VLSI DESIGN
1997, Vol. 5, No. 2, pp. 141–153
Reprints available directly from the publisher
Photocopying permitted by license only

Statistical Module Level Area and Delay Estimation

AKHILESH TYAGI

Department of Computer Science, Iowa State University, Ames, Iowa 50011-1040, USA

The increasing complexity of VLSI design process has led to an increasing use of layout synthesis systems. For many components of a high-level synthesis system such as module generators and module generator development environments, an accurate model of area and delay for the layouts generated by a layout synthesis system is extremely desirable. We have experimented with a statistical model for area and delay of function modules. This model is surprisingly accurate for a standard cell based layout synthesis system—VPNR. The area of adder and shifter modules can be modeled to with in 5% accuracy while the error in delay model is bounded by 4%. This model can be taken through another level of indirection without significant loss in accuracy. The area of all the modules that fit a ripple-template (such as carry-ripple adder) can be modeled with in 30% accuracy. The delay of these modules has a better fit, 15%. The square-template designs (such as array multiplier) have an area model with 1.7% coefficient of variance. In these cases, the model is parametrized by the area and delay of the leaf cells in the template.

Keywords: module generator, module generator, development environment, area and delay estimation, statistical modeling, high-level synthesis

1. INTRODUCTION

Module generators have been used in design synthesis for close to a decade now. High-level synthesis systems have incorporated them in order to generate layouts for datapath modules. They have also been used in stand-alone mode by the system designers. However, the integration of a module generator into a high-level synthesis system introduces an entirely new set of problems. A high-level synthesis system bases many of its crucial tasks, such as scheduling and allocation, on the layout attributes of area and delay of its datapath operators. Hence, we need a model for the area and delay of the designs produced by a module generator, that can be built into the scheduling and allocation algorithms. In general, this model is parametrized by the datapath width n. Note that the accuracy of these models is of paramount importance since it directly affects the quality of the schedule and allocation. The area and delay modeling gets more complicated for a module generator that can build a variety of designs for a datapath function for the same datapath width. This capability is usually provided to enable a module generator user to explore a broad area-time design space. This type of module generator is indispensable in a similar system level design-space search by a high-level synthesis system. An area and/or delay model for these module generators has to incorporate several design types, *i.e.* function implementation architectures.

The problem, then, can be stated as follows. The problem input is a module generator which builds a

function level netlist in response to the datapath width specification. If the module generator incorporates a large design space for the given function, it may either choose the most optimal design for the given datapath width on its own or the user might be required to specify the design type in addition to the datapath width. In either case, the area and delay modeling task has to face the existence of multiple design types. The problem output is a set of equations, parametrized at least by one parameter: datapath width n, that model the area and delay of the designs produced by the given module generator. Note that the model need not be continuous in n, in the sense that a different set of equations could model the behavior of the module generator for n in the range 1–8 and for n in the range 8–16. Hence the model need not be continuous at $n = 8$ in this example.

In this paper, we propose to use statistical modeling as a solution to this problem. In particular, we demonstrate this technique on a specific set of module generators that use netlist leaf cells and build netlist level function designs. The final layout for these module generators is derived by deploying a standard cell based place and route system, *VPNR* [Brglez, Kedem [6]]. The reason behind using these module generators for adders and shifters was that they were developed by the author [15]. The only reason to use VPNR was the easy availability of VPNR expertise and the OASIS compilation environment. Our decision to build module generators for netlist level designs was a pragmatic one. The integration of design-space exploration into a module generator with full-custom layout leaf cells is at best a tedious task. The full-custom layout leaf cells also make the module generator quite inflexible. However, with a netlist leaf cell even the cell area figure A is not available. In fact, the layout area per ripple cell is not even a fixed value A since each leaf cell netlist can be placed and routed differently. One way out of this situation is to see if the area and delay values of the layouts generated by the chosen layout synthesis system for the given module are an acceptably accurate statistical function of a model. We explain the statistical layout area and delay modeling idea further in the following.

The module generators are designed for structured datapath functions such as adders, shifters and RAM. These functions have well-defined analytical area and delay models as a function of datapath width n. For instance, the area of an n-bit carry-ripple adder is expected to be $c_0 + c_1 n$ where c_1 may correspond to the area of the carry-ripple logic cell. For a full-custom carry-ripple adder design we should be able to get a very good area model from the area of the carry-ripple leaf cell, if we are willing to ignore the aspect ratio, by arranging the leaf cells in a linear array. Similarly, we expect the area of an n-bit parallel-prefix adder to be of the form $c_0 + c_1 \log n + c_2 n + c_3 n \log n$. These constants must depend on the layout system—for full custom, there might be a set of constants for each individual designer style; for standard cell based placement and route system, the constants depend on the standard cell sizes and the routing algorithms. For a standard cell based placement and route system, it is not even evident that the layout system would preserve the known analytical behavior. It may introduce dominant area terms resulting from the routing algorithm's idiosyncrasies. The question then is if a layout synthesis system generates layouts whose area values fit a statistical model. We show that for a variety of adder and shifter designs a statistical area model for the VPNR-generated layouts with in a coefficient of variance (*C.V.*) 5% can be developed. The delay models show an even tighter coefficient of variance, 3%. This means that with a high probability (at least .9) the area value predicted by the statistical area model is with in 15% of the correct area value. The delay model is within 9% error with probability at least .9. Section 3 contains the details of the experiments and results.

A natural extension of this question is "what does it take to extend statistical area and delay estimation into *module generator development environment*?". The most common approach to module generator development environment design is to identify the common, basic building blocks and their relationships. The system, then, provides either graphical or proce-

dural means of specifying a function with these blocks and their compositions. For example, a user wishing to build a module generator for an adder will first choose a design to be implemented, say, a carry-ripple adder. S/he would then have to represent this design in the environment's form, *e.g.*, either as a collation of some blocks, or as a class in an object-oriented language. An environment is capable of producing a (module-generator) program from this description. In order to incorporate design-space exploration into the resulting module generators, we need area and delay models for a module type. For a module generator environment we do not even know *a priori* the type of function for which a module-generator would be built. How can we, then, predict area for the layouts to be produced by not-yet-determined module-generator—a capability, seemingly indispensable for a program which can explore design spaces in order to produce a design that matches the area specified by a user in λ^2 units?

