System and Processor Design Effort Estimation

Cyrus Bazeghi Francisco J. Mesa-Martinez Jose Renau

University of California Santa Cruz
Dept. of Computer Engineering
http://masc.soe.ucsc.edu

Abstract. Design complexity is rapidly becoming a limiting factor in the design of modern high-performance digital systems. The increasing levels of design effort required to improve and implement critical processor and system structures have led to staggering design costs.
As we design ever larger and more complex systems, it is becoming increasingly difficult to estimate how much time it takes to design and verify them. Novel quantitative and optimization approaches are needed to understand and deal with the limiting effects induced by design complexity, which remain for the most part hidden from the architect. To address part of these shortcomings, this work introduces *μComplexity* and *μPCBComplexity*, a set of methodologies to measure and estimate design effort for modern processor and PCB (printed circuit board) designs.

1 Introduction

While the ability to fabricate ever larger and denser circuits is still increasing as predicted by Moore's Law, the semiconductor industry is facing several serious challenges. One of them is the cost of new processor development. Current development costs for top of the line designs are staggering, and are doubling every 4 years [10]. Another challenge is the growing difficulty to correctly design and verify the circuits — which has been called the Design and Verification Gaps [1]. As a result, according to the ITRS 2002 update [1], "the increasing level of risk that design cost and design quality present to the continuation of the semiconductor industry" is of serious concern. The design effort of modern digital systems is further compounded by the need to meet aggressive design constraints such as rising clock frequencies, thermal and power issues, reduced area, increasing number of layers, mixed signal devices, and the ever increasing component count and density.

All of these factors combined have made it increasingly difficult to estimate how much time would be required to design and verify these modern high-performance systems. Ironically, for such a resource-intensive endeavor, there is little systematic work (at least in the public domain) on measuring, understanding, and estimating the effort required by each step in the design of high-performance digital systems. If effort estimates were available early in the design process, they would help identify the critical paths in the whole design process, thus allowing resources to be more effectively allocated and procured. This is essential to keep design costs down and to increase the competitiveness of a company, as architects can access new quantitative approaches to make better design trade off decisions. This work focuses on two of the main areas of complexity in modern systems; circuit boards and processors.

Design effort is defined as the time required, in person-hours, to design and implement a given system. Design effort is equivalent to design time when the project has a single developer. For a given effort requirement, it is possible to reduce the design time by increasing the number of workers. However, as several studies in software metrics and business models have shown, increasing the number of workers may lead to decreases in overall productivity per worker. Since the conversion between design effort and design time can be approximated, the remainder of this work focuses only on design effort.

Different designs have different constraints, leading to specific challenges; typical design constraints are power, area, frequency, and manufacturing cost. For example, having area being a primary design constraint, may lead to a requirement for additional layers, more expensive package types, and/or more complex placement and routing. A design constrained by cost, on the other hand, may require a balance between number of layers, area, drill density, types of packages and possibly the number of different drill sizes. Having clear constraints is necessary in estimating layout effort as it can drastically affect complexity.

This work describes a set of methodologies for measuring and estimating the design effort for modern digital systems. These methodologies are based on the study and analysis of the correlation between multiple design statistics and the overall design effort required to implement these designs. Metrics or combinations there of with good correlation characteristics with overall design time are expected to be good design effort estimators.

This work estimates the design effort for a modern processor as being equivalent to the effort in person-months required to implement and verify the RTL (register transfer level) description of its design. This processor design estimation is based on the μ*Complexity* methodology [2], which consists of three parts: a procedure to account for the contributions of the different components of the design, accurate statistical regression of experimental measures using a nonlinear mixed-effects model, and a productivity adjustment to account for the differential in skills and productivity levels across different design teams.

In order to address some of the concerns related to PCB design time estimation, this work follows a similar approach taken in [2] as the principles that are applicable to microprocessors are also applicable to PCBs. In this work, design effort corresponds to the number of engineering-hours required for implementation (layout) of a PCB or microprocessor design.

To isolate good design metrics for PCB design effort, we explore statistics such as area, component count, pin count and device types and sizes for many PCBs. We analyze several of these statistics, and propose a metric, obtained after applying nonlinear regression over the different statistics, which we call

μPCBComplexity. In addition, we provide insights on the correlation between several statistics and the design effort for many systems with known layout times.

The evaluation shows that a simple statistics like PCB area size and number of components yield some correlation with design effort. With a 90% confidence, pins has a (0.47, 2.09) confidence interval. This means that roughly by looking at the number of pins, the typical design time error is half/double with a 90% confidence. Much better results can be achieved with the proposed *μPCBComplexity* metric. In that case the confidence interval for a 90% confidence is (0.58, 1.72). This roughly means that less than 40% estimation error is achieved with a 90% confidence.

On the processor side, our data shows that any one of number of statements (Stmts), lines of code (LoC) or the fan in of logic cones (FanInLC) is a good single-metric estimator of design effort. Interestingly, this shows some similarity between hardware and software design efforts. On the other hand, it appears that the hardware estimators used elsewhere such as number of cells and transistor count used by the SIA Roadmap and Sematech are not so effective. Most of the other synthesis tools metrics such as area, power and frequency are not well correlated with design effort either. Further evaluation shows that the best estimator is a combination of the two most accurate, which we call Design Effort Estimator 1 (DEE1).

2 Overall Design Flow

The goal of this work is to develop a quantitative approach to estimate design effort based on several easily gathered statistics. This is important because being able to estimate/measure design effort is advantageous in helping to reduce design costs. In order to build a design complexity model, we analyze and gather data from several commercial PCB and processor designs. The layout times for the PCBs and the design times for the processors were well documented, which was a requirement for this analysis.

μComplexity and *μPCBComplexity* are methodologies to measure and estimate the design effort required for a processor design or PCB layout. They comprise three components. The first one is an accounting procedure whereby the design is partitioned into disjoint modules that can be measured individually. A quantification for the entire processor/PCB is obtained by aggregating all the module measurements. The second component is the application of statistical regression to these design measures to obtain an unscaled estimate of the design effort. The final component involves the multiplication of the unscaled effort estimation by a productivity factor, this is done to obtain the estimation of the design effort for a given design team.

In the following subsections, we first review a typical design flow and define the design effort that we are trying to estimate. Next, we discuss the three-component *μComplexity* and *μPCBComplexity* methodologies in detail. Finally, we examine some concerns about the methodologies.

