The Effects of Variations in Component Styles and Shapes on High-Level Synthesis

Pradip K. Jha[‡] Champaka Ramachandran[†]
Nikil D. Dutt[‡] Fadi J. Kurdahi[†]

Technical Report #92-115
December 18, 1992

[†]ECE Department, [‡]ICS Department,
University of California, Irvine
Irvine, CA 92717

Abstract

High-level synthesis (HLS) has long relied on point models *for RT-components that assume fixed area and delay values for a given component style. However, aspect ratio variations alone can result in substantially different area-delay characteristics for a component. In this work, we explore the combined effect of style and aspect ratio variations on the area and delay of individual RT-components, as well as on complete RT-level designs produced by HLS. We describe the results of extensive experiments which indicate that point models are inadequate for use in the synthesis process. We believe that our results have some deep implications on the formulation of HLS algorithms that attempt to realistically incorporate physical design information early in the design process.*

Contents

List of Figures

List of Tables

1 Introduction

High-Level Synthesis (HLS) typically uses generic RT components during the tasks of scheduling, allocation and binding. The use of generic components simplifies HLS algorithms and standardizes the output of HLS to a common component set, so that these these generic components can then be implemented in a particular technology through RT-component synthesis (e.g., logic synthesis, technology mapping and physical design). Consequently, during HLS, each generic component is assumed to have a rather simple model for its design attributes, usually characterized by a single set of quality measures such as component area and delay.

However, for each such generic RT component, there is tremendous variation in the attributes based not only on the target technology chosen, but also based on the physical design of each implementation. For high-level synthesis algorithms (e.g., scheduling, allocation and binding) to make effective decisions that eventually result in high-quality layouts, we need to incorporate physical design information during HLS. Indeed, we must account, not only for place and route effects, but also global considerations such as RT wiring, component styles, aspect ratio, floorplanning, and the combination of "all of the above". This is illustrated in Figure 1 which shows the sources of design variations during the different design steps. Based on these observations, a high-quality component model must specifically:

1. *capture the variations in implementation styles of a given RT level component in an efficient and structured manner.* The work on component libraries is of importance in establishing a realistic model of hardware since it encompasses a number of considerations such as word versus bit level implementation (i.e. word delay/area may not be a multiple of bit delay/area), alternative implementations and design styles, and control signal requirements. These considerations are generally overlooked by most high-level synthesis systems but can have a significant impact on the final implementation,

2. *realistically and accurately account for the layout effects on area and delay.* The layout considerations affecting area and delay include wiring, floorplanning, fanin, and fanout effects. This requires a good approximation of the physical design topology. In other words, the model should reasonably reflect the relative locations of the various constituents in a component layout.

3. *allow for uncertainties in the shapes of components.* Eventually, these components will be used in a design layout and the final shape of each component will be determined by a global floorplanning step which chooses an "optimal", or preferred aspect ratio of that component so as to efficiently pack the overall design layout. Since the precise shape of the final component layout will not be determined until after the synthesis tasks have concluded, the uncertainty in their layout configurations should be accounted for by allowing the final shape(s) to be