The module generator development environments typically rely on a template to represent a type or class of modules. For instance, a ripple template can represent all the ripple designs for adders, counters, parity generators. The particular ripple logic used in each case is different, but the global template is the same, a linear array of cells. A parallel-prefix template can similarly implement parallel-prefix designs for adders, counters and parity generators. In this case, does there exist a good statistical area model which is parametrized by n and the area of the ripple logic cell for the given function? We show in Section 4 that the coefficient of variance varies from 2% to 30% for the area models. The delay models' coefficient of variance is bounded by 15%.

Previous Work

Chen and Bushnell [3] show a methodology to characterize the area requirements of a layout system. Kurdahi and Parker describe a general methodology for estimating the expected area of a netlist in [7]. Wallace and Chandrasekhar [18] give a technology-independent technique to derive a delay model for a set of logic equations assuming an underlying structure for the logic. Ji *et al.* [5] estimate area and delay of an RT level structure given its leaf cells and structural description. But all these methods involve expensive computation for estimating the approximate area for a *random* netlist. In contrast, the module netlists have a very regular structure and their area as a function of the datapath width n is known *a priori*. Gajski *et al.* [[20, 2] considered abstract layout area and delay models for high-level synthesis. The datapath and controller were modeled separately in these models. Kurdahi *et al.* [12] present an area and delay modeling technique for RT-level designs that allows a high-level synthesis system to undertake design trade-offs. Jha and Dutt [4] also develop statistical models for area and delay of RT level components. We do not know of any previous work in the module generator development environment area and delay estimation.

2. STATISTICAL BACKGROUND

In this section, we introduce some statistical concepts. For more details, the reader is referred to Norusis [[10], [11]].

The *linear regression* analysis models a population as a linear function of a variable, $y = b_0 + b_1x$. It determines the values for the coefficients b_0 and b_1 such that the sum of the squares of the distance of the model from the sample data is minimized. This method is also known as the method of *least squares*. The *multiple linear regression* builds a model linear in many independent variables, $y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_kx_k$. Note that any two variables x_i and x_j are independent if their contributions to the value of y are independent. There will be many instances of two variables in this paper which appear to be dependent functionally, such as $x_i = n$ and $x_j = \log n$, but their contributions to y (area or delay) are independent.

Once the coefficient b_i's values have been determined $b_i = \hat{b}_i$, we have a prediction model $\hat{y} =$

$\Sigma_{i=0}^{k} \hat{b}_i x_i$. The predicted value $\hat{y}$ models some population parameter, area and delay of modules in our case. Its accuracy can be characterized by its statistical characteristics such as mean, variance, and standard error. The regression analysis minimizes the prediction error for the mean of the sample data. Hence, for instance, if the input data for area consists of n = 2,4 and 9 then the model is most accurate for n = 5, the mean of 2,4,9. The error in the predicted value attributable to the difference in the input value and the sample mean is called *regression error*. In addition, the actual value y and the predicted value $\hat{y}$ can differ. This difference $ABS(y - \hat{y})$ is referred to as *residual error*.

What parameters can be used to determine how good a fit does the model provide? A commonly used measure of the goodness of fit of a linear model is R^2, the *coefficient of determination*. It can be thought of as the square of the correlation coefficient between input variables x_i's and the dependent variable y. It is also the square of the correlation coefficient between y, the observed value of the dependent variable, and $\hat{y}$, the predicted value from the model, which is $1 - SSE / SST$, where SSE is sum of residual error squares and SST is the sum of residual squares and regression value squares. If all the observations fit the model exactly, the value of R^2 is 1.

Another measure of the goodness of fit is *coefficient of variance* (*C.V.*). It is the ratio of the *standard error of the estimate* and the mean value of the regression expressed as a percentage. The standard error of the estimate corresponds to the standard deviation of the residual error values for a large sample. Hence the *C.V.* value measures the magnitude of error with respect to the expected value of the parameter. The Chebyshev's inequality says that the probability that the value of a random variable falls with in k times its standard deviation of its mean is at least $1 - 1 / k^2$, *i.e.*, $\text{Prob}\{|X - \mu| \leq k\,\sigma\} \geq 1 - 1 / k^2$ where μ and σ are X's mean and standard deviation respectively. Hence a *C.V.* of 5% for an area model, for instance, can be interpreted as "the probability that the area value is within 10% of its mean is at least .75.".

Note that the absolute error in a model could be lower than the coefficient of variance. This is because *C.V.* measures the variance of the whole error sample as a percentage of the mean. However, the points where the error is large may also have a large observed value as compared to the mean and hence the error as a fraction of correct value might be smaller than the *C.V.* value. Hence another measure we report is the largest error.

3. AREA AND DELAY MODELS FOR MODULES

In this section, we build statistical area and delay models for the standard-cell based placement and route system VPNR [Brglez, Kedem][6] for various adder and shifter designs. The statistical area models have coefficient of variance at most 4.8% and the coefficient of determination R^2 at least .99957. This is a surprisingly good fit. This says that we can predict the area within 5% of the actual value. The *RC* delay is dominated by the routing capacitance. The routing resistance is ignored in these calculations (as in RNL [8] or ESIM [14]). Only the device resistance is used to compute the delay values. The routing capacitance is proportional to one dimension of the layout intuitively, which is the square root of the area. It would seem that if the area is tracked so nicely then the wire lengths also might be tracked very closely. We found that the delay can also be estimated within a 2% coefficient of variance. Let us explain our experiments and results at length.