2.1 Design Effort Defined

The system development timeline can be broken down into several overlapping stages as shown in Figure 1. Note that the duration of the different stages is not drawn to scale. The figure also shows an approximation of the size of the engineering team working on the processor portion of the project during each stage. For PCB design it is still fairly typical to have only small teams of 1 or 2 engineers working on the layout stage of the design, which is what we focus on in our PCB analysis.

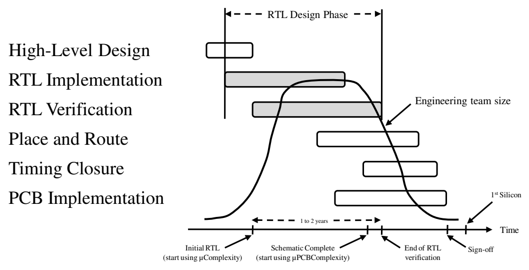

Fig. 1. System development timeline with the size of the IC engineering team. Note that the timeline is not drawn to scale.

In the *High-Level Design* stage, architects perform functional simulation and power estimation of multiple candidate designs. Based on that, they select one microarchitecture and produce a complete functional and interface description of each of its components. Examples of such components are the branch predictor, load-store queue, or floating-point unit. These components are then assigned to engineering teams for implementation. In the case of an ASIC, or if the processor is being designed for an embedded system, the PCB is also be planned during this stage. A Product Requirement Document (PRD) is produced which details the goals of the processor/ASIC and the system board (PCB).

In the *RTL Implementation* stage, engineering teams implement their assigned components in an HDL such as VHDL or Verilog. They continue refining the description until they reach an RTL-level implementation, which can be automatically translated to a gate-level netlist. Functional bugs are fixed as the verification teams discover them. Synthesis is performed to ensure that the timing, area, and power goals are being met.

In the *RTL Verification* stage, engineers create test cases to verify the functionality of individual components and of the whole chip. They perform cycle-accurate simulations and compare the results with the expected values. At this point, the verification team is only concerned with the functional correctness of the design — whether it produces correct answers in a logic-level simulation.

Circuit-level verification, in which electrical and timing parameters are verified, comes later. RTL verification is complete when the number of outstanding bugs reaches zero and stays there for a pre-agreed amount of time.

In the *Place and Route* stage, the synthesized netlist is physically placed within the chip-defined core area based on timing constraints. During the placement phase, gates are resized and some additional logical optimization may be performed. After the initial placement, the routing phase occurs and, if needed, minor placement changes are made. Once the design is successfully placed and routed, clock tree synthesis happens, whereby the clocks in the design have their buffer trees placed and routed.

In the *PCB Implementation* stage, engineers design the schematic and start the layout for the system board onto which the processor/ASIC resides. As chip interfaces become defined and stabilized the requirements for a system board design is gathered. This would include traces and foot prints for any IOs such as PCI, USB, or Ethernet. It would also include the memory system, either a bridge chip with memories, or possible just the memories if the processor/ASIC has a memory controller integrated on chip. A schematic is created and a BOM (bill of materials) is produced. These are then passed on to the layout person or team for implementation of a PCB design.

Finally, in the *Timing Closure* stage, engineers perform timing analysis of the gate-level implementation to determine the maximum clock speed of the design and to identify critical paths. A redesign may be required which could involve RTL or placement–and–route changes. A refine–test–refine loop exists between the Place-and-Route and Timing Closure stages.

As shown in Figure 1, the focus of this part of the work is the period that includes both the RTL Implementation and the RTL Verification stages. We define *Design Effort* as the number of person-months spent implementing the description of the processor in a hardware design language such as VHDL or Verilog, refining it to an RTL description, and verifying the latter for functional correctness. We exclude any additional time required to revise the design later, during the Timing Closure process. While the period considered excludes some design time, we believe that it includes the bulk of it.

In the following sections we describe the accounting procedure we use for the PCB and processor evaluations. In 2.2 we look at the accounting procedure stage of μ*Complexity*, whereby the design is partitioned into disjoint modules that can be measured individually. In 2.3 we discuss the critical design parameters of a PCB and how the accounting for μ*PCBComplexity* is developed from them. A quantification for the entire processor is obtained by aggregating all the module measurements. In 2.4 we discuss the use of a productivity adjustment.

2.2 Approach to μ*Complexity*

This assumes a processor design to represented as a collection of hardware description language (HDL) statements, thus ignoring certain design issues introduced to processor components implemented using custom layout and not standard cell designs. As described in Section 1 the design effort for a modern processor is directly proportional person-months required to implement and verify the RTL description of its design.

To measure overall design effort, estimates of the effort for each processor component must be obtained, and then added into a compounded index. However, components may be instantiated several times through any given design. Some components may also be *parameterized*, and different-sized instances could be generated. Parameters could be the width of the input or output buses, queue depth, or pipeline depth. To address these cases, we use the following two rules.

Account for a single instance of each component. When a design reuses a component (e.g., an ALU), we only count the design effort of one instance of it. The rationale is that, in accordance with the principles of modular design, the effort required to design and verify the component is a one-time cost. Once the component is designed and verified, it can be re-used elsewhere with negligible effort.

Minimize the value of component parameters. To estimate the design effort of a parameterized component, we set each parameter to the minimal value that does not result in a degenerate case. We refer to this minimization of parameters as *scaling*. The rationale is that, while different parameter values can drastically change the size of the component instance (in terms of chip area or number of gates), it is not much harder to write parameterized code than it is to write code for the smallest nontrivial instance.

More formally, consider a VHDL description where the parameterized component is implemented with `GENERATE` loops. We select for each parameter the smallest value that does not cause any loops or conditional statements in the RTL description to be optimized away by traditional program analysis techniques such as constant propagation and dead code elimination. The process for Verilog is more difficult to formalize because Verilog did not have an equivalent of the `GENERATE` construct until Verilog-2001 was introduced. However, the determination of what constitutes the minimal non-degenerate parameterization is conceptually the same.