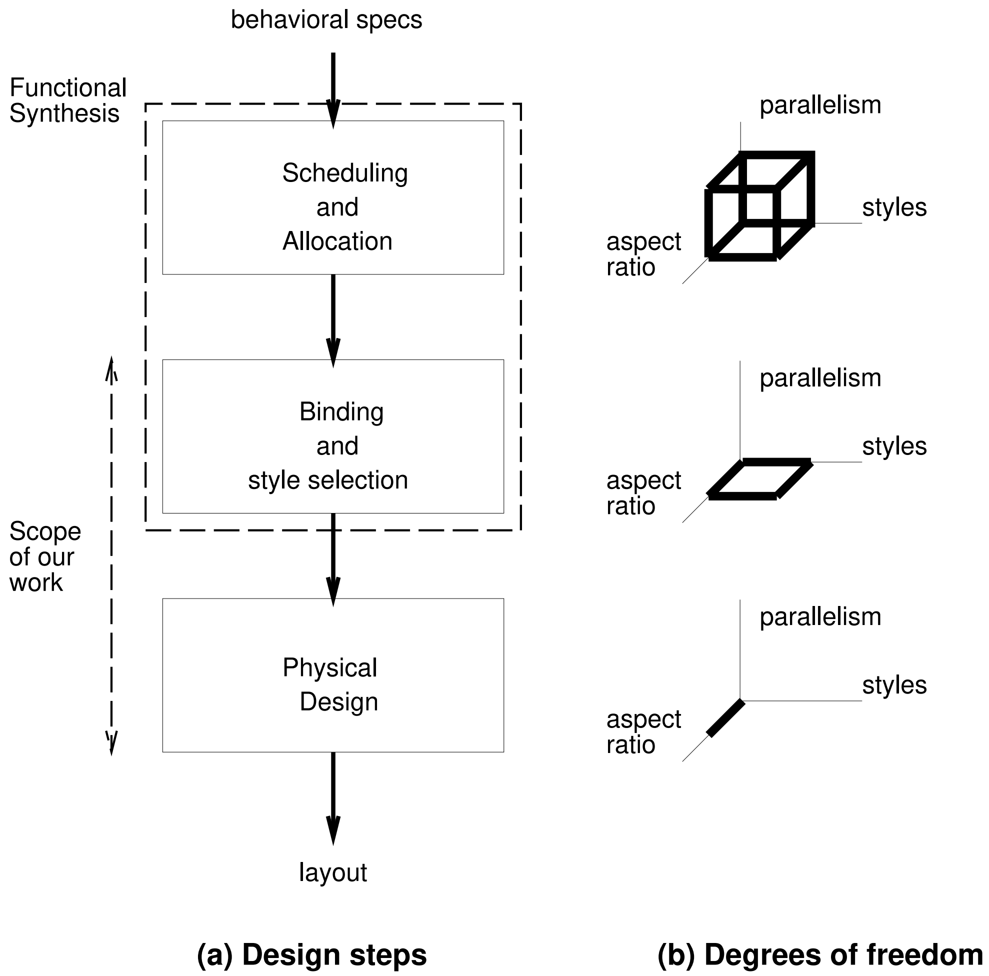


Figure 1: Sources of variations (or degrees of freedom) in design implementations.

approximated by a shape function, rather than by a single design point.

4. *be efficient to evaluate.* The design model must not be too costly to evaluate. In extreme cases, the physical design process can be carried out in full and an exact model can thus be realized. Obviously, this is not practical as one can easily see that generating layouts for a large number of different layout configurations of different design styles of a given RT level component could easily take enormous amounts of time, especially when extensive delay characterization and analysis is applied through timing analysis and simulation of the resulting layouts.

Although there has been a recent interest in coupling HLS with physical design, thus far it has not considered the variation of aspect ratios of individual RT components, and its effects on the area and delay of these components, as well as on the final design. Intuitively, there should be substantial variations in these attributes due to additional wiring effects introduced by different physical design configurations resulting from aspect ratio variations. However, to the best of our

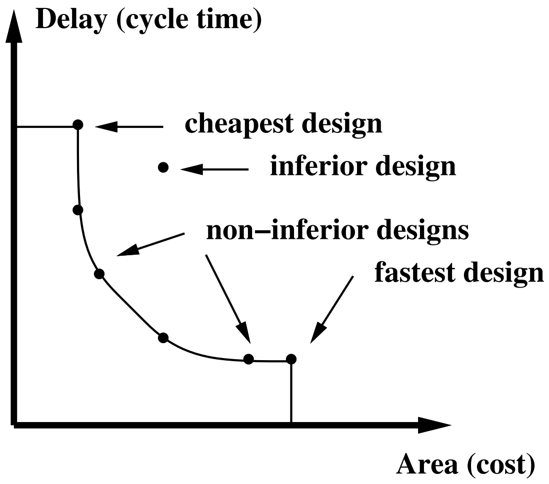


Figure 2: The “traditional” area-delay HLS paradigm.

knowledge, no previous work to date has examined and quantified the magnitude of these variations.

In this work, we attempt to realistically explore the effects of varying aspect ratios on the area and delay of generic components, as well as on the area and delay of complete designs generated by HLS using generic components. Section 2 describes previous and related work, while Section 3 defines the problem. In Section 4, we describe the experiments performed to observe the variation of a generic component's area and delay with respect to its aspect ratio. The variations were quite substantial and are indicative of the need to factor in aspect-ratio effects at the RT-component level. In Section 5 we describe the experiments performed to observe the effect of aspect ratio variations on complete designs generated by high-level synthesis tools – the results of these experiments show substantial variation in the area and delay of each final design, indicating the importance of factoring in aspect ratio variations during HLS. Section 6 concludes with some observations and a summary.