Experiment

We generated a wide variety of n-bit modules for n ranging from 4 to 128. Specifically, the following adder and shifter designs were considered.

1. Carry-ripple adder: expected area c_0+c_1n.
2. Carry-skip adder [9]: expected area $c_0+c_1\sqrt{n}+c_2n$.

3. Carry-select adder [16]: expected area $c_0 + c_1\sqrt{n} + c_2 n$.
4. Parallel-prefix adder [1]: expected area $c_0 + c_1 \log n + c_2 n + c_3 n \log n$.
5. Select-prefix adder [16]: expected area $c_0 + c_1 \log n + c_2 n + c_3 n \log n$.
6. Linear shift register: expected area $c_0 + c_1 n$.
7. Square shifter [17]: expected area $c_0 + c_1\sqrt{n} + c_2 n$.
8 Barrel Shifter [17]: expected area $c_0 + c_1 n \log n + c_2 n^2$.

For each of the above mentioned designs we generated the netlist modules for at least 6 datapath widths, usually 4, 8, 16, 32, 64 and 128 bits. Then the standard cell based system VPNR was used to generate the *magic* layout from these netlists. We constrained all these layouts to place their input/output signals at the borders. We let VPNR choose a squarish aspect ratio. Now we had 6 design points for each expected area profile listed above. We used a statistical package SPSS [11] to perform the multiple linear regression analysis on this set of data. Table I lists all the model equations with corresponding *C.V.* and R^2 values. Recall that the coefficient of variance (*C.V.*) is the error variance as a percentage of the mean of area values. Thus all the designs generated by VPNR were within (*C.V.*) % of the expected area estimate from the prior analysis. A small *C.V.* value signifies a good fit. A high value of R^2, the coefficient of determination, also signifies a good fit. Note that a better R^2 fit need not necessarily guarantee a better *C.V.* fit.

3.1 Observations on Area Models

The area expressions specify the area value in λ^2 units. Note that all the modules in the list have a fairly tight fit, an R^2 value at least .999 and a *C.V.* value at most 4.8%. There are several points to be considered in building such a model.

data set size: In all the models reported, the data set consisted of only 6 data points, for $n = 4, 8, 16, 32, 64$ and 128. This may lead to the question if the results are statistically valid despite the small data set size. The number of independent variables, the number of terms in the model with large coefficients, in all these models seems to be at most four: a constant and 2 or 3 of $\log n, \sqrt{n}$ and n terms. The six data points are certainly sufficient to determine four independent terms. However, it is possible that a larger data set may lead to a better fitting model, perhaps with a lower value for C.V. and a higher value for R^2. The reason that it might not be practical to generate a large data set is the effort involved in generating it. One design with a complete layout has to be built for each data point. It has to be extracted and then simulated for delay. The maximum information about the area and/or delay model is derived from the first 4–10 data points. The incremental improvement to the model beyond that is usually not worth the effort of enlarging the data set. We did increase the data set size for the carry-ripple adder to 30 data points. The resulting model is shown in Figure 4. The new area model is given by$166002n - 2279112\sqrt{n} + 1590468 \log n + 456102$.

TABLE I Area expressions

module	area model (λ^2)	*C.V.*	R^2
Carry-ripple adder	$165735n - 2278948\sqrt{n} + 1603469 \log n + 456210$	1.8%	.99995
Carry-select adder	$223380n - 2939404\sqrt{n} + 1960515 \log n + 858236$	2.05%	.99994
Carry-skip adder	$235316n - 3284777\sqrt{n} + 2313077 \log n + 678056$	3.3%	.99984
Parallel-prefix adder	$51791n \log n - 234586n + 962032$	3.4%	.99969
Select-prefix adder	$597n^2 + 43395n - 250036 \log n + 601928$	2.38%	.99993
Linear shifter	$26752n - 221860\sqrt{n} + 115245 \log n + 130967$	4.8%	.99957
Square shifter	$-19n^2 + 33327n - 86843\sqrt{n} + 101607$	.45%	.99999
Barrel shifter	$2446n^2 + 58669n - 103437$	.39%	1

It's coefficient of variance improves to 1.4% from 1.8% and the R^2 value improves to .99999.

average value of sample datapath width: The residual error is minimum at the average of the sample data. It increases as a function of distance from mean. For instance, for a sample data set with n = 4, 8, 12, 16, 32, the average value of n is *14.4*. Hence this model is most accurate for the values of n in the vicinity of 14. The care should be taken to select the data set in such a way that the average parameter value is closest to the desired range for the model, *i.e.* if it is known that the model would be primarily used for the range n = 32–64 then a data sample with the average value for n = 48 will make sense.

layout synthesis system behavior: The analytical model for a carry-ripple adder's area is plain $c_0 + c_1 n$. However, a regression fit to this model had *C.V.* = 22% and R^2 = .97871. Adding the log n term to the model makes *C.V.* = 10.3% and R^2 = .99688. The introduction of $\sqrt{n}$ term improves it further to *C.V.* = 1.8% and R^2 =. 99995. This observation leads us to the conclusion that VPNR tends to introduce some log n and $\sqrt{n}$ components to the layout area. Note that the initial combined placement and route phase of VPNR is based on a recursive quadrisection [13] algorithm. It is very likely that the overhead of combining two recursive solutions (channel routing between two solutions) adds some area cost not intrinsic to the analytical layout area model. This cost, then, is proportional to the number of recursion stages log n. The $\sqrt{n}$ cost can be due to our insistence on a squarish aspect ratio. Note that a carry-ripple adder is primarily a chain (linear) structure. Folding it around to make it squarish must add some signal routing overhead from border to the internal points. This distance (and hence area) is proportional to $\sqrt{n}$. We tabulate some of these incremental improvements in the model in Table II with the most improved model, shown in Table I.