Design Effort Estimator There are multiple metrics that may be related to design effort. Examples include the number of logic gates or the number of HDL lines in the design description. Consequently, for each component in the design (subject to the constraints of Section 2.2), we measure these metrics. Then, we select a single metric or a set of metrics (e.g., the number of gates and the number of HDL lines) and use statistical regression [11] to find how well they correlate with the person-months design effort reported by the processor designers. For each set of metrics m_1, m_2, $\ldots m_n$, we find the best values for the coefficients w_1, w_2, $\ldots w_n$ in Equation 1. The result is a *Design Effort Estimator* (*eff*):

$$\text{eff} = \frac{1}{\rho} \times \sum_{k=1}^{n} (w_k \times m_k) \qquad (1)$$

The regression model used is described in Section 3. In the equation, ρ is the productivity factor for the design team. It allows the same set of coefficients w_k to be used in different projects. The rationale for ρ is discussed next.

2.3 Approach to $\mu PCBComplexity$

Printed circuit board (PCB) design effort keeps growing due to such constraints as rising clock frequencies, thermal issues, reduced area, increasing number of layers, mixed signal devices, and the ever increasing component count and density. All of these factors combined have led to a steady rate of increase in development costs for current systems. As we design ever larger, denser and more complex systems, it is becoming increasingly difficult to estimate how much time would be required to design and verify them. To compound this problem, PCB design effort estimation still does not have a quantitative approach.

The lists of critical components of PCB designs is determined by [4]. These parameters contribute to the complexity of a design, and hence the time required to do layout. Some design parameters are dependent on other factors. For example, the size of the board is defined by the number of embedded and discrete passive components and total wiring requirements. However, the total wiring requirements are governed by the number of embedded and discrete passive components in the PCB. And furthermore, the total number of layers in the PCB depends on the size of the board, the number of embedded and discrete resistors and bypass capacitors [4].

These critical design parameters are focused towards manufacturability, not design effort estimation. We used them as a starting point in determining what parameters or metrics to analyze and include for correlation with design effort. None of the boards in our study have embedded passive components; instead we focus on the total number of all components (passive and discrete) and the pin count for them. These are easily obtainable values.

Since the routing data is not easily obtainable, the number of pins for all the components in the design is taken into account instead. While this is not an ideal metric since not all pins are used or have very short traces (VDD or GND), it is readily obtainable and does not hamper the focus of this paper, namely effort prediction starting from higher level design descriptions, such as a bill of materials (BOM) or schematics.

In order to find a metric highly correlated with design effort, several statistics were gathered from the existing designs. For each isolated board with a known design effort, we look at several statistics and apply nonlinear regression to find a highly correlated metric.

We present our design effort model as the aggregate of a set of statistics (S_i). Each of which has a specific constant (w_i), associated with it, which assigns a weight to the importance of every statistic used as input in the model. The aggregate of the statistics is inversely proportional to the productivity of a specific design team which is represented by a constant (ρ). The model is presented in Equation 2. In order to find suitable values for each of the data weights (w_i) we perform mixed nonlinear regressions on this equation. The design team productivity factor (ρ) is constant per design group, and it needs to be adjusted on a per company or design team basis. If the ρ is unknown, then the absolute design effort is invalid and only the breakdown inside the project is correct. Obtaining the value of ρ is simple; all that is needed is to have the design effort for a single project. Alternatively, it is possible to develop a productivity benchmark suite that calibrates ρ for a given company.

$$\text{Design Effort} = \frac{1}{\rho} \times \sum_{k=1}^{n} (w_k \times S_k) \tag{2}$$

In order to determine the weights that give a generalized solution to Equation 2, [2] proposes to use a mixed nonlinear regression model. If there are no productivity adjustments, it is possible to use a simpler nonlinear regression model. While the sum of a large number of random variables is distributed normally, the product of a number of random variables is distributed *lognormally* — a distribution where the logarithm of the variable is normally distributed [5]. Therefore, since the random variables have a log normal distribution an even simpler linear regression model can not be used.

To evaluate the accuracy of the model (Section 4.2), we use σ as a measure of error associated with the fit. Consequently, it is important to understand what different values of σ tell us about the quality of the estimate. For a given σ, we can find a *confidence interval* for the estimated effort. The x% confidence interval for a metric is defined to be the range of efforts $(Estimate_{low}, Estimate_{high})$ such that $P(Estimate_{low} <$ metric prediction $< Estimate_{high}) = x/100$. For example, the 90% confidence interval gives us two values a and b such that there is a 90% chance that the actual effort is between metric prediction $\times\, a$ and metric prediction $\times\, b$.

2.4 Productivity Adjustments

In software development projects, it is well known that different development teams have different productivities. For example, it has been shown that the productivity difference between teams can be up to an order of magnitude [8]. We believe that a similar effect occurs between PCB and processor design teams. The productivity differences may be due to multiple factors, including the average experience of the designers in the team and the tools used. In our model, ρ captures this effect.

The designs under study in this analysis were produced either by a single manufacturer, or a just one design from a specific manufacturer was provided. Therefore the use of a productivity factor was not necessary as we did not obtained competing designs from multiple manufacturers of from multiple competing teams among a single manufacturer.

A Model Without Productivity Adjustments For the processor analysis, we can eliminate productivity adjustments by setting $\rho_i = 1$ for all i simplifies the statistical model. Instead of using the nonlinear mixed-effects model described in Section 3.1 to fit the weights, we can use a simpler multiple regression technique. Unfortunately, as we show in Section 4.1, the model without productivity factors fits the data poorly. We present it only for comparison with the recommended nonlinear mixed-effects model of Section 3.1.

A model without productivity adjustments may be acceptable for industrial practitioners with a very large single project, perhaps representing thousands of person-months of effort. In this case, they can set $\rho = 1$, since there is only one project and therefore no need to account for productivity differences across projects.

2.5 Issues

Ideally, we would like to use design effort estimators as soon as possible in the system design timeline. The earlier the estimations can be made, the more useful they are likely to be. After adjusting the coefficients w_i shown in Equation 1, early estimation presents a clear challenge: how to ensure that the values of the early metrics remain relevant (and valid) at later stages of the design.

To address this, we use metrics whose value changes little from initial stages of the design until completion of the RTL implementation and verification in the case of the processor, or pins and components, in the case of a PCB. Specifically, the metrics analyzed in this work can be measured once a module has been designed and before it starts to be verified. This corresponds to the point shown with an arrow in Figure 1, which is often 1 to 2 years before completing the RTL verification. The values of the metrics remain largely unchanged until the end of RTL/PCB verification. The exception is if the verification finds substantial bugs that require a major re-design.

