Rapid Estimation for Parametrized Components in High-Level Synthesis†

Pradip K. Jha and Nikil D. Dutt
Department of Information and Computer Science
University of California, Irvine, CA-92717

Abstract

High-level synthesis tools often use parameterized component generators for describing the synthesized RT design. Rapid area and delay estimation of the instantiated components is necessary for effective component selection and allocation, and is crucial for performing design space exploration. Although constructive methods yield good estimates, they require excessive run times and thus have to be run off-line. In this paper, we describe a fast (on-line) method for estimating the area and delay of regular-structured generic RT components that are tuned to a particular technology library. The estimation models are generated using a least-square approximation on a set of sample data points from selected component implementations. Our approach has been tested against real data points on combinational as well as sequential components and the results show a prediction of the area and delay to within 10% of the actual values. These models have also been integrated with a high-level synthesis system to permit on-line estimation of a component's area and delay.

1 Introduction

Behavioral or High-Level Synthesis (HLS) maps the behavior of a design to a RT structure and a controller that execute the input behavior under user specified design constraints. A major task in HLS is design space exploration in terms of selecting and allocating a proper set of RT components. This task of design space exploration is guided by metrics such as the area and delay of the components. Extensive design space exploration and trade-off analysis between different design alternatives can only be achieved if fast and accurate area and delay estimators exist for evaluating selected components; such estimators are crucial to the success and acceptance of high-level synthesis as a design methodology [GDWL92] [MiLD92].

High-Level Synthesis typically relies on a library of well-defined, parameterized RT component generators to simplify the mapping of behavioral variables and operators to physical components. These parameterized components are used as the building blocks for the tasks of allocation and binding. Each component is customized by parameterized attributes such as the required bit-width and functionality. Such a library of RT component generators provides a complete component set for HLS [Dutt88], and provides a path to physical design through logic and layout synthesis [Dutt91].

Ideally, we would like to store the area-delay attributes for all the component instantiations so as to provide accurate estimates for component selection and binding. However, storage of all components is infeasible since there is virtually an infinite number of possible components that can be generated by varying the parameter attributes. For example, consider the class of ALU component generators that can perform any subset of 26 arithmetic, logic and comparison functions. There are 2^{26} possible combinations of the functions alone. The bit-widths of the two inputs to the ALU can also be parameterized. Furthermore, each ALU component (with a fixed set of parameters) can have multiple implementations (e.g., ripple-carry, carry-lookahead). If we assume that width varies from 1 to 128, and that each ALU component has two possible implementations, we get $2^{26} * 128 * 2 = 1.72 * 10^{10}$ area-delay values for the class of ALU generators alone! Although it may be unrealistic to assume that all possible values of the parameters can be attained, this number is indicative of the large design space of possible components.

A constructive approach is an obvious alternative to storing all the designs and their metrics; we can generate the component implementations on demand, and return the actual area-delay metrics of the constructed

†This research was supported in part by NSF grant #MIP9009239 and in part by SRC contract #91-DJ-146.

design. For example, DTAS[Kipp91] is a system that constructs component designs by decomposing a generic RT component into primitive cells and performing a mapping to a given RTL technology library [DuKi91]. However, such constructive tools have to search a large space of candidate designs and exhibit relatively long run-times, and hence cannot be directly integrated with HLS tools for the task of rapid design space exploration.

We propose a practical solution to this problem by developing a set of estimation functions that generate these area-delay metrics on-line. These estimation functions accept generator parameters such as *input-width* and *list-of-functions* and return area-delay metrics for the component specified by the parameters.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 defines the problem of estimating the area and delay of generic component generators, given a component's parameters and a few technology-specific design data points. Section 4 describes the estimation models we have developed. Section 5 describes the experiments performed to validate our models. We show that our models are simple, fast and provide fairly accurate results (within 10% of actual values). Furthermore, the estimators are integrated with an existing HLS system[LiGa88] and run on-line during the tasks of component selection and allocation. Section 6 concludes with a summary.

2 Related Work

The problem of area and delay estimation has been studied at several design levels and in several contexts. At the level of a complete datapath design, work has been done to predict the area-time tradeoff for a datapath given area and delay values for the primitive modules used to construct the structural design[JaMl88]. Component (module) selection techniques[Jain90] have been proposed to perform area-time analysis for a datapath, given a library of RT components with area and delay characteristics.

At the logic design level, work has been done to predict the area and delay of an RT component, given its structural implementation as a netlist of logic cells or blocks [JOLS92]. Early approaches such as PLEST[KuPa89] analytically estimated the area of a component represented as a netlist of cells, and returned the area as a sum of the component's functional and estimated wiring areas. LAST[ChKu91] and TELE[ChKu92] use a combination of analytical and constructive techniques to predict the area and delay for a netlist of cells, while accounting for wiring. Recent work in layout-based estimators[WCGa91] [CWGa92] also provide area and delay values, given the structure of the modules (components); these approaches take into account technology factors such as layout architecture and wiring.