TABLE II Incremental improvement in the area models

module	Model Improvement term, *C.V.*, R^2
Carry-ripple	n, 22%, .97871
	log n, 10.3%, .99688
Carry-select	n, 22.7%, .97789
	$\sqrt{n}$, 6.4%, .99891
Carry-skip	n, 23%, .9683
	$\sqrt{n}$, 7.8%, .9982
Parallel-prefix	n log n, 15.5%, .99062
Select-prefix	n^2, 7.7%, .99778
	n, 2.5%, .99984
Linear Shifter	n, 12.1%, .99186
	log n, 6.9%, .99824
Square Shifter	n, 5.2%, .99684
	$\sqrt{n}$, .5%, .99998
Barrel Shifter	n^2, 10.7%, .99423

TABLE III Maximum error points for adder and shifter area models

module	maximum error (%) at n' bits	sample data average n
Carry-ripple	5.1%, 4-bit	49
Carry-select	5.3%, 4-bit	49
Carry-skip	8.5%, 4-bit	49
Parallel-prefix	21%, 8-bit	49
Select-prefix	11.3%, 4-bit	49
Linear Shifter	8.5%, 4-bit	24
Square Shifter	.44%, 4-bit	23
Barrel Shifter	1.7%, 4-bit	24

The objective of this discussion is to point out that one has to be extremely careful in understanding the internals of the layout synthesis algorithms in order to determine a good model. From the layout synthesis system's perspective, this analysis points out the area factors attributable to the layout synthesis algorithms.

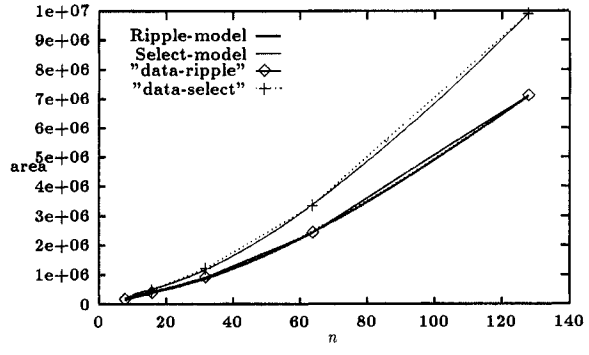

FIGURE 1 Area Model vs Layout Data for Ripple and Select Adders *'Ripple-model' and 'Select-model' are the area models while 'data-ripple' and 'data-select' are the actual layout area values.*

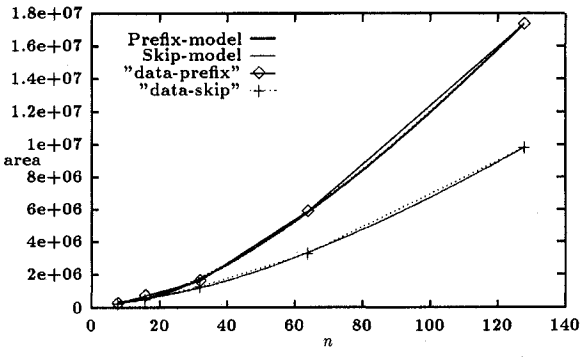

FIGURE 2 Area Model vs Layout Data for Skip and Parallel-Prefix Adders *'Prefix-model' and 'Skip-model' are the area models while 'data-prefix' and 'data-skip' are the actual layout area values.*

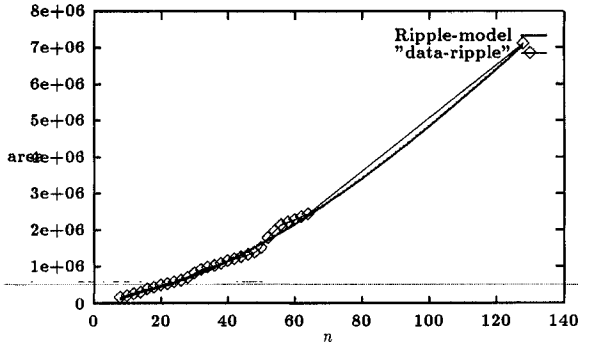

FIGURE 4 Area Model vs Layout Data for Carry-Ripple Adders with Large Dataset *'Ripple-model' is the area model and 'data-ripple is the actual layout area values for carry-ripple adders'.*

absolute errors: Note that the *C.V.* value is making only a statistical statement about the probable values of error. As we pointed out earlier, by Chebyshev's inequality, a 5% *C.V.* implies that the predicted value is within 10% of the observed value with probability at least .75 *or* the predicted value is within 15% of the observed value with probability at least .89. Also note that the errors of this magnitude occur for the values of n away from the sample data average n. For instance, we found the maximum error for a carry-ripple adder between the observed and predicted area values to be at $n = 4$ which was 5.1%. The average value of n for the sample data for the carry-ripple adder was 50. Note that it doesn't mean that the observed value for $n = 50 + 46 = 96$ should also have 5.1% error. In fact, for $n = 128$, the error is only .005%! It only says that the values of n further away from the sample data average n can have large errors. We tabulate the maximum errors for all these designs in Table III. We also show this data graphically for carry-ripple and carry-select adders in Figure 1, for parallel-prefix and carry-skip adders in Figure 2, for select-prefix adder in Figure 3, and for shifters in Figure 5.

The delay values were derived as follows. For all the *magic* layouts generated by VPNR, we extracted the netlists that contain both the device and routing capacitance. This netlist was used in a simulation by RNL [8], an extension of ESIM [14]. It is a switch level simulator based on *RC* delay model. The routing resistance is not considered in the *worst-case* delay calculations, but the routing capacitance is included. These delay figures (in nano-second units) were then used for statistical multiple regression analysis by SPSS in a way similar to the area case.