One potential objection to the accounting procedure described is that counting each component only once regardless of its number of instances may not be appropriate. For example, at a very low level, we could consider that the entire processor is made out of logic gates, and that there are only a dozen or so types of gates. The analysis would clearly be inaccurate. However, at the high level of the functional components that we are discussing, the count-only-one heuristic is appropriate. Regardless, any given component is likely to have fewer than ten instances. At this level, scaling the effort estimate linearly with the number of instances does not seem appropriate.

In our discussion of parameter scaling in section 2.2, we argued that writing code for a parameterized component is no more difficult than writing code for the smallest nontrivial instance of it. In practice, however, the parameter values chosen for a given instance may affect the number of test vectors required for verification and, therefore, the verification time. For example, model checking and automatic theorem-proving tools may require more time to run with larger parameter values, since the size of the state space may be larger. However, this issue could be addressed, at least conceptually, by allocating more computational resources to the verification budget — not more engineer-hours.

The parameter scaling rule has another undesirable consequence. Specifically, varying the value of certain parameters may have implications on the difficulty of timing closure and, therefore, on the number of RTL redesign iterations. An example is the degree of associativity of a time-critical structure: higher associativity may make it hard to perform timing closure and may induce several redesigns. This issue suggests the need for future design effort estimators that are aware of back-end physical design and timing concerns.

Finally, our analysis has implicitly assumed that each component in the design is implemented from scratch. In practice, components are sometimes reused from older designs, often with little modifications. Integrating a reused component incurs some design effort, even if it requires no modification at all. The software engineering literature has discussed effort estimation for reused components [3]. We regard the study of reuse in hardware as a subject for future work.

Productivity Adjustments The volatility of ρ may make it difficult to use the model to make extrapolations across different projects. Once RTL coding is completed, all of the metrics are available, but it is still difficult to determine the productivity factor until after at least some of the components are completely verified. One option is to estimate ρ using data from a very recent project or to extrapolate the current value of ρ given a time series of previous values.

Unfortunately, we have no means of evaluating this approach. A second option is to assume $\rho = 1$ and use the model to make *relative* estimations only. Even without knowing ρ, we can still say that a component with an estimated design effort of $e = x$ is likely to take half as much effort to design as one with $e = 2x$. These relative estimates may be useful when allocating engineers to verification teams. They may also allow an early determination of which components are likely to delay project completion.

3 Regression Model

As indicated in Section 2.2, given a set of metrics m_1, m_2, $\ldots m_n$, the goal of the regression procedure is to find the w_1, w_2, $\ldots w_n$ values for Equation 1 that provide the best fit for the person-months design effort reported by the designers. Each component in the design for which we know the design effort (e.g., fetch unit or load-store queue), is a data point consisting of the reported design effort and the measured metrics. The more data points we have, the more precise the determination of w_k is.

The data points for this work come from several small projects implemented by unrelated design teams at different times. Consequently, in addition to the usual statistical variation across data points, there is variation across teams. In statistical terms, this forces us to introduce a per-project *random effect* (represented by the productivity ρ). Therefore, we use a *nonlinear mixed-effects* model [15], which is able to deal with both *fixed* and *random* effects better than more conventional linear methods [7, 11].

In the following section, we describe the mixed-effects model that we use and then consider what would happen if we attempted to fit a simpler model without productivity adjustments.

3.1 A Nonlinear Mixed-Effects Model

When we use Equation 1 with data from multiple projects, we have one data point for each component j designed in project i. The estimated design effort eff_{ij} is given by Equation 3. Note that for each component j from project i, we have a set of n metrics m_{ijk}. There is a productivity factor ρ_i specific to each project. However, the coefficients w_k are assumed invariant across all data points. In reality, of course, the fit is not perfect and the actual (reported by designers) design efforts Eff_{ij} are different from the estimated ones eff_{ij} (Equation 4). The difference is accommodated by the ϵ_{ij} error term, which we assume is multiplicative.

$$\mathrm{eff}_{ij} = \frac{1}{\rho_i} \times \sum_{k=1}^{n} (w_k \times m_{ijk}) \tag{3}$$

$$\mathrm{Eff}_{ij} = \mathrm{eff}_{ij} \times \epsilon_{ij} \tag{4}$$

To fit the mixed-effects model and determine the w_k, we need to treat ρ and ϵ as independent random variables. As such, we must provide a probability distribution for each. From software engineering, we know that productivity is determined by the product of a collection of variables (e.g., team cohesiveness, tool quality or process maturity) [3]. Since the sum of a large number of random variables is distributed normally, the product of a number of random variables is distributed *lognormally* — a distribution where the logarithm of the variable is normally distributed [5]. Similarly, software engineering studies tell us that the multiplicative error ϵ is also lognormally distributed [16]. Consequently, we use a lognormal distribution for both ρ and ϵ.

The lognormal distribution is described by two parameters: μ and σ. They represent, respectively, the mean and standard deviation of the *log* of the variable. For the ρ and ϵ distributions, we choose to set $\mu = 0$, and then let the fitting procedure determine the standard deviations σ_ρ and σ_ϵ. The result of setting $\mu = 0$ in both cases is that the median of the distributions is 1. Intuitively, this means that half of the projects have $\rho > 1$ and half have $\rho < 1$. Similarly, half of the estimations have $\epsilon > 1$ and half have $\epsilon < 1$. Figure 2 shows a lognormal distribution with $\mu = 0$, showing the difference between mean, median, and mode.

Our choice also means that the resulting estimated effort eff that we obtain is the *median* design effort. To determine the estimated mean design effort $\overline{\mathrm{eff}}$ rather than the estimated median design effort, we would apply Equation 5.

$$\overline{\mathrm{eff}} = \mathrm{eff} \times e^{(\sigma_\epsilon^2 + \sigma_\rho^2)/2} \tag{5}$$

In Section 4.1, we use σ_ϵ as a measure of goodness of fit. Consequently, it is important to understand what different values of σ_ϵ tell us about the quality of

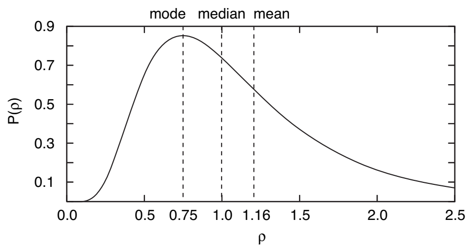

Fig. 2. Example of a lognormal distribution with $\mu = 0$.

