

IC Development Capability— What to Measure?

White Paper

Developed by
Numetrics Management Systems, Inc.
20863 Stevens Creek Blvd., Suite 510 Cupertino, CA 95014
Tel: (408) 351 5800 Fax: (408) 351 5850
Email: info@numetrics.com
Web site: www.numetrics.com

INTRODUCTION

As development capability is now a primary differentiator for semiconductor companies, its health must be benchmarked and monitored vigilantly. These tasks are simply no longer optional. The effectiveness of a company's engineering organization and overall product development process is becoming the lynchpin of competitive advantage in the semiconductor industry.

Developing a measurement framework of key performance indicators is the first step in calibrating development capability. The basic challenge, though, is what to measure—there are a myriad of possible metrics that fall into a range of different categories. For example, there are efficiency-based metrics, such as productivity. There are time-based metrics, such as cycle time and the intervals from one milestone to the next. There are cost-based metrics, such as overall development cost and cost per unit of output. There are design-process quality metrics, such as spin count, which can be further divided into metal-only versus all-layer spins. And then there are financial metrics such as return on development investment. Which are most important, and which should be included into a performance measurement framework?

THE KEY PERFORMANCE INDICATOR

Not surprisingly, development productivity is the most important metric in a KPI framework because it's the core of design capability. If a particular IC development project boasts high productivity, it's almost certain to have a short cycle time, low development cost, and high quality (e.g., no unplanned spins). Moreover, as discussed in [Benchmarking IC Development Capabilities-Why?](#), one of the most important benefits of benchmarking is the capability to improve the predictability of cycle time and staffing requirements for newly starting projects. Productivity is the cornerstone metric that makes that possible.

However, productivity gives only one view of design capability. To see the full picture, managers must also measure several other KPIs—cycle time, schedule performance, spin count, throughput, development cost, and so on—all of which are appropriate to the company, organization, or business line.

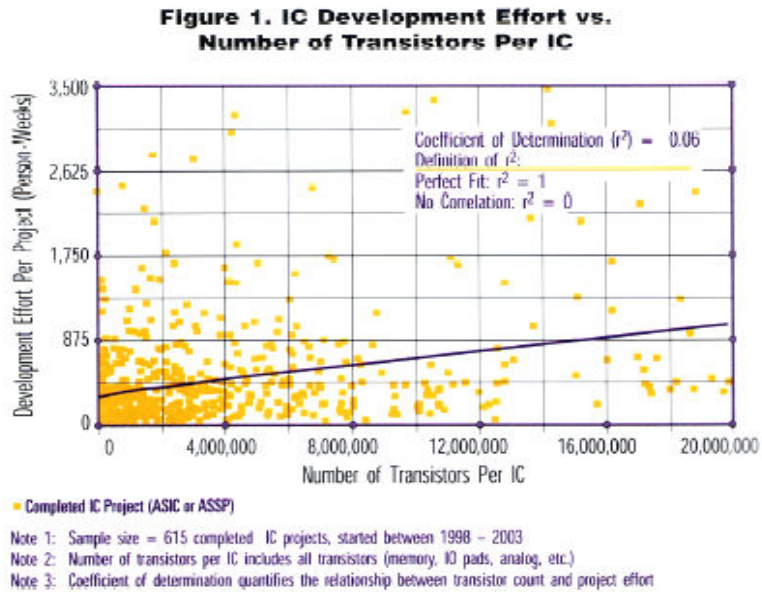
Measuring cycle time, schedule performance, spin count, and the like, is fairly simple. How to measure productivity, on the other hand, is neither obvious nor straightforward. As a starting point, one needs an appropriate definition of productivity. Consider the meaning of productivity in general. The U.S. Department of Commerce defines manufacturing productivity as the "output produced" divided by the "labor expended" on producing that output:

$$\text{Manufacturing Productivity} = \text{Output} \div \text{Labor Expended}$$

This definition yields a metric whose dimensions are output per unit of labor input. Output is expressed in terms of value-added dollars, which is calculated by subtracting the end product's cost of materials from its selling price. This is a good definition, but it's not directly applicable to IC development productivity. If it is applied to IC development productivity, the denominator is similar—in this case, total chip development effort expended (measured in,

say, person-weeks)—but what’s the numerator? That is, what is the output of a design team? What does it produce? Certainly, it produces circuitry in the form of transistors and gates, but neither transistor count nor gate count serves as an accurate measure of design output from one chip to another.