Purely constructive approaches can be used to estimate the area-delay characteristics of RT components. One constructive method generates Boolean equations for the RT component and represents these in a canonical form to provide area-delay estimates of the component[BRSW87]. Another constructive method builds a tree of possible component implementations by decomposing the component into an interconnection of primitive logic blocks. This large design space can then be pruned and searched for candidate designs. Tyagi[Tyag90] uses an algebraic model for representing the design space and a set of area-delay-power functions to prune design search space. DTAS[Kipp91] also provides functional area and delay values for a component by pruning the design space with a performance filtering function. Although these constructive techniques yield good estimates, they are too slow for direct integration with high-level synthesis tools that require real-time (on-line) estimators.

[WaCh90] proposes a simple technology-independent model for predicting the delay of combinational control (random) logic, given the Boolean equations for the logic (it does not deal with area estimation). Their technique assumes a certain structuring of logic. Consequently, it cannot handle multiple design implementations – a common occurrence for RT datapath components. Although our approach is similar to [WaCh90], we provide both area and delay estimates at a level higher than logic equations.

[Brew88] presents a few simple estimation functions for datapath components, derived from a general structure of the component's implementation and technology-specific values from a databook. However, these estimation models have not been tested against actual designs, and do not account for parameterized multi-function components, or for a range of component implementations.

Previous approaches have thus dealt with estimation either at a higher level (i.e., a complete RT datapath design) or at lower level (i.e., logic equations or structural implementation of a particular component). However, HLS tools require fast (on-line) estimates of area-delay metrics for RT-components to support component selection, scheduling and allocation decisions, as well as design tradeoffs between different component implementations. Furthermore, at the behavioral level, it is convenient to specify an RT-component using a set of parameters for the component's functionality, bit width and other attributes. This convenience comes at cost:

estimation becomes difficult, since the possible space of design implementations is infinitely large. To alleviate this problem, we propose a technique to rapidly estimate the area-delay metrics for such RT-datapath components used in HLS.

3 Problem Definition

We define the area-delay estimation problem in terms of a set of generic component generators, their possible parameters and the estimation functions.

Let G be the set of component generators, P be the set of parameter names and D be the set of domains for the parameters.

- $G = \{G_i | G_i$ is a RT component generator.$\}$
- $P = \{P_i | P_i$ is a parameter.$\}$
- $D = \{D_i | D_i$ is a domain for parameter P_i.$\}$

An ordered set of parameters PG_i is associated with each generator G_i.

- $PG_i = \{P_{i1}, P_{i2}, ..., P_{in}\}$ such that P_{ik} is the k_{th} parameter of generator G_i and there are n parameters associated with G_i.

Each component generator G_i is a function that maps its parameter values PG_i to the instantiation of a specific component. That is, the set of components covered by a generator G_i is obtained by applying the functions G_i on the cross product of all the domains associated with the parameters belonging to G_i.

- $C_i = G_i(D_{i1} \times D_{i2} ... \times D_{in})$ and D_{ik} is the domain associated with k_{th} member of PG_i.

Consider the ALU generator, i.e., let G_i = ALU. This generator has three parameters: input-width, num-functions and function-list, i.e.,

- PG_i = (input-width, num-functions, function-list)

The set of components C_i covered by the ALU generator is given by the cross product of the domains of these parameters. For example, a 4-bit ALU component that can perform two functions: (ADD, SUB) is a member of C_i. This component is specified as ALU(4, 2, (ADD, SUB)).

Each component C_{ij} may have several alternative hardware implementations. Let S_{ij} be the set of implementations for the component C_{ij}.

- $S_{ij} = \{S_{ijk} | S_{ijk}$ is the k_{th} implementation of component C_{ij}.$\}$

Each implementation S_{ijk} has two metrics, area A_{ijk} and delay D_{ijk} associated with it. Thus we have a set of areas and delays corresponding to different implementations of a component.

We develop area-delay models for each component generator G_i based on the area and delay for a subset of components C_{ij} covered by G_i. Let A_{ij} be the set of areas and D_{ij} be the set of delays for the subset of components under consideration. Based on the members of A_{ij}, we propose the area model FA_{ij}. Similarly, based on the members of D_{ij}, we propose the delay model FD_{ij}. These area-delay models FA_{ij} and FD_{ij} predict metrics for the components C_{ik} that are not members of C_{ij} but that are derived from the generator G_i.

Two distinct components C_{ij} and C_{ik} that are derived from the same generator G_i, vary in terms of their values of parameters PG_{ij} and PG_{ik}, respectively. The proposed models should be able to predict the area and delay across a range of values for these parameters. Furthermore, each component C_{ij} has a set of possible implementations S_{ij}, with multiple area (A_{ij}) and delay (D_{ij}) values. For example, an ADDER component has multiple implementations and multiple area-delay values. Hence, we need a set of area-delay models FA_i and FDi to handle multiple S_{ij} and multiple metrics (A_{ij} and D_{ij} for a component C_{ij}). It is obvious that the models FA_{ij} and FD_{ij} should be some function of the parameters PG_i specifying the component C_{ij}:

- $FA_{ij} = a_{ij0} + \sum_k (a_{ijk} \star fA_{ijk}(PG_{ijk}))$
- $FD_{ij} = d_{ij0} + \sum_k (d_{ijk} \star fD_{ijk}(PG_{ijk}))$

The problem thus translates into two steps: formulation of the functions FA_{ij} and FD_{ij}, and determination of the constants and coefficients (a's and d's). In this paper, we briefly describe these steps and provide experimental results to support their validity.