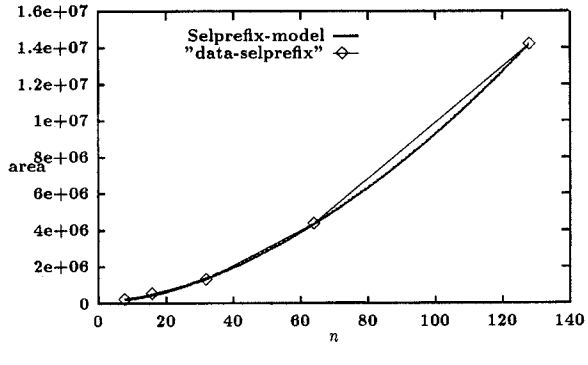

FIGURE 3 Area Model vs Layout Data for Select-Prefix Adders *'Selprefix-model' is the area model and 'data-selprefix' is the actual layout area values for select-prefix adders.*

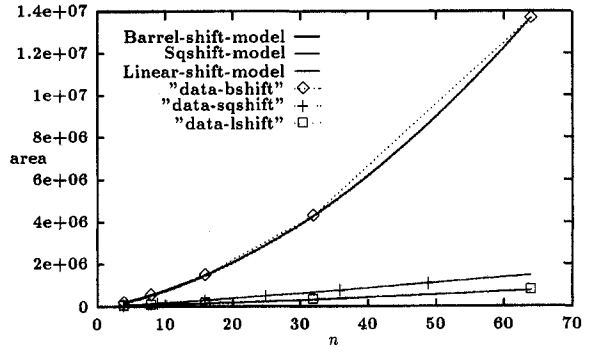

FIGURE 5 Area Models vs Layout Area Values for Barrel, Square and Linear Shifters

Table IV lists the *RC* delay models for all the adder modules. Note that the value of *C.V.* is below 2.1% in all the cases except parallel-prefix adder. We also list the maximum residual error for the delay values in Table V. The maximum error did not exceed 3% in all the cases. Figures 6 and 7 show plots of delay models versus simulation data.

TABLE V Maximum error points for adder delay models

module	maximum error (%) at n' bits	sample data average n
Carry-ripple	1.1%, 4-bit	49
Carry-select	2.9%, 4-bit	49
Carry-skip	2.1%, 4-bit	49
Parallel-prefix	2.1%, 4-bit	49
Select-prefix	.12%, 4-bit	49

3.2 Applicability of Statistical Modeling to Other Layout Methods

In the preceding discussion, we have demonstrated that the area and delay models can be built for our module generators using VPNR generated layout area values. The natural question is how can this technique be applied by some one using a different module generator and layout synthesis method. Here is a summary of steps that can be followed to generate a statistical model.

- First determine the analytical area and/or delay model for the module in question. Note that most datapath functions consist of constant terms, n terms, $\log n$ terms and $\sqrt{n}$ terms. The constant term accounts for any logic that is used only once instead of being repeated in several bit slices. The n term accounts for the repetitive structure with respect to the datapath width. A $\log n$ term could be introduced either by the extraneous routing factors attributable to the recursive nature of the layout synthesis algorithms or it could be intrinsic to the module implementation as in parallel-prefix adder. Similarly, a $\sqrt{n}$ term could either signify the recursive algorithm's contribution or it could be the repetitive structure contribution as in carry-select adder. Hence, a naive model would consist of $c_0 + c_1 n + c_2\sqrt{n} + c_3 \log n$.
- Build the layouts for at least as many values of n as the number of terms in your model. This will usually mean that at least 4 data points need be generated. Determine their area and simulate the designs for their delay values.
- Use a statistical package to perform regression analysis for this model. Ideally, the regression variables should be presented in the order of their contribution to the model. However, their coefficients are not known in advance! In almost all situations, n term carries the highest weight. You will have to choose an order between the remaining terms based on your intuition about the underlying module. In worst case, all the orders (usually less than 6) can be tried. When the introduction of a term into the regression model gives rise to a higher value of the standard error, you probably have a dependent term. A dependent term always improves the R^2 value, but degrades the standard error.

TABLE IV Delay expressions

Module	Time expression (ns)	*C.V.*	R^2
Carry-ripple	$2.23n - 9.6\sqrt{n} + 7.34 \log n - 1.07$	.5%	.99999
Carry-select	$.024n + 4.22\sqrt{n} - 4.85$	1.5%	.99955
Carry-skip	$.39n - 4.61\sqrt{n} + 8.62 \log n - 9.09$	2.1%	.99977
Parallel-prefix	$-.33n + 12.87\sqrt{n} - 9.25 \log n + 3.18$	4.5%	.99854
Select-prefix	$.18n - 1.52\sqrt{n} + 3.46 \log n - .03$	.87%	.99993

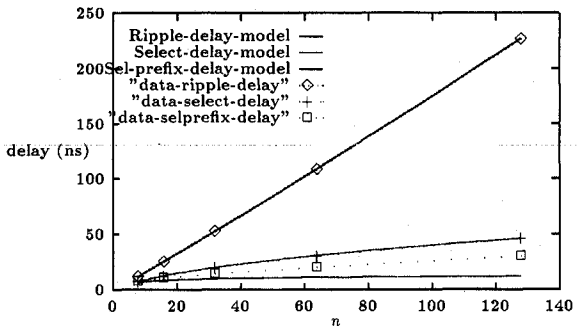

FIGURE 6 Delay Models vs Simulation Data for Ripple, Select and Select-Prefix Adders.

4. MODULE GENERATOR DEVELOPMENT ENVIRONMENTS

In this section, we extend the module level area and delay modeling to module generator development environments. The tricky part here is that the area and delay models are for a template, where the area and delay parameters for the template components are not known in advance. The results are encouraging despite this handicap.

Template Specification

Note that a ripple design has a carry-chain type of computation. We view a ripple template as consisting of the skeleton structure shown in Figure 8. The ripple computation is performed in the R cells. There is one R cell for each bit-slice. Sometimes, the actual input bits to the circuit need to be preprocessed to compute the signal participating in the ripple computation. The I cells at the bottom perform this task. For full generality, the output signals of the R cells are not necessarily the primary outputs of the function. The O cells postprocess the R cell output into the output bits for the function. Note that I and O cells can be very trivial cells (or even null/empty cells as in the case of a parity-generator). Figure 9 shows these cells for an adder.