As an example, consider that analog circuits are typically far more complex than digital circuits and therefore demand much more engineering effort to create. A 10-million-transistor chip with only 5% analog circuits typically requires much more design effort than a 10-million-transistor chip that’s purely digital. Indeed, many high-transistor-count designs require significantly less effort than many small designs, as Figure 1 shows. In other words, not all circuits are created with equal effort. So a productivity metric whose dimensions are transistors or gates per unit of labor input can’t be an accurate measure.



Instead, a measure of development output is needed that accurately reflects the amount of engineering and development effort typically required for that device. The effort, of course, is a function of the difficulty of design, or stated another way, the chip’s complexity. Indeed, it’s axiomatic that the more complex a design, the more effort it demands. The greater the complexity, the greater the output. But how is complexity to be measured?

If there were an objective, reliable way to calculate complexity based on the required effort, such that the calculation yielded the standard amount of effort required to design the chip, then that would solve the problem. In this context, “standard” refers to the industry average. It would be a standard industrial measure of effort required, or difficulty. For instance, if one chip’s complexity is, say, 10 million “complexity units” and another’s is five million, then according to industry norms, the first chip should require exactly twice as much effort to develop as the second. A complexity measure based on average development effort establishes a standard baseline for quantifying output.

If this complexity measurement paradigm were possible, then dividing a chip’s calculated complexity by the actual amount of effort expended on developing it, would yield an accurate measure of productivity (expressed, for instance, as “complexity units developed per person-week”). If, for example, the complexity of one chip was 5 million complexity units and

the total effort expended was 5,000 person-weeks, the development team's productivity would be 1,000 complexity units per person-week ($5 \text{ million} \div 5,000$). Likewise, if the total effort expended on developing a 10 million– complexity–unit chip was 8,000 person weeks, the productivity would be 1,250 complexity units per person-week ($10 \text{ million} \div 8,000$).

It is possible to develop a model that calculates the precise number of complexity units of a given chip design. Using data from more than 1,500 projects consisting of more than 20,000 functional blocks, Numetrics has developed both a model and a methodology that accurately quantify complexity. Moreover, such a complexity calculation model can be applied not only at the conclusion of the IC development project, but also at its outset, if the target design's characteristics are either known or can be reasonably estimated. Calculating a design's complexity before initiating development greatly bolsters project planning capability.

CALCULATING COMPLEXITY

Developing a model that calculates chip design complexity is a multistep process. The first step is identifying the major “technology and design attributes” of an IC that influence its development effort. If complexity is to be an objective measurement, rather than a subjective evaluation, the attributes to be taken into account must have a statistically significant impact on the development effort. They must also be observable and quantifiable.