For the rest of the paper, we use the terms RT *component* and RT *module* interchangeably. We also use the term *metrics* to refer to the area and delay values for a component, and *estimates* to denote the estimated values for the area and delay metrics.

4 Area-Delay Models

Our estimation technique uses a sample space of design points on which we perform a least-square regression fit of the formulated functions FA_{ij} and FD_{ij}. We believe that this is a useful approach since designers often store the attributes of commonly occuring designs (e.g., 4-, 8- and 16-bit adders). Since a regression analysis using least-square approximation may not capture the intricacies of certain component implementations, we have to pay attention to the appropriate selection of the sample data points. The following steps summarize our approach:

Step 1 *Generate some real design structures for a component and obtain sample data points for area and delay.*

Step 2 *Study the structure of the design generated, and the the variations of the area-delay metrics with respect to the parameters that define the component.*

Step 3 *Formulate functions for estimating the area and delay of the generator.*

Step 4 *Run least square approximation to calculate the constants (a_{ij0} and d_{ij0}) and the coefficients of various terms (a_{ijk} and d_{ijk}) used for the functions modeling the metrics.*

Step 5 *Test the estimation model.*

We test the accuracy of the results against a user-specified error bound. If our models do not satisfy the user-specified error bounds, we go through an iterative experimentation phase, where we repeat some of the Steps 1-5 above. We note that the error bound is often satisfied by the simple addition of linear and log factors to the estimation functions, as suggested by the design's structure. In an extreme case when the error bound is not satisfied, we may have to generate more implementation data points and repeat all of the steps to obtain new coefficients. However, in our experiments, we have observed convergence within one or two iterations for an error bound of 10%.

In the rest of this section, we briefly describe the formulation of the models and the determination of the coefficients. [JhDu92] contains a detailed description of our approach.

4.1 Formulation of Estimation Models

Recall that the area-delay estimation models FA and FD for a generator G are functions of its parameters PG. One significant parameter for a generator is the component's bit-width, which clearly has a significant impact on the estimation models. Tyagi [Tyag90] has developed an information-theoretic model to understand the relationship between a module's parameters (e.g., bit-width) and its performance metrics. Based on the communication between n-bit slices of of a component, he classifies some combinational components into categories and provides a formulation of models for each category. These models describe the asymptotic behavior of area and delay with respect to major parameters of the generators. For example, both the area and delay of a ripple-carry adder vary linearly with respect to the bit_width of the inputs.

We use a similar formulation for the primary factors of the area-delay models that account for the major contribution towards the performance metrics of a component. In addition to these primary factors, we add some secondary factors that are decided by some rules of thumb and by the structure of the components.

For example, consider the class of Multiplexer generators that has two parameters: num-inputs(ni) and input-width(iw). The area per bit of a multiplexer grows linearly with ni. This constitutes the primary factor in the area model of multiplexer. A constant and a logarithmic term with ni form the secondary factors. The delay of a multiplexer has logarithmic behavior with respect to ni (primary factor), and we add a constant and a linear term as secondary factors. Since we know that the area of multiplexer is directly proportional to iw and that its delay is independent of iw, we get the following area-delay models for a parameterized multiplexer:

$$Area = iw * (a_1 + a_2 \star log_2\lceil ni \rceil + a_3 \star ni)$$

$$Delay = d_1 + d_2 \star log_2\lceil ni \rceil + d_3 \star ni$$

It is interesting to note that at a first glance, certain components may not seem to have a variation with respect to the component's bit-width (e.g., bitwise AND). However, when the components are laid out, we find that the routing effects begin to appear for larger bit-widths. We therefore need to add a function of the bit-width for these components also.

4.2 Estimation Model Generation

We use a least square approximation method on a set of sample design implementations, to determine the coefficients of the formulated estimation equations. We therefore need to address the following questions: (1) Which of the infinitely many possible components for a generator should be considered for calculating the coefficients? (2) For each selected component, how do we actually obtain the actual area and delay values?

The set of components that are to be used for calculating the coefficients should be fairly representative of all the possible components for a generator. We attempt to select the set of components so as to avoid a bias towards any particular parameter or any particular subset of values for a parameter. For example, several components that are implemented as tree-based

Gates	*Percentage error*			
	Area		*Delay*	
	Average	*Max*	*Average*	*Max*
AND	7.14	19.35	2.32	4.96
OR	4.19	13.92	0.99	3.38
NAND	2.49	9.93	5.65	21.88
NOR	4.34	13.13	5.72	14.86
XOR	0.00	0.00	0.00	0.00
XNOR	0.00	0.00	0.00	0.00

Table 1: Results for Logic gates compared to DTAS

logic structures show distinct behaviors for *num_inputs* that are powers of two. Consequently, we often use the following values for the parameter *num_inputs* : $(2, 3, 4, 5, 6, 7, 8, 12, 15, 16, 32, 48, 64)$.

The real area-delay values can be obtained by physically laying out each component and measuring the actual area and delay. Although this provides very accurate measures, it is a very time consuming process. Since our approach needs a substantial number of a component's area-delay values (on the order of 15), we use existing tools to provide these metrics.