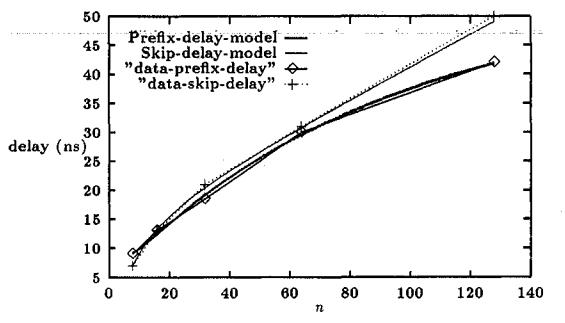

FIGURE 7 Delay Models vs Simulation Data for Skip and Parallel-Prefix Adders.

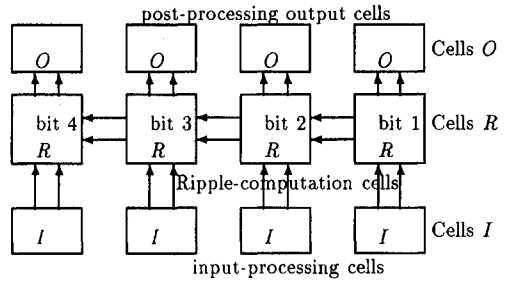

FIGURE 8 The Skeleton Structure of a Ripple Template

Another type of template we consider is the square template as shown in Figure 10. A square design for shifter [17] fits this template with empty B cells. The A cells in the array contain some logic to process the inputs derived from their neighbors. The output of an A cell is also available to its neighbors only. An array multiplier [19] also fits this template with B cells corresponding to adder cells.

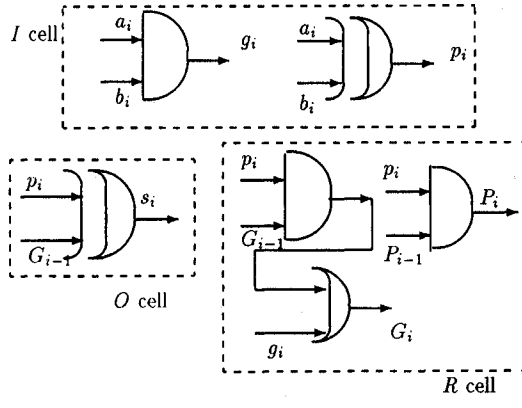

FIGURE 9 An Example of I, R and O Cells for an Adder Generator

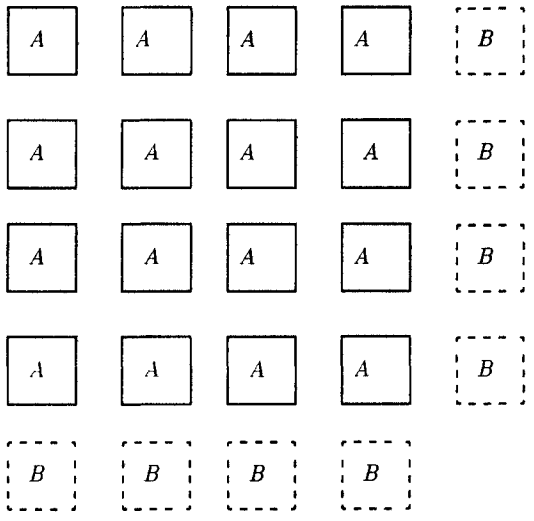

FIGURE 10 A Template for Square Designs

Statistical Area and Delay Models

In this section, we show that the area of the ripple template layouts produced by VPNR follow a statistically derived model with in a coefficient of variance (*C.V.*) of 30%. Their delay has a much tighter *C.V.* bound of 14.4%. The square template design area values, on the other hand, fit with in a *C.V.* of 1.7%. The following experiment was used to determine the area and time models with VPNR. We generated VPNR layouts for three ripple functions (adder, parity-generator and a hybrid counter function) with datapath width n ranging from 8 to 128. The layouts for the three component cells: *R*-cell, *I*-cell and *O*-cell, for each of these three functions were also generated with VPNR to determine their areas A_R, A_I, A_O (in λ^2 units) respectively. Then the expected area of these layouts ought to fit a statistical profile for $c_0 + c_1(A_I + A_R + A_O)n$ for some values of the constants c_0, c_1. Once again, we constrained all these layouts to place their input/output signals at the borders. These layouts were also simulated for *RC* delay with RNL. In addition, the leaf cells *I*, *R* and *O* were also simulated for their *RC* delays to determine T_I, T_R and T_O respectively. Note that the delay of a ripple design should fit the profile $c_0 + c_1(T_I + T_O) + c_2T_Rn$. These area and delay values were put through SPSS multiple linear regression analysis. The models are presented in Table VI. The area model is plotted along with the actual data for ripple designs in Figure 11. The data set in Figure 11 results in R^2 value of .96176. The delay models with the simulation data are displayed in Figure 12. The coefficient of determination, here, is .98101.

We also generated layouts for two square template functions, shifter and multiplier for n = 4, 9, 16, 25, 36, 49. Note that n refers to an n-bit shifter design, but in the case of multiplier, it refers to a $\sqrt{n}$-bit by $\sqrt{n}$-bit multiplier. The layouts for the cells *A* and *B* were produced to determine the cell area values A_A and A_B. Note that for a shifter, $A_B = 0$. The area profile for square template is $c_0 + c_1 A_A n + 2c_2A_B\sqrt{n}$. Table VI shows the area model for the square template, which has a remarkably low *C.V.* of 1.7%. The area models are plotted along with the layout data in Figure 13.