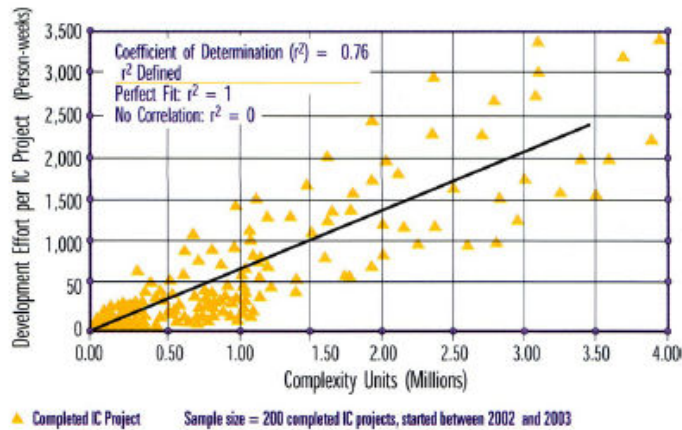
Examples of attributes are the different kinds of circuits on the chip, the number and size of the blocks, clock frequencies and clocking schemes, process technology characteristics, power consumption, reuse, and so on. By themselves, the attributes are really nothing more than a jumble of parameters. Therefore, the second step is to determine the impact each attribute has on development effort. How much, for example, do 90- nanometer design rules influence effort compared with those of a 130- nanometer process? How much more effort is required to develop a chip running at 700 MHz versus one running at 400 MHz? How much less effort is required if a block reuses not just the netlist, but 50% of the layout? What's the impact of radio frequency (RF) circuitry? mixed-signal? etc.

There's only one way to make these determinations—gather a large quantity of reliable experimental data. Although difficult, as noted earlier, it is possible.

After gathering the data, the third step is to extract the relationships between the effort and the attributes. These relationships must then be reduced to weights, or coefficients, which is the fourth step. Finally, an algorithm must sum up and combine the attributes to determine the complexity.

The result is a “normalization” model that quantifies development complexity, or difficulty, across dissimilar projects and expresses it numerically in terms of complexity units (CUs). Accurately quantifying complexity provides a gauge of what each design team produces and sets the stage for measuring productivity across projects. Figure 2 quantifies how well Numetrics' model predicts effort—in other words, its accuracy.

Figure 2. Linear Regression of Complexity Normalization Model



spec development phase than designing a functionally equivalent application specific IC (ASIC). But this difference in the spec development process isn't readily observable by examining the two chips. In fact, if the two devices' functionality were identical, the two would have the same number of CUs, which would mean that the standard amount of effort to develop each is the same. But that would be wrong. The ASSP would likely take longer and require more effort. Therefore, ASSPs must be benchmarked against other ASSPs, not against ASICs.

So, although normalization is necessary to make fair comparisons, it alone is not sufficient. Comparisons must be made among similar projects—similar in terms of scope (i.e., platform versus derivative), application (wireless, transportation, computing, etc.) and in terms of anything that can't be normalized. Fortunately, normalization can neutralize most of the factors influencing effort, so only a modest amount of grouping is required. In sum, grouping similar projects encompasses factors that normalization can't accommodate.

DEFINING THE IC DEVELOPMENT LIFECYCLE

Now that the numerator (output) and the methodology used for measuring it have been established, the specifics of the denominator (effort expended), must be considered. To start, the period over which the effort is to be measured must be precisely defined. The problem is that different companies, and even different business units within companies, often use different start and end times for defining the IC development cycle.

For example, it's common for companies to focus just on engineering, thus thinking in terms of the design cycle, which is typically considered from the time a spec has been created until either first tape-out or final silicon. Slicing off the time expended both earlier and later, which affects time-to-market, development cost, and resource availability, gives a truncated picture of a development project. (Indeed, the first tape-out generally occurs approximately halfway through the development cycle.) So not looking at the full product development cycle yields an incomplete picture that can be misleading.

For an ASSP, we will define the IC development lifecycle as beginning with the "start of

concept,” that is, when resources are committed to investigating the product concept. For an ASIC, the start is “requirements defined,” which may include the functional description, process technology, performance targets, die size, pin count, and the like. For both, the end is the release to volume production. This definition is far more useful than one that’s limited to the design cycle, however defined. It takes into account the full amount of time and effort expended in bringing an IC product to the point at which it starts generating revenue.

It’s also important to define standard interim milestones, as these are vital for benchmarking the various stages of the development lifecycle.

As for the units to be used in measuring effort, it’s appropriate to use person-weeks, as that is the most practical unit for making comparisons among projects.