the estimate. Specifically, we say that σ_ϵ determines a *confidence interval* for the estimated effort. The $x\%$ confidence interval for eff_{ij} is defined to be the range of efforts (el_{ij}, eh_{ij}) such that $P(el_{ij} < \text{Eff}_{ij} < eh_{ij}) = x/100$. For example, the 90% confidence interval gives us two values a and b such that there is a 90% chance that the actual effort is between a and b. Figure 3 plots the 68% and 90% confidence intervals for a range of σ_ϵ. To compute the confidence interval for a given σ_ϵ and eff_{ij}, find the value y_h corresponding to the top of the interval and the y_l corresponding to the bottom of the interval. The confidence interval is then $(y_l \times \text{eff}_{ij}, y_h \times \text{eff}_{ij})$. For example, if $\sigma_\epsilon = 0.45$ then $y_h \approx 2.1$ and $y_l \approx 0.5$. Therefore, the 90% confidence interval for Eff_{ij} is $(0.5 \times \text{eff}_{ij}, 2.1 \times \text{eff}_{ij})$.

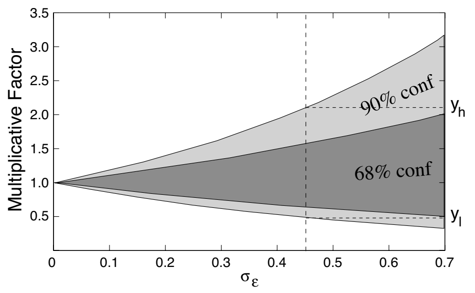

Fig. 3. 68% and 90% confidence intervals corresponding to $0 \le \sigma_\epsilon \le 0.7$. The figure demonstrates finding the multiplicative factors y_h and y_l for the 90% confidence interval corresponding to $\sigma_\epsilon = 0.45$.

We perform model fitting computation using the `NLMIXED` procedure from *SAS* [15], although we could also use the `nlme` package from *R* [19]. Equation 6 shows an alternative method for approximating rho_i given the w_k.

$$\frac{\sum_j \bar{e}_{ij}}{\sum_j E_{ij}} = 1$$

$$\rho_i \approx \frac{exp\left(\sigma_\epsilon^2/4\right)\ \sum_j \sum_{k=1}^{n} (w_k \times m_{ijk})}{\sum_j E_{ij}} \tag{6}$$

4 Evaluation of Processor Designs

The evaluation of this work examines how accurately each of the software and synthesis metrics correlate with design effort. This section is divided into two parts, Section 4.1 shows processor design metrics and Section 4.2 shows PCB design metrics. For both cases, we also examine a few combinations of metrics.

4.1 Processor Designs

In our processor design analysis, we compare against some of the design effort estimators currently being used. Specifically, Sematech [10] and the SIA Roadmap [1] which use the number of cells and the number of transistors, respectively, to estimate effort. We also analyze other synthesis statistics which are often used to make effort estimations.

As indicated in Section 3.1, to assess the accuracy of an estimator, we report the standard deviation of its error (σ_ϵ). Lower values of σ_ϵ are better, and zero is the minimum possible value. Given a σ_ϵ, we can compute the interval for, say, 90% confidence for the true value. For the lognormal distribution used, the mapping between σ_ϵ and the 90% confidence interval.

In the following analysis, we first measure the accuracy of the different design effort estimators using our model. Then, we repeat the process without the productivity adjustment or without the *µComplexity* accounting procedure.

Accuracy of Design Effort Estimators Table 1 shows the accuracy of various design effort estimators. First, Column 2 lists the reported design effort in person-months for each component of each design. Then, each of remaining columns shows data for one design effort estimator. Most of the estimators are simply the individual software or synthesis metrics. The only exception is the DEE1 estimator, which is the linear combination of two metrics — we analyze DEE1 in Section 4.1. For a given estimator, the column shows its value for each component of each design and, in the penultimate row, its σ_ϵ.

From Table 1, we see that there are a group of estimators that have a relative high accuracy (i.e., low σ_ϵ). They include Stmts, FanInLC, and Nets. For example, Stmts and FanInLC have σ_ϵ equal to 0.50 and 0.55, respectively, which, correspond to a 90% confidence interval of (0.44,2.28) and (0.40,2.47), respectively. Really, within the margin of error of our study, any one of Stmts or FanInLC has the same accuracy. The other estimators, namely Freq, *Power*, $Area_L$, $Area_S$, Cells, and FFs, have lower accuracy. For example, $Area_L$ has σ_ϵ equal to 1.23, which corresponds to a 90% confidence interval of (0.13,7.56). None of these metrics is a reasonable estimator.

Module Name	Effort (Months)	DEE1	Stmts	FanInLC	Nets	Freq (MHz)	$Area_L$ (μm^2)	$Power$ (mW)	$Area_S$ (μm^2)	Cells	FFs
Leon3-Pipeline	24	12.8	2070	10502	4299	56	50199	80	68411	3586	1062
Leon3-Cache	6	7.3	1172	6325	1980	94	37456	57	12556	3	210
Leon3-MMU	6	4.4	721	3149	1130	84	60136	23	112765	246	699
Leon3-MemCtrl	6	5.4	938	2692	853	138	7394	5	11938	704	275
PUMA-Fetch	3	2.2	586	5192	1292	68	147096	226	555168	1809	1786
PUMA-Decode	4	6.2	1998	4724	5662	65	78076	11	47604	5189	464
PUMA-ROB	4	2.2	503	6965	9840	41	82527	733	1022	9709	922
PUMA-Execute	12	12.6	3762	18260	10681	49	92473	44	119746	10867	1725
PUMA-Memory	1	3.3	976	5034	1089	60	43418	80	115841	4337	1549
IVM-Fetch	10	8	1432	15726	4914	71	212663	8	135074	1859	1661
IVM-Decode	2	1.7	391	1044	504	104	2022	2	73	2	0
IVM-Rename	4	2.7	566	3307	1134	159	70146	1	26740	121	510
IVM-Issue	4	3.6	624	8063	4603	60	90388	2	68667	3414	2729
IVM-Execute	3	5.4	961	11045	4476	91	619561	5	154655	940	0
IVM-Memory	10	11.6	2240	19021	23247	54	267753	73	625952	12050	2510
IVM-Retire	5	5	1021	6635	3357	71	36100	2	50375	1923	924
RAT-Standard	0.6	0.7	64	3889	2905	137	34254	4	17603	2596	288
RAT-Sliding	1	1	78	5586	4936	119	52210	10	60713	4507	612
σ_ϵ	–	0.46	0.50	0.55	0.67	0.94	1.23	1.34	2.07	2.09	2.14
σ_ϵ ($\rho_i = 1$)	–	0.53	0.60	0.82	1.08	1.12	1.35	1.82	2.07	2.55	2.18

Table 1. Accuracy of various design effort estimators.