DTAS [Kipp91] is one such tool that provides performance metrics. DTAS maps RT-level design components from the GENUS generic component library [Dutt88] to technology-specific library cells and macros. Given the specification of a component, DTAS generates a set of alternative designs corresponding to different design decompositions using the primitive building blocks. The area provided by the DTAS is the sum of the functional areas of the building blocks used in the design implementation. DTAS computes the delay values for all critical paths through a design implementation (pin-to-pin, rising and falling). This calculation takes into account the fan-out of the designs. Thus, DTAS provides a good source of sample design points for our approximations, using functional area and delay values.

To incorporate wiring effects, we use LAST[ChKu91] and TELE[ChKu92], which are estimators that provide more accurate metrics by considering not only functional blocks but also the wiring contributions. Since LAST and TELE have been benchmarked against actual layouts produced by commercial tools as well as against custom-designed layouts, they provide a good source of sample design points for area-delay values that include routing information.

5 Experiments and Results

Parameters		*Percentage error*	
Bit-width	*#Functions*	*Area*	*Delay*
6	4	0.01	2.96
2	8	0.65	1.99
20	2	17.31	13.99
6	12	7.48	5.14
30	13	6.93	4.39
26	7	1.48	4.90
19	12	7.73	3.64
12	14	4.69	3.70
16	16	1.64	3.03
Average error		5.33	4.86
Maximum error		17.31	13.99

Table 2: Results for Logic unit as compared to DTAS

Parameters		*Percentage error*	
Bit-width	*#Functions*	*Area*	*Delay*
7	6	7.3	0.6
11	4	3.5	0.9
10	1	2.3	5.1
11	2	4.8	2.4
15	3	6.2	0.1
14	3	4.6	8.5
10	5	3.6	0.5
5	4	10.2	4.9
6	3	7.9	3.6
Average error		5.6	2.96
Maximum error		10.2	8.5

Table 3: Results for Comparator as compared to DTAS

Parameters		*Percentage error*	
Num-inputs	*Bit-width*	*Area*	*Delay*
7	2	9.65	2.96
21	8	10.11	1.70
19	30	17.97	0.03
18	25	13.70	0.82
23	25	2.12	3.32
24	25	1.22	2.97
20	5	14.11	0.87
14	5	5.13	0.51
3	5	8.46	22.71
12	27	1.51	0.12
9	27	15.83	1.77
2	27	4.22	4.80
8	7	5.18	9.21
21	7	10.11	1.70
30	7	3.74	6.17
Average error		8.20	3.98
Maximum error		17.97	22.71