Validation

We ran the following experiment for a validation of the statistical area and delay models. We built a mod-

TABLE VI Area and delay model for ripple and square templates

Template	Area and delay Models	*C.V.*	R^2
Ripple	Area-Model (λ^2)		
	$10.81(A_I + A_R + A_O)n + 125.41(A_I + A_R + A_O)\log n -$ $164.19(A_I + A_R + A_O)\sqrt{n} + 58009$	29.6%	.96176
	Delay-Model (*ns*)		
	$1.344T_Rn + 8.16(T_I + T_O) - 13$	14.4%	.98101
Square	Area-Model (λ^2)		
	$2.12A_An + 2.84A_B\sqrt{n} + .4A_A \log n - 5.34A_A\sqrt{n} + 105882$	1.7%	.99973

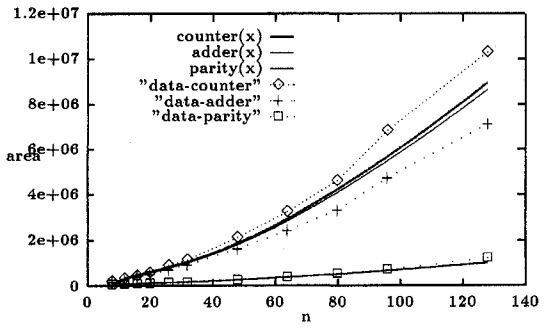

FIGURE 11 Area Model vs Layout Area Values for Ripple Designs.

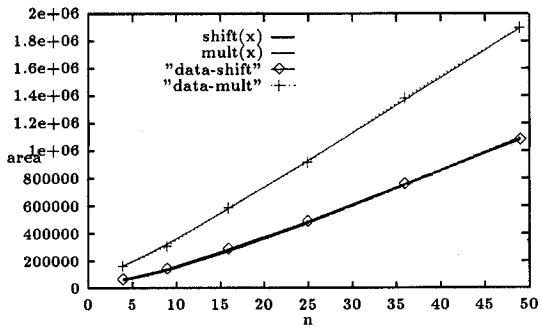

FIGURE 13 Area Models vs layout data for square designs.

ule generator for an artificially defined ripple template function (with two bits in the carry-chain). Let us call this module **func**. The bit-slice for this function **func** is shown in Figure 14. Note that this bit slice has not been optimized for logic minimality. Getting some area and delay figures for a new ripple design is the sole objective of this exercise. We used this module generator with several area and delay specifications and datapath width (n) values to generate 6 **func** designs. The actual area of the layout produced by VPNR from the netlist generated by the module generator was always within 18% of the area estimated by the model. Figure 15 shows a plot of the model-predicted area values vs actual layout data for **func**. The delay figures were even tighter—within 14% of predicted values. Figure 16 displays the delay-model values for **func** along with the simulation delay data.

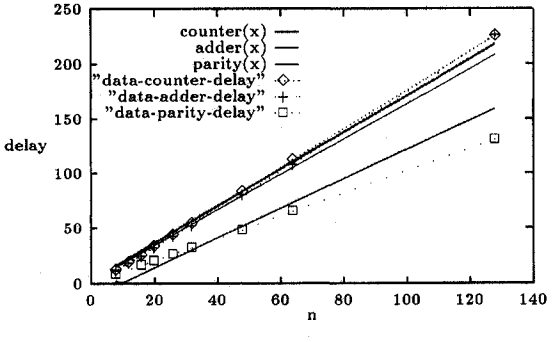

FIGURE 12 Delay Models vs Simulation Data for Ripple Designs.

5. CONCLUSIONS

We presented a statistical technique to build models to predict area and delay for modules. It involves generating layouts for many values of the datapath width n with the layout synthesis method that would be employed in generating layouts from the netlists produced by the module generators. We showed that remarkably tight area and delay models for various adders and shifters can be produced for a standard cell based layout synthesis system—VPNR. The coefficient of variance ($C.V.$) for these designs was at most 5% with a coefficient of determination (R^2) at least .999. The maximum error was less than 10% for every design except parallel-prefix adders. Note that these results do not imply that VPNR is the only *well-behaved* layout synthesis system. Many comparisons at the International Layout Synthesis Workshop have shown time and again that most of the layout synthesis systems tend to produce designs with area values

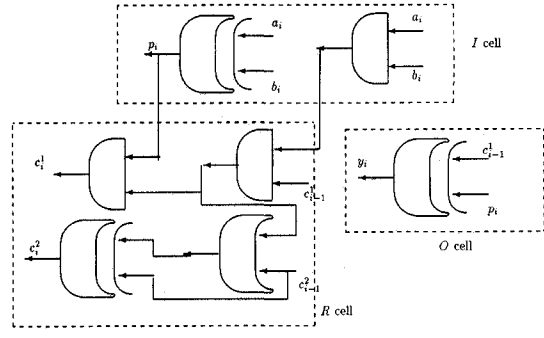

FIGURE 14 Bit Slice for a Ripple Function **func**

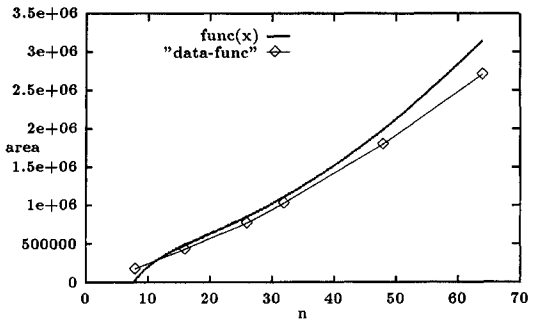

FIGURE 15 Area Model and Layout Data Comparison for Ripple Function **func**.

with in 10% of each other. Hence, it is very likely that such statistical models for area and delay with comparable *C.V.* and R^2 values can be built for most of the layout synthesis systems.