Freq has a 90% confidence interval as large as (0.21,4.69). While increasing processor frequency requires additional design effort, other metrics like Nets or FanInLC have higher correlation with design effort. The reason is that, to increase frequency, it is necessary to add extra pipeline stages or more complex logic. This increased effort is better measured by Nets and FanInLC.

Perhaps unsurprisingly, $Area_S$ and FFs are not well correlated with design effort. Their 90% confidence intervals are (0.03,30.11) and (0.03,33.78), respectively. The reason is that storage structures such as RAM banks are relatively simple to design. Similarly, $Area_L$ and Cells are not well correlated because simple to implement structures can occupy a lot of area and have large numbers of logic cells. Moreover, neither dynamic nor static power is well correlated with design effort as their confidence intervals are (0.11,9.06) and (0.09,10.68) respectively. Larger designs probably require more power, but are not necessarily more complicated to design.

Overall, our data shows that any one of Stmts or FanInLC is a good single-metric estimator of design effort. Interestingly, this shows some similarity between hardware and software design efforts. On the other hand, it appears that the hardware estimators used elsewhere such as Cells and transistors used by the SIA Roadmap and Sematech are not so effective. Most of the other synthesis tools metrics such as area, power and frequency are not well correlated with design effort either.

Design Effort Estimator 1 (DEE1) We have also analyzed the accuracy of estimators generated with the linear combination of groups of two metrics. As usual, we use Equation 1 from Section 2.2. We find that two-metric combinations that include Stmts, FanInLC, and Nets tend to have slightly more accuracy than those with a single metric. The ones that are the most accurate are Stmts plus

Nets, and Stmts plus FanInLC. They have the same accuracy, but we prefer the Stmts plus FanInLC estimator because, individually, the metrics are more accurate. We call the resulting estimator Design Effort Estimator 1 (DEE1).

As shown in Table 1, DEE1 has the lowest σ_ϵ, namely 0.46. This corresponds to a 90% confidence interval of (0.47,2.13). The slightly higher accuracy of DEE1 comes from the fact that its two component metrics measure slightly different underlying factors in the design.

To see the correlation between DEE1 and the reported design effort better, Figure 4 shows a scatter plot of DEE1 estimations versus reported design effort. The Figure has one data point per component and design. From the figure, we see that most of the DEE1 estimations are very close to the reported design effort. The exception is the data point for the Leon3 pipeline, where the DEE1 estimation is 12.8 months, and the reported effort is 24 months. In practice, most of the estimators in Table 1 underestimate the effort for the Leon3 pipeline. The reason is that this pipeline is more sophisticated than the other components and designs. Indeed, while IVM and PUMA only execute a subset of Alpha and PowerPC, respectively, Leon3 is a full SPARC V8 compliant processor. In addition, Leon3 is highly configurable, for example the user can select different processor and cache parameters.

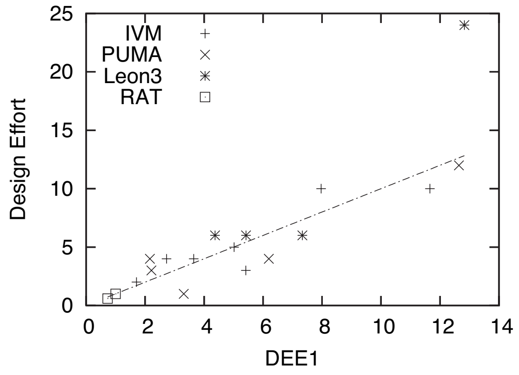

Fig. 4. Scatter plot of DEE1 estimations versus reported design effort.

Accuracy without the Productivity Adjustment The last row of Table 1 shows the σ_ϵ values that would be obtained if no productivity factor was used – in other words, if ρ_i was 1 for the Leon3, PUMA, IVM, and RAT teams. This approach was mentioned in Section 2.4.

From the values of σ_ϵ, we can see that practically all the estimators lose a significant amount of accuracy. For example, the σ_ϵ for Stmts and FanInLC becomes 0.60 and 0.82, respectively, which correspond to 90% confidence intervals of (0.37,2.68) and (0.26,3.85), respectively. Similarly, DEE1 expands its 90% confidence interval to (0.41,2.39).

The loss of accuracy for Stmts is due to several factors. Specifically, while Leon3 uses VHDL, the other designs use Verilog. Moreover, while RAT uses the more compact Verilog-2001, PUMA and IVM use the more verbose Verilog-95. Additionally, different coding styles add much noise to any correlation without productivity adjustment. To compound the problem, it is known from software projects that productivity across teams can vary by an order or magnitude [8].

The FanInLC and Nets estimators lose accuracy because each processor was designed under a different set of constraints and a different set of tools. For example, since Leon3 was designed for an area-constrained environment (FPGAs), a substantial effort was needed to reduce area and interconnections. On the other hand, PUMA's target was a high frequency CGaAs process. All these effects again add noise to any correlation.

Overall, we conclude that, to have good processor design estimation accuracy productivity adjustments are required.

4.2 Evaluation of PCB Designs

We analyze 12 different printed circuit boards from two separate companies. Table 2 shows the main results and characteristics for each of these. The first column corresponds to each of the statistics or metrics measured. Columns B1 to B12 correspond to each of the boards. The last column corresponds to the σ between the row and design effort. Since the boards either were designed by the same team, or we only had one board from a particular company, we do not evaluate the productivity factor (ρ). This simplifies the analysis, and we can use nonlinear regression instead of the mixed-effects nonlinear regression model. With σ we can compute the confidence interval. For the lognormal distribution used, the mapping between σ and the 90% confidence interval is shown in Figure 3. We use this chart to compare the accuracy of different estimators.

The design effort values were obtained by interviewing the original designers. Obviously, there is perfect correlation with itself so $\sigma = 0$. A zero σ results in a perfect $(1, 1)$ confidence interval. We now proceed to analyze easily available statistics like number of components and pin count. These two sets of statistics are easily available before the PCB design starts. They are part of the PCB specification.