Table 4: Results for Mux as compared to DTAS

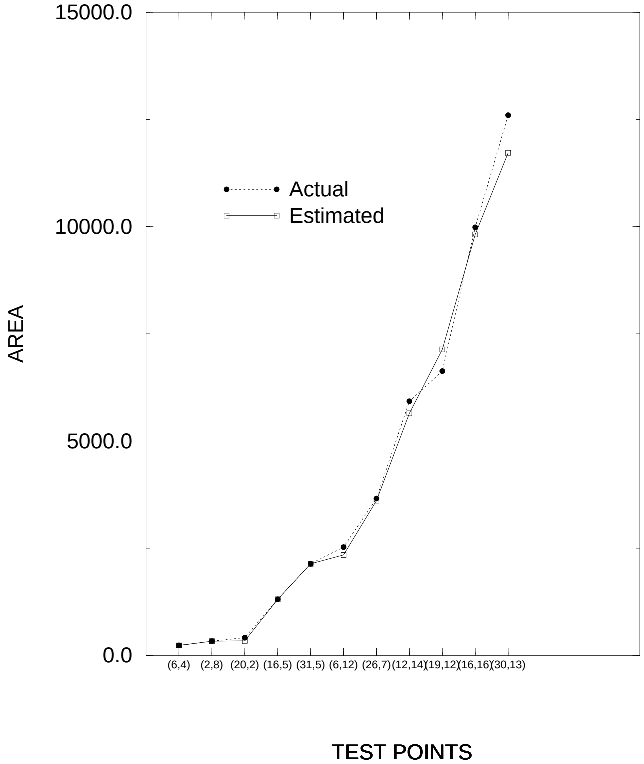

Figure 1: Comparison of actual and calculated area for Lu as compared to DTAS

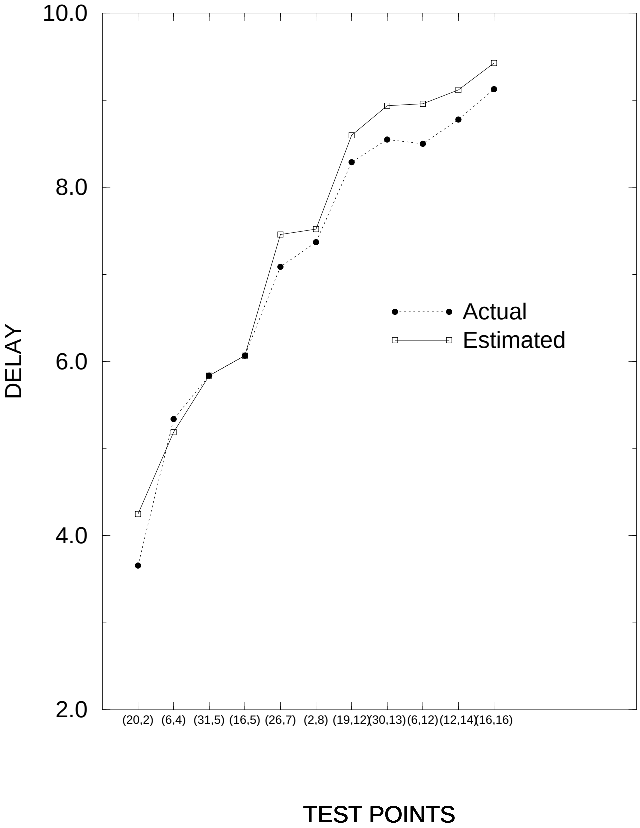

Figure 2: Comparison of actual and calculated delay for Lu as compared to DTAS

Parameters		*Percentage error*	
Num-inputs	*Bit-width*	*Area*	*Delay*
10	11	5.88	31.42
7	12	4.86	26.12
9	16	5.37	26.14
4	1	11.19	10.17
14	21	23.17	6.47
7	24	1.70	20.12
4	35	0.07	11.23
9	44	0.11	13.38
6	47	8.77	15.59
Average error		8.31	18.68
Maximum error		23.17	31.42

Table 5: Results for generic And gate as compared to LAST/TELE

Parameters		*Percentage error*	
Num-inputs	*Bit-width*	*Area*	*Delay*
25	2	1.03	7.36
2	7	7.20	2.18
3	5	15.98	17.43
5	14	2.47	17.33
5	20	0.84	8.15
7	21	1.69	6.49
7	30	16.88	2.91
7	8	2.16	9.65
Average error		6.03	8.94
Maximum error		16.88	17.43

Table 6: Results for Mux as compared to LAST/TELE

Parameters		*Percentage error*	
Bit-width	*#Functions*	*Area*	*Delay*
39	1	1.92	7.98
60	2	0.00	2.64
30	1	1.31	4.80
48	2	0.21	8.62
50	1	3.72	3.42
47	1	0.07	4.36
36	3	3.60	14.43
24	2	0.21	12.70
20	2	1.33	13.82
3	2	13.70	16.30
Average error		2.61	8.91
Maximum error		13.70	16.30

Table 7: Results for Shift-register as compared to LAST/TELE

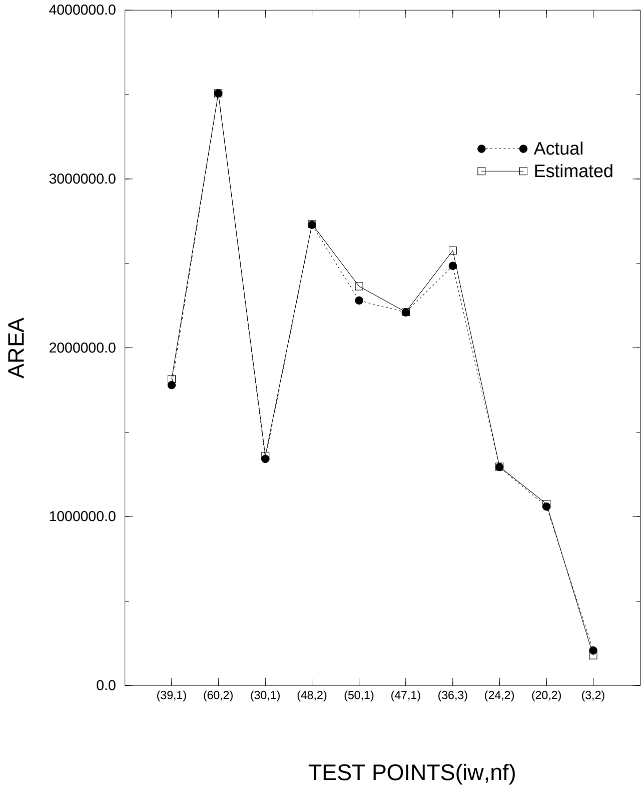

Figure 3: Comparison of actual and calculated area for Shift-register as compared to LAST/TELE

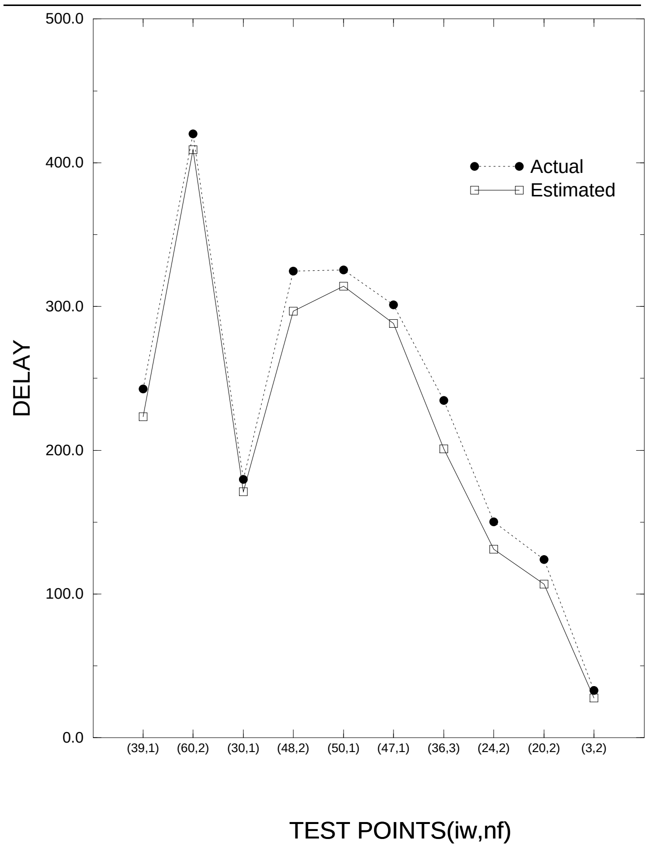

Figure 4: Comparison of actual and calculated delay for Shift-register as compared to LAST/TELE