When this technique is tried for area models for templates, instead of modules, the error increases. But, the *C.V.* is still below 30% for ripple template and is only 1.7% for the square template. Given that the exact area of the leaf cells is not known at the time of model-building, we consider this to be an excellent fit. In summary, we believe that statistical area and delay models have acceptable accuracy. The ease of building them makes a case in their favor. The computation for model-building involves generating many layouts and multiple linear regression analysis, which is not excessive. However, the main advantage comes in the computation savings in area and delay prediction, which involves a simple expression evaluation. This makes it especially attractive for design-space exploration, where approximate models (certainly errors less than 10% will do) with quick runtime will allow a high-level synthesis system to undertake extensive design space search.

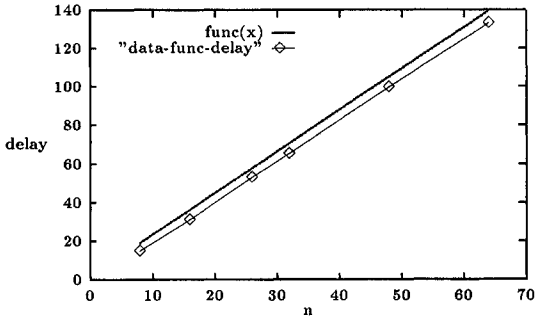

FIGURE 16 Delay Model and Simulation Data Comparison for Ripple Function **func**.

Acknowledgements

This work was partially supported by NSF Grant #MIP-8806169. Arun Rajanala of Cadence ran the initial statistical experiments. This paper has improved due to several comments made by three anonymous reviewers. I thank all these people for their contributions to this work.

References

[1] R. P. Brent and H. T. Kung. A Regular Layout for Parallel Adders. *IEEE Transactions on Computers*, pages 260–264, March 1982.

[2] V. Chaiyakul, A. C.-H. Wu, and D. D. Gajski. Timing Models for High-Level Synthesis. In *Proceedings of EuroDAC-92*, 1992.

[3] X. Chen and M. L. Bushnell. A Module Area Estimator for VLSI Layout. In *Proceedings of 25th Design Automation Conference*, pages 54–59. ACM-IEEE, 1988.

[4] P. K. Jha and N. D. Dutt. A Fast Area-Delay Estimation Technique for RTL Component Generators. In *Proceedings of HLSW-92*, 1992. also appears as UC Irvine, ICS Department TR #92-33.

[5] Q. Ji, Y. S. Oh, M. R. Lightner, and F. Somenzi. Technology Independent Estimation of Area in Logic Synthesis. In *Proceedings of the Synthesis and Simulation Meeting and International Interchange, SASIMI '92*, pages 171–180. Daichisha Press, Kyoto, April 1992.

[6] G. Kedem and F. Brglez. OASIS: Open Architecture Silicon Implementation System. Technical Report MCNC TR 88-06, Microelectronics Center of North Carolina, February 1988. Also appears in Proc IEEE International Symposium on Circuits and Systems, 1990 as *OASIS: A Silicon Compiler for Semicustom Design* by Kedem, Brglez and Kozminski.

[7] F. J. Kurdahi and A. C. Parker. Techniques for Area Estimation of VLSI Layouts. *IEEE Transactions on Computer Aided Design*, 8:81–92, January 1989.

[8] L.I.S. VLSI Design Tools Reference Manual. Technical Report 87-02-01, Northwest Laboratory for Integrated Systems, University of Washington, Seattle, Washington, February 1987.

[9] S. Majerski. On Determination of Optimal Distributions of Carry Skips in Adders. *IEEE Trans. on Electronic Computers*, EC-16:45–58, February 1967.

[10] M. J. Norusis. *SPSS—Introductory Guide: Basic Statistics and Operations*. McGraw-Hill, NY, 1982.

[11] M. J. Norusis. *SPSSX Advanced Statistics Guide*. McGraw-Hill, NY, 1985.

[12] C. Ramachandran, F. J. Kurdahi, D. D. Gajski, A. C.-H. Wu, and V. Chaiyakul. Accurate Layout Area and Delay Model-

ing for System Level Design. In *Proceedings of ICCAD*, pages 355–361. IEEE, 1992.

[13] P. R. Suaris and G. Kedem. A Quadrisection-Based Combined Place and Route Scheme for Standard Cells. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, pages 234–244, March 1989.

[14] C. J. Terman. *Simulation Tools for Digital LSI Design*. PhD thesis, Department of Electrical Engineeing, M.I.T., Cambridge, MA, 1983.

[15] A. Tyagi. An Algebraic Model for Design Space with Applications to Module Generation. In *Proceedings of the First IEEE European Design Automation Conference*. IEEE Computer Society Press, 1990. *Also available as The Dept. of Computer Science, UNC, Chapel Hill, TR89-032.*

[16] A. Tyagi. A Reduced Area Scheme for Carry Select Adders. *IEEE Trans. on Computers*, 42(10):1163–1170, October 1993.

[17] J. D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, Rockville, Md., 1984.

[18] D. E. Wallace and M. S. Chandrasekhar. High-Level Delay Estimation for Technology-Independent Logic Equations. In *Proceedings of ICCAD*, pages 118–191. IEEE, 1990.

[19] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley, Reading, Mass., 1985.

[20] A. C.-H. Wu, V. Chaiyakul, and D. D. Gajski. Layout-Area Models for High-Level Synthesis. In *Proceedings of ICCAD*, pages 34–37. IEEE, 1991.

Author Biography

Akhilesh Tyagi received B.E. (Honors) in Electrical and Electronics Engineering from (1981) Birla Institute of Technology and Science, Pilani followed by M. Tech. in Computer Engineering (1983) from Indian Institute of Technology, New Delhi, India. He received Ph.D. in Computer Science from University of Washington, Seattle in 1988. He was an assistant professor with the Department of Computer Science at the University of North Carolina at Chapel Hill from August of 1987 to June of1993. He is with the Department of Computer Science at Iowa State University, Ames, Iowa since August of 1993. His research interests include VLSI: complexity theory, design, synthesis and architectures.