From the boards analyzed, we observe that it is best to use the total number of components to estimate design effort ($\sigma = 0.53$). Although traces for analog components and digital components are more difficult than traces for passive components, the low amount of digital and/or analog components on several of the boards make it difficult to use them as a method to estimate effort. Using Figure 3 and a $\sigma = 0.53$, the intersection between the components line and the confidence interval line is $(0.41, 2.39)$. This means that using the number of components on the specification, we have a 90% confidence that the design effort would be between 0.41 and 2.39 times the prediction.

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	σ
Design Effort	68	35	43	21	48	48	24	40	32	24	12	400	–
Components													
# Passive	213	165	101	80	108	222	116	86	83	19	47	2643	0.56
# Digital	15	0	17	0	8	2	0	11	8	4	4	94	1.79
# Analog	35	24	8	10	24	53	28	4	16	1	11	91	1.18
Total #	263	189	126	90	140	277	144	101	107	24	62	2828	0.53
Total Area	6214	9053	6964	2719	9144	6579	8104	12193	12296	777	5430	38611	0.75
Pins													
Passive	563	429	365	182	414	578	414	194	188	39	109	5843	0.62
Digital	154	0	518	0	107	32	0	175	173	88	32	6889	1.88
Analog	360	208	216	98	72	448	150	25	53	14	65	924	1.10
Total	1077	637	1099	280	593	1058	564	394	414	141	206	13647	0.45
PCB Size	221	221	221	162	387	204	221	109	109	12	254	726	0.93
# of Sides	1	1	1	1	1	1	1	2	2	2	1	2	0.81
# of R. Layers	2	2	3	2	2	2	3	2	2	4	2	6	0.66
# of Layers	4	4	6	4	4	4	4	2	2	4	2	8	0.67
Comp. Density	70	50	33	33	21	80	38	27	29	55	14	115	0.60
Pin Density	54	32	55	19	17	57	28	40	42	122	9	207	0.64
µPCBComplexity	60	38	37	18	30	61	25	36	37	25	12	543	0.24

Table 2. Statistics, design effort, and correlation results of study boards.

Statistics about the pins are as easily available as components even before the design starts. The number of pins is a better predictor ($\sigma = 0.45$) than the number of components. The resulting 90% confidence interval for the number of pins is $(0.47, 2.09)$. This means that just by using the pins, we have a 90% confidence that the prediction is around half or double the expected design effort. Not shown in the table is the result of combining the number of pins and the components to predict design effort. The results did not improve because there is a high correlation between pins and components.

Area is not such an effective metric. Even assuming a perfect knowledge if the final dimension of the board, we can just estimate design effort with a $(0.21, 4.61)$ confidence interval. Table 2 also shows other statistics such as number of sides used, routing layers, and number of layers. Those statistics are not so useful by themselves because they are highly quantized, and this makes them difficult to use to predict effort.

To obtain the proposed *µPCBComplexity* metric shown in Table 2, we analyzed multiple combinations of parameters and followed suggestions from experienced board designers. The best results were achieved when using the following equation:

$$\text{Effort} = \text{w1} * \#\ \text{Components} + \text{w2} * \text{Comp. Density} + \text{w3} * \text{Pin Density} \quad (7)$$

To capture component and pin density, we define them with equation 8 and equation 9 respectively.

$$\text{Component Density} = \frac{\#\ \text{Components}}{\text{PCB Area} \times \#\ \text{Sides w/ components}} \quad (8)$$

$$\text{Pin Density} = \frac{\#\ \text{Pins}}{(\text{PCB Area})} \quad (9)$$

To obtain the factors on equation 7, we perform nonlinear regression as explained in Section 2.3. Although neither pin nor component density can achieve better predictions than the number of pins, when integrated together in the *μPCBComplexity* metric we achieve a 0.24 σ. As Figure 3 shows, this represents a $(0.58, 1.72)$ confidence interval. This roughly means that by using the proposed *μPCBComplexity* metrics, with a 90% confidence designers can predict design effort with less than 40% error.

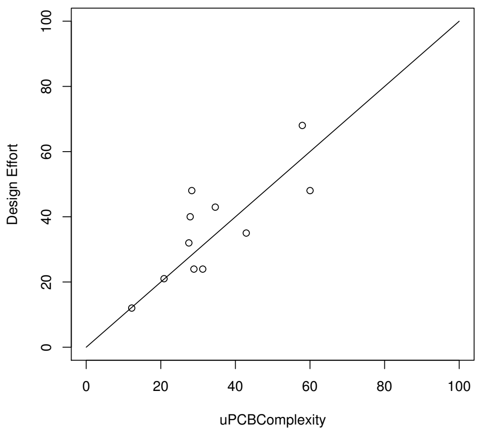

Fig. 5. Scatter-gather plot of design effort vs. PCB metric

Figure 5 shows a scatter-gather plot between design effort and *μPCBComplexity*. Each point corresponds to a different board. The plot does not include the B12 board to zoom on the area where most of the boards are located. This plot is an intuitive way to see that there is a high correlation between design effort and the metric proposed.

μPCBComplexity works well because PCB design complexity increases as the component and pin density increases. Designers can increase the number of layers on the PCB to decrease the pin density or increase the area to reduce both densities. The problem is that both approaches require more costly boards. As a result, designers tradeoff between time to market and density.

5 Related Work

The work most related to ours in processor analysis is done by Numetrics, a company specializing in enterprise software and services product development [18].

They propose a “complexity unit” to measure the level of project difficulty and to quantify the development team’s output. Patent 6,823,294 describes a method to estimate design effort. If we apply the method to our data, the result is considerably less accurate than DEE1. After discussions with Numetrics, they informed us that the patent represented preliminary work, and that their current models are more advanced. Unfortunately, little detail is available on these models because it is considered a technological advantage for their company.

Kahng [12] identifies the need for standards or infrastructures for measuring and recording the semiconductor design process. The author proposes improving design technology, time-to-market, and quality-of-result by addressing the Design Productivity Gap and the Design ”Technology” Productivity Gap. However, this previous work focused mostly on the problems associated with the infrastructure and design tools related to the physical implementation of semiconductor designs, while the focus of this work is layout effort associated with PCB designs and design effort for processor flows.