In this section, we describe the experiments performed to test our models. We compared the estimates generated by our model against the metrics derived from the design structures generated by DTAS [Kipp91], LAST [ChKu91] and TELE [ChKu92]. The area values provided by DTAS counts the number of equivalent two-input NAND gates used to implement the component. For a component's delay (measured in nanoseconds), DTAS returns the worst-case delay for all paths through the design. LAST and TELE provide area and delay values respectively, based on GDT 3μ CMOS standard cell technology.

Our experiments attempted to cover a wide range of possible component implementations. We did this by generating parameter values randomly for each component generator. The number and set of functions (for multi-function components) were also chosen randomly. For each such randomly chosen component, we ran our models, and compared the results with an actual design generated by above mentioned tools.

We considered the following generators: And, Or, Nand, Nor, Xor, Xnor, Mux, Comparator, Lu, Alu and Shift-register . Two parameters are needed for the gates and multiplexer components: num-inputs and input-width. The Comparator, Logic unit, Alu and Shift-register require not only the input-width, but also a set of functions. We first present the data in Section 5.1, and then provide an analysis in Section 5.2.

5.1 Results

Tables 1, 2, 3, 4 and 8 summarize the results of our approach as compared to DTAS. For each data point, we report the percentage error for area and delay. The percentage error is defined as :

$$Percentageerror = \mid \frac{estimated - actual}{actual} \mid * 100$$

Table 1 lists the percentage errors for six gate generators. For the sake of brevity, we show the aggregate result for each gate, instead of the errors for each test point. Both the average and the maximum percentage errors are shown. For other generators, the error per test point, average and maximum errors are reported. Figure 5 depicts the aggregate results for all the components. A complete set of results is too long to be included in this paper, but can be found in [JhDu92].

Table 8 enumerates the percentage error for each of the designs for an ALU component. For each ALU component with fixed parameters, we generated two-to-three alternative design implementations. The average percentage error over all the designs for each of ALU component is also listed.

Figures 1 and 2 present a graphical plot of the actual and estimated area and delay for the Logic Unit (LU) generator with respect to DTAS. The X-axis indicates different test points with increasing values of area or delay. Each test point on the X-axis is a 2-tuple (ni, nf), where ni represents the parameter num-inputs and nf represents the parameter num-functions for a LU component. Similarly, Figures 3 and 4 compare the actual and estimated area and delay for the Shift-register with respect to LAST/TELE.

Tables 5, 6 and 7 describe the results as compared to LAST and TELE. Table 5 lists the percentage error per test point for generic And gates, Table 6 for the Multiplexer generator and Table 7 for the Shift-register.

5.2 Analysis

Tables 1, 2, 3, 4 and 8 show that the maximum average error for all the generators in our study is 8.20% for area and 6.86% for delay. Thus on the average, the data generated by our model is within ±10% of the values generated by DTAS using the LSI library[LsiC87]. The maximum percentage error over all generators is 19.87% for area and 22.71% for delay. However, very few data points reach these extreme values: our low averages for the errors reflect this fact.

Figure 5 shows the aggregate error profiles for area and delay. For roughly half the data points, the error is less than one percent. After this huge concentration in the range 0-0.99, the frequency of error tapers off as the error increases. For area, *85% percent of the test points have error less than 10 percent* and *95% test points have errors less than 15 percent.* For delay, figures are 91% and 97% respectively. These results validate our hypothesis about the goodness of our estimators.

Table 8 reports the errors associated with each implementation for each ALU test point. This table verifies our claim that our approach can handle multiple implementations of a given component.

Figures 1 and 2 illustrate the fidelity of our estimates for the LU generator with respect to DTAS. Similarly, Figures 3 and 4 depict the fidelity of our estimates for a sample sequential component, the Shift-register, with respect to LAST/TELE. We see that the estimated area and delay follow the actual values very closely for different classes of generators (combinational and sequential), as well as with respect to two different backend tools. We observe that the absolute error values are larger for the data points with bigger metric values. This phenomenon results in a relatively uniform percentage error over different test points. We also observed similar results for the other generators.

Tables 5, 6 and 7 show the estimation results with respect to LAST/TELE, which includes the functional and routing components. We observe that the average error for area is between 6-9%.However the average error for delay has increased to 9-19%. This can be attributed to the effects of wiring delay which are sensitive to physical design characteristics such as placement and routing, and that are difficult to capture in general.

We conclude our analysis with two important observations. First, our test points were generated randomly (i.e., we randomly selected the component generator parameter values). Note that in a real design situation, components with certain design properties (i.e., parameters) will be invoked more often, and can be stored with precomputed metrics in the component database. This will lower our average error. Second, our estimation models are integrated on-line with HLS tools. This is possible because of the simple estimation functions chosen, which use only a few additions, multiplications and logarithmic operations. We thus tradeoff accuracy of the metrics (i.e., ±10%) for real-time evaluation of the estimates.

