parasitics, transient noise analysis, and supports all

vendor model formats (TMI, Simkit.) The timing verification tool supports statistical timing analysis, arbitrary clock domains, structural checks, interactive environment, and automatic characterization of cell libraries. By using proprietary tools, we can make them do exactly what we want. All of these tools are tightly coupled and support feedback to the designs.

The back-end side employs similar automation to the front-end side. We draw layout using a Berkeley Magic-based layout editor with powerful additions. It is best to think of it as the "vi" or "emacs" of a software project relative to the back-end flows. Our netlist tool supports directed synthesis to allow equations of signals and buses. We have a custom place and route tool designed specifically for high-speed analog and digital designs. It uses placement hints in the netlist and generates estimated parasitics. The placement and routing is repeatable, and independent of technology. We use a third generation layout remapping tool that allows GDSII structures to be mapped to a different set of design rules. It can start with layout drawn in any set of design rules and then port them to a new target process. We use global routing tools that can route signals between blocks on a chip based on the top level chip netlist. Finally, we use a set of layout manipulation tools for block assembly. These tools are organized in a parameter driven flow ultimately driven by the block name. They perform logic operations on layers, create special layers, perform options and re-sizing, and generate structures (equivalent to P-cells, etc.).

The lab testing environment follows suit with the other flows. The environment is fully automated. It is setup to drive all equipment (DUT, sources, scopes, and environment.) The environment is controlled by scripts that are parameter-driven from chip ID. It can be used interactively or in batch mode. It can be run remotely. The lab characterization software leverages the simulation environment with a common data representation format and common characterization functions (e.g., measuring jitter). It is setup to generate test reports automatically.

The goal of the database representation and custom tool and flows is to make maintenance of designs easy. It is common knowledge that support time and number of bugs will be proportional to the number of lines, polygons, etc. required to specify a design. By representing designs using parameters to control differences, a large number of designs can be represented using a small number of descriptive elements, reducing the support cost and risk of a problem. Because knowledge of designs is folded into the databases and tools, it is very easy for designers to perform operations on the design

without specific knowledge of the particular design version. For example, a single command will run all simulations for a design in a new process. The unified design representation makes it easy to incorporate fixes, improvements, etc. on all related designs.

In conclusion, the goal in IP development is to allow customers to get what they want with reduced risk and design time, while still allowing the IP supplier to amortize design investments. This goal is accomplished by maximizing reuse with a custom CAD environment and robust and flexible circuits that incorporates design and characterization information and make maintenance easier with the use of unified design specifications. The challenge is that you need designers who can control the tools and design at the margin. If you can find the right people, they will see a significant productivity boost and you can manage a large IP portfolio with a much smaller staff.

John Maneatis received the B.S. degree in Electrical Engineering and Computer Science from the University of California, Berkeley, in 1988 and the M.S. and Ph.D. degrees in Electrical Engineering from Stanford University, Stanford, CA, in 1989 and 1994. From 1994 to 1998, he was a lead circuit designer at Silicon Graphics, Inc., Mountain View, CA. In 1998 he founded True Circuits, Inc., Los Altos, CA., where he is currently the President and Chief Technologist, working in the areas of Phase-Locked Loops, high-speed mixed-signal design, design automation, and CAD software. True Circuits, Inc. is a leading provider of high-speed mixed-signal analog IP. It was founded 1998 with headquarters in Los Altos. True Circuits builds easy to use, highly programmable hard macros, like Phase-Locked Loops, Delay-Locked Loops, and DDR PHYs, in most foundries and process variants. True Circuits has an experienced mixed-signal design team and close technical relationships with major foundries. Its customers include most semiconductor suppliers and design service providers. Over the 18 years that it has been in business, True Circuits has shipped countless IP blocks used on billions of chips in about 100 process variants.

4300 El Camino Real, Suite 200, Los Altos, CA 94022
www.truecircuits.com