In [17] introduces a weighting approach similar to the productivity factor described in our work. They use the “process productivity parameter” to tune the estimating process for software projects. They contend that if you know the size, time, and the process productivity parameter you can use it to make estimates for a new project. So long as the environment, tools, methods, practices, and skills of the people have not changed dramatically from one project to the next.

In [4] the issue of embedded passive components is discussed as a necessity to the smaller electronic devices requiring ever smaller PCBs. They note that board area is becoming so critical that to keep pace with the size constraints new techniques are required. Our goal would be to eventually develop a set of metrics and a model that estimates design effort by also taking into account manufacturing times.

Recently, some research has focused on reducing the number of RTL redesigns during the timing closure process. To streamline timing closure, new methods have been developed to predict logic criticality [13] and wire congestion [14] early in the RTL design phase. With these predictors, logic designers can focus their attention on the critical logic during the initial implementation, reducing the number of redesign cycles.

As process technology has improved, the major source of signal propagation delay has shifted from gates to wires. In [6] a new metric for evaluating interconnect architectures is proposed. The metric is computed by looking for an optimal assignment of wires from a given wire length distribution. This information is used to generate an interconnect architecture. That metric compares impacts of geometric parameters as well as process and material technology advances on designs.

Fornaciary *et al.* [9] propose a methodology to predict the final size of a VHDL project on the basis of a high-level description. With this, they seek some indication of development effort by estimating the number of lines of code from starting specifications. While their method is shown to be accurate in predicting lines of code, it dies not address design effort aspects, such as the number of engineering person-months required for the project.

6 Conclusions

Design complexity is rapidly becoming a limiting factor in the design of modern, high-performance microprocessors and systems. This work addresses the lack of quantitative approaches to estimate the design effort for modern systems and processors by making three major contributions:

First, we use the *μComplexity* methodology to measure and estimate processor design effort. *μComplexity* consists of three main parts, namely a procedure to account for the contributions of the different components, accurate statistical regression using a nonlinear mixed-effects model, and a productivity adjustment to account for the productivities of different teams.

Second, we apply *μComplexity* to four designs and evaluating a series of estimators based on synthesis and software metrics. The evaluation uncovered a few simple, good design effort estimators, namely the number of lines of HDL code (or HDL statements) and the sum of the fan-ins of all the logic cones. A slightly more accurate estimator is DEE1, which is the linear combination of HDL statements and fan-ins of all the logic cones. We recommend this estimator, but using estimators that combine a larger number of metrics may make sense for a practitioner that has access to more data.

Third, we introduce a procedure, *μPCBComplexity*, to estimate PCB design effort. PCB design effort is estimated by correlating some easily obtained metrics from the design of a PCB, and the design time required during the layout stage of development.

The evaluation section reveals how multiple metrics, traditionally used by the design community to estimate design effort, are fairly uncorrelated with actual design time. These include dynamic or static power, logic or storage area, frequency, number of flip-flops and, somewhat surprisingly, the number of standard cells. The number of cells and transistors are two popular design effort estimators used by Sematech and the SIA roadmap. Finally, the evaluation shows that both the productivity adjustment and the *μComplexity* accounting procedure are necessary to produce accurate estimators.

The PCB evaluation shows how simple statistics like the area size and number of components yield some correlation with design effort. With a 90% confidence, pins has a (0.47, 2.09) confidence interval. This means that roughly by looking at the number of pins, the typical design time error is half/double with a 90% confidence. Much better results can be achieved with the proposed *μPCBComplexity* metric. In that case the confidence interval for a 90% confidence is (0.58, 1.72). This roughly means that less than 40% estimation error is done with a 90% confidence.

References

1. Semiconductor Industry Association. International Technology Roadmap for Semiconductors (ITRS), 2002.
2. C. Bazeghi, F. Mesa-Martinez, and J. Renau. μComplexity: Estimating Processor Design Effort. In *International Symposium on Microarchitecture*, Nov 2005.
3. B. Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.
4. M. Chincholkar and J. Herrmann. Modeling the impact of embedding passives on manufacturing system performance. September 2002.
5. E.L. Crow and K. Shimizu. *Lognormal Distributions: Theory and Application*. Dekker, 1988.
6. P. Dasgupta, A. B. Kahng, and S. Muddu. A Novel Metric for Interconnect Architecture Performance. In *Design, Automation and Test in Europe Conference and Exhibition*, March 2003.
7. M. Davidian and M.D. Giltinan. *Nonlinear Models for Repeated Measurement Data*. Chapman & Hall, 1995.
8. T. DeMarco and T. Lister. *Peopleware Productive Projects and Teams*. Dorset House Publishing, 1999.
9. W. Fornaciari, F. Salice, and D.P. Scarpazza. Early Estimation of the Size of VHDL Projects. In *International Conference on Hardware/Software Codesign and System Synthesis*, pages 207–212, Oct 2003.
10. R. Goodall, D. Fandel, A. Allan, P. Landler, and H. R. Huff. Long Term Productivity Mechanisms of the Semiconductor Industry. www.sematech.org, 2002.
11. J.P. Hoffmann. *Generalized Linear Models*. Pearson, 2004.
12. A. B. Kahng. Design technology productivity in the dsm era (invited talk). In *Conference on Asia South Pacific Design Automation*, pages 443–448. ACM Press, 2001.
13. P. Kudva, B. Curran, S.K. Karandikar, M. Mayo, S. Carey, and S.S. Sapatnekar. Early Performance Prediction. In *Workshop on Complexity-Effective Design*, Jun 2005.
14. P. Kudva, A. Sullivan, and W. Dougherty. Metrics for Structural Logic Synthesis. In *International Conference on Computer-Aided Design*, pages 551–556, Nov 2002.
15. R.C. Littell, G.A. Milliken, W.W. Stroup, and R.D. Wolfinger. *SAS System for Mixed Models*. SAS Publishing, 1996.
16. T. Little. Value Creation and Capture: A Model of the Software Development Process. *IEEE Software*, 21(3):48–53, 2004.
17. L. H. Putnam and W. Myers. *Five Core Metrics: The Intelligence Behind Successful Software Management*. Dorset House Publishing, May 2003.
18. Numetrics Management Systems. Key Performance Indicators of IC Development Capability-A Framework. Technical report, Numetrics Management Systems, Inc., 2005. http://www.numetrics.com.
19. The R Development Core Team. *The R Reference Manual - Base Package*. Network Theory Limited, 2005.