6 Summary

In this paper, we presented an accurate on-line method for estimating the area and delay of RT component generators used in RT and HL synthesis. Our approach can handle the area/delay contributed by functional blocks as well as the total area/delay including the wiring. Furthermore, we have demonstrated the estimation technique on both combinational and sequential RT components. The experiments show very good results, with aggregate errors in the range of ±10%. Our area-delay models are simple, fast and fairly accurate, and have been integrated with an existing high-level synthesis system [LiGa88] [RaGa91]. Although our experiments were based on area/delay values generated by previously benchmarked tools and not by actual layouts, we believe that the estimation approach we presented is general and that it can produce even better results if provided with more accurate sample design data points.

7 References

[Brew88] Forest D. Brewer, "Constraint Driven Behavioral Synthesis," *Ph.D. Dissertation, University of Illinois, Urbana-Champaign,* May, 1988.

[BRSW87] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli and A. R. Wang, "MIS : A multiple-level logic optimization system," *IEEE Trans-*

action on Computer-aided Design, pp1062-1081, November 1987.

[ChKu91] F. J. Kurdahi and C. Ramachandran, "LAST: A Layout Area and Shape function esTimator for High Level Applications," *Proc. of The European Conf. on Design Automation'91* pp351-355, February 1991.

[ChKu92] C. Ramachandran and F. J. Kurdahi, "TELE: A Timing Evaluator using Layout Estimation for High Level Applications," *Proc. of The European Conf. on Design Automation'92,* March 1992.

[CWGa92] V. Chaiyakul, A. C-H. Wu and D. D. Gajski, "Timing Models for High-level Synthesis," *Proc. of The European Design Automation Conference'92,* September 1992.

[DuKi91] N. D. Dutt and J. R. Kipps, "Bridging High-Level Synthesis to RTL Technology Libraries," *Proc. 28th Design Automation Conference,* June 1991.

[Dutt88] N. D. Dutt, "GENUS:A Generic Component Library for High Level Synthesis," *Technical Report 88-92, University of California at Irvine,* 1988.

[Dutt91] N. D. Dutt, "Generic Component Library Characterization for High Level Synthesis," *VLSI Design '91: The Fourth CSI/IEEE International Symposium on VLSI Design,* New Delhi, 1991.

[GDWL92] D. Gajski, N. Dutt, A. Wu and S. Lin, "High-Level Synthesis: Introduction to Chip and System Design," *Kluwer Academic Publishers* 1992.

[Jain90] R. Jain, "MOSP: Module Selection for Pipelined Designs with Multi-cycled Operations," *Proc. IEEE Int. Conf. on Computer-aided Design'90,* pp212-215, November 1990.

[JaMl88] R. Jain, M. J. Mlinar and A. C. Parker, "Area-Time Model for Synthesis of Non-Pipelined Designs," *Proc. IEEE Int. Conf. on Computer-aided Design'88,* pp48-51, November 1988.

[JhDu92] P. Jha and N. D. Dutt, "A Fast Area-Delay Estimation technique for RTL component generators," *Technical Report 92-33, University of California at Irvine,* April 1992.

[JOLS92] Q. Ji, Y. S. Oh, M. R. Lightner and F. Somenzi, "Technology Independent Estimation of Area in Logic Synthesis," *Proc. Synthesis and Simulation Meeting and Inter. Interchange (SASIMI92),* pp171-180, April 1992.

[Kipp91] J. R. Kipps, "An Approach to Component Generation and Technology Adaption," *Ph.D. Dissertation, University of California at Irvine,* December 1991.

[KuPa89] Fadi J. Kurdahi and A. C. Parker, " Techniques for Area Estimation of VLSI Layouts," *IEEE Transactions on Computer-aided Design,* pp81-92, January 1989.

[LiGa88] J. S. Lis and D. D. Gajski, "Synthesis from VHDL," *Proc. IEEE Int. Conf. on Computer Design'88,* pp.378-381, 1988.

[LsiC87] "LSI, CMOS Macrocell Manual," *LSI Logic, Inc., Milipitas, CA,* 1987.

[Mano88] M. M. Mano, "Computer Engineering Hardware Design," *Prentice-Hall, Inc. New Jersey,* 1988.

[MiLD92] P. Michel, U. Lauther and P. Duzy (Editors) "The Synthesis Approach to Digital System Design," *Kluwer Academic Publishers* 1992.

[RaGa91] L. Ramachandran and D. D. Gajski, "An Algorithm for Component Selection in Performance Optimized Scheduling," *IEEE International Conference on Computer-Aided Design,* pp92-95, November 1991.

[Tyag90] A. Tyagi, "An Algebraic Model for Design Space with Applications to Function Module Generation," *Proc. of The European Conf. on Design Automation'90,* pp114-118, March 1990.

[WaCh90] D. E. Wallace and M. S. Chandrasekhar, "High-level Delay Estimation for Technology-Independent Logic Equations," *Proc. of IEEE Int. Conf. on Computer-aided Design'90* pp118-191, November 1990.

[WCGa91] A. C-H. Wu, V. Chaiyakul and D. D. Gajski, "Layout-Area Models for High-Level Synthesis," *Proc. of IEEE Int. Conf. on Computer-aided Design'91* pp34-37, November 1991.