With the numerator represented as CUs and the denominator as project effort, development productivity (P) is expressed as follows:

$$P = \text{Complexity} \div \text{Effort Expended (or CUs/person-week)}$$

THE KPI FRAMEWORK

The most important performance metric is development productivity. However, chip complexity itself is a very important metric. In fact, it’s the foundation metric of the KPI framework. Comparing the complexity of one’s designs with the competition’s is very valuable. That’s because many companies often develop IC products whose complexity is higher than necessary to win in the market. Likewise, many development teams are forced to tackle excessively high complexity because their organizations don’t have good reuse strategies—good reuse practices and infrastructure obviously reduce design complexity.

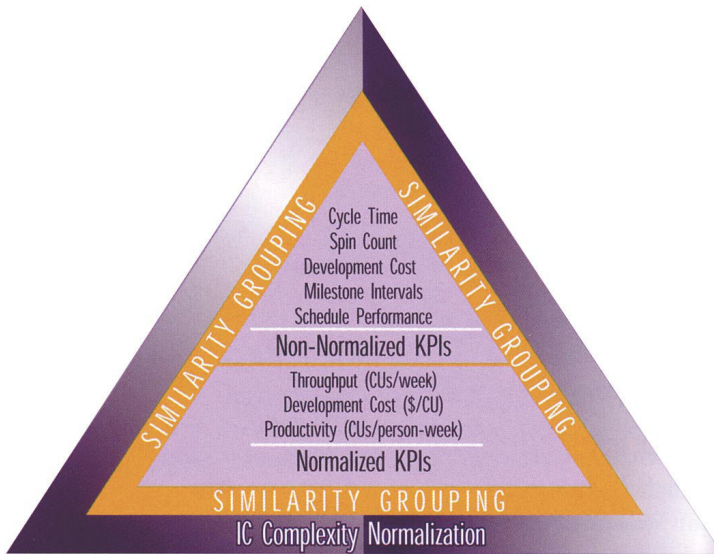
Almost as important as complexity and development productivity is a metric made possible by quantifying chip complexity—design throughput. Throughput is the rate at which output is produced. It can be thought of as normalized cycle time and is expressed as CUs developed per week. Determining design throughput is simple once the output (CUs) is known. Design throughput is the number of CUs divided by the length of the development lifecycle, measured in weeks.

Calculating complexity also makes another normalized metric possible—development cost per unit of output produced. Taking the development cost for a completed IC project—labor, electronic design automation (EDA) tools, Libraries, prototypes, etc.—and dividing it by the chip complexity yields the development cost per complexity unit (dollar per CU). This metric ranks right behind design complexity, productivity, and throughput, because it enables managers to compare project development cost objectively across projects.

Viewing productivity, throughput, and development cost through a normalization lens yields a powerful framework for benchmarking performance. All three are expressed in terms of the same measure of output (CUs) and therefore present an unbiased picture of the three most telling indicators of development performance—efficiency, cycle time, and development cost.

KPIs thus fall into two main categories: normalized and non-normalized. Productivity, throughput, and development cost per CU are normalized metrics. Non-normalized KPIs include cycle time, spin count, schedule performance, time to first tape-out, and many others. Both kinds are integral to a true performance measurement framework.

Figure 3. Key Performance Indicator Framework



Non-normalized metrics are important because they reveal the aggregate effect of a development team’s productivity, throughput, and cost per unit of output. For example, low throughput translates into a long cycle time. Normalized indicators, on the other hand, are an objective reflection of performance, because normalization neutralizes differences in the complexity of the group of designs being considered.

Figure 3 shows the basic metrics of a performance measurement framework. The underpinnings of the framework combine complexity normalization with the grouping of similar projects to ensure apples-to-apples comparisons. This is the foundation that supports the full range of non-normalized metrics, the most important of which are cycle time, the time and the effort consumed between key milestones, schedule performance, spin count, and aggregate development cost.