[Wolf89] Wayne H. Wolf, "How to Build a Hardware Description and Measurement System on an Object-Oriented Programming Language," *IEEE Transactions on Computer-Aided Design'89* pp288-301, March 1989.

Parameters		*Percentage error*							
Bit-width	*#Functions*	*Design1*		*Design2*		*Design3*		*Average*	
		Area	*Delay*	*Area*	*Delay*	*Area*	*Delay*	*Area*	*Delay*
32	9	14.65	4.34	14.18	0.54	14.16	1.17	14.48	2.02
7	7	10.93	8.34	6.38	13.73	6.30	16.00	7.48	12.65
2	12	5.45	1.52	5.43	1.74			5.44	1.63
32	7	13.64	6.14	12.79	8.70	12.75	8.27	13.06	7.70
31	10	5.40	1.90	5.25	0.46	5.24	1.53	5.30	1.29
24	5	7.71	3.45	6.87	7.41			7.30	5.44
28	6	4.69	5.55	4.33	9.66			4.46	7.00
30	5	13.22	4.91	11.96	6.41	13.10	8.06	12.32	6.13
17	8	1.24	7.12	1.18	16.39			1.21	11.76
4	10	4.79	9.66	4.71	10.44	4.69	10.49	4.73	10.20
Average error								7.58	6.58
Maximum error								14.65	16.39

Table 8: Results for Alu as compared to DTAS

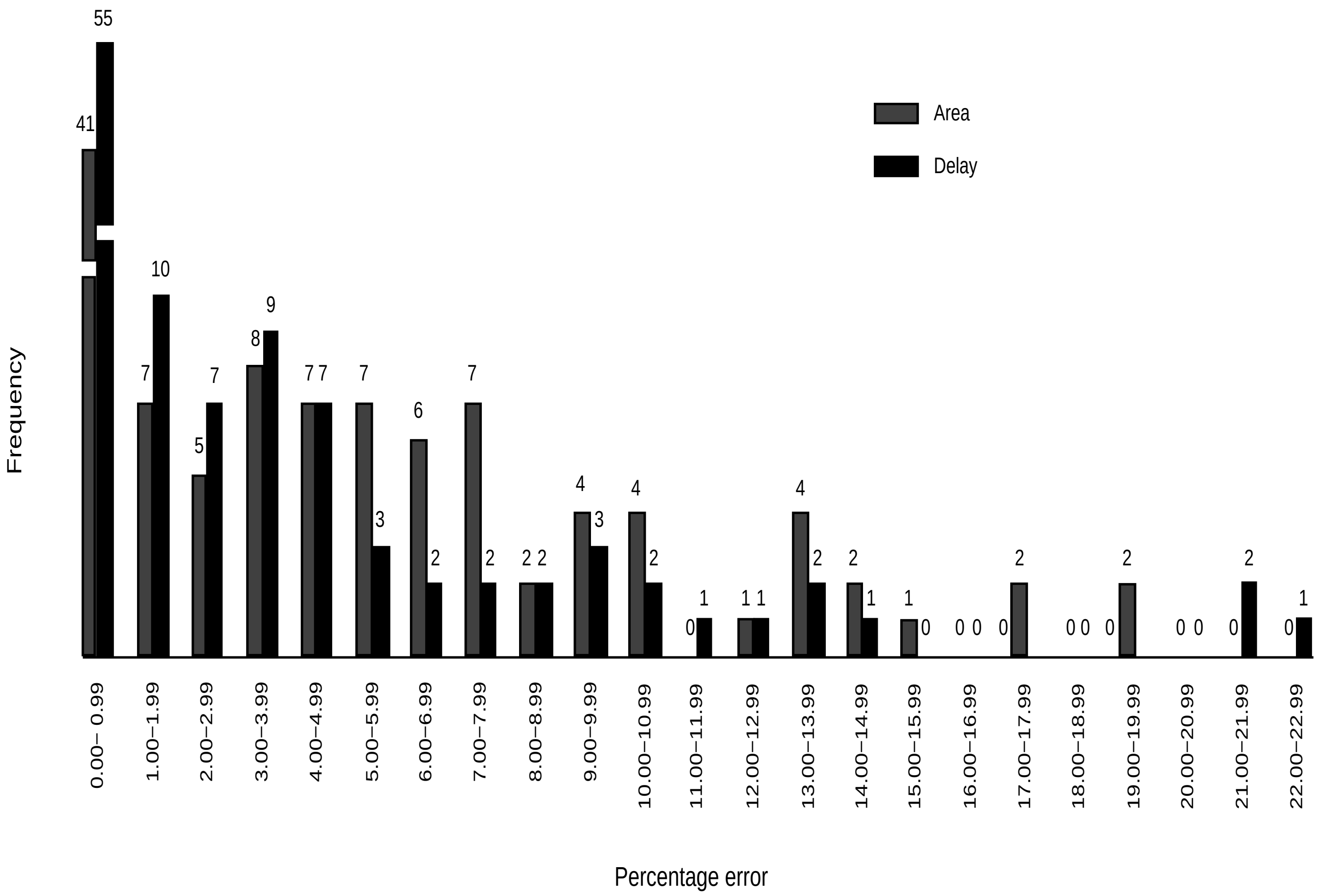

Figure 5: Aggregate Percentage Error as compared to DTAS