2 Previous work

Researchers are beginning to realize the importance of physical design (PD) considerations before, during, and after almost every phase of HLS (e.g., scheduling, allocation, module selection, binding, etc.).

One of the main principles of traditional high-level synthesis is the area-time tradeoff paradigm shown in Figure 2. The basic assumption here is that, within limits, one can always trade off cost (area) for performance (speed) and that faster chips always consume more layout area, and vice versa. In [1], Granacki and Parker compared the area time tradeoffs of various implementations of a given behavior both at the RT level and the layout level. It was concluded that while the AT tradeoff was still somewhat applicable, the abstraction of layout effects may mislead the designer into generating implementations which can be inferior in quality.

One of the earliest high-level synthesis systems to consider layout effects was BUD (Bottom-Up Design) by McFarland [2],[3]. In BUD, the flow graph operations are combined into a clustering tree. By controlling the clustering depth, various configurations can be achieved. Each can be evaluated based on a layout model. BUD was used as a design space exploration tool. When applied to some example designs, the results indicated that, when layout effects such as wiring and multiplexors were considered, the AT tradeoff "law" did not apply. In fact, it was observed that designs which had smaller layout area were indeed *faster* that those with larger area [3]. This behavior can be explained by the fact that designs with smaller layout area tend to have shorter wires. In large designs, wires tend to dominate in both area and delay. When the designs were reevaluated without incorporating the layout effect, the AT tradeoff curve was observed.

CHIPPE [4] employs an expert system paradigm to control the synthesis process. A candidate implementation is generated and evaluated. Based on the results of the evaluation and the designer constraints, some transformations are applied to the design by invoking some synthesis tools for scheduling and allocation [5]. The knowledge in this system is represented in the form of rules which indicate the actions to be taken based on the evaluation results.

A more recent work, Fasolt was reported in [6]. Here, a data path is first synthesized and evaluated using a Layout Estimator (LE). The resulting design is then analyzed and, based on the analysis, some transformations are applied in order to optimize the layout. The design is then reevaluated and the process of transformation followed by evaluation is iteratively repeated until no further improvements are possible. This approach was applied to three examples and improvements in area and performance were reported over the initial designs. The methodology described in this work is based on the concepts of incremental repair and partial designs previously reported in [7].

The work in [8] reported an attempt at simultaneous synthesis and floorplanning, referred to as 3-D scheduling (the three dimensions being x,y, and time). Here the starting point is a scheduled data flow graph. An assignment of operations to hardware is generated and the resulting design is floorplanned and the critical path in each cycle is analyzed. The system then tries to reduce the cycle time by exchanging the bindings between operations or adding extra hardware so as to reduce the length of the physical critical path. One of the interesting results reported was that an improvement in performance can be achieved by adding redundant operators, a move which may be somewhat counter intuitive when considering an abstract hardware model. This work only addresses operator binding which is a subtask of synthesis.

In [9] and [10], an abstracted layout area model for high-level synthesis was presented. This model was experimentally shown to accurately and efficiently reflect the effects of the data path design tradeoffs on the layout area and delay. It has also shown that traditional cost functions are not good indicators for optimization in high level synthesis.

A chip level model of layout was described in [11]. This model was successfully used to estimate layout area and delay of chips containing a mixture of macrocells, datapath, and random logic with 10% or better accuracy. This model was compared against traditional high level synthesis area and delay models and has shown the inadequacy of the latter models both in terms of their lack of both accuracy and fidelity when benchmarked against real layouts.

3 Problem Description

As noted earlier, HLS tools typically use abstract models of RT components for the synthesis tasks of scheduling, allocation and binding. These models range from extremely simple unit-delay, generic components, to more realistic models that attempt to factor in technology-effects. On the one hand, we would like to incorporate technology-specific information and PD effects early in the design cycle (e.g., during HLS). This requires detailed characterization of the component libraries and possibly customization of HLS tools to accommodate specific technology constraints. On the other hand, we would like to abstract out technology-specific information so as to simplify the HLS tasks and make the resulting designs targetable to different layout styles and cell libraries. In order to effectively factor in lower-level design information during HLS, we first need to understand the sensitivity of high-level synthesis techniques to physical design considerations such as wiring and module shape functions. In this report we specifically address the effect of varying RT-component aspect ratios on the area-delay attributes of RT-components, as well as its effect on a complete RTL design produced by a high-level synthesis tool.

We begin by defining the class of RT-component generators, their parameters and their attributes. HLS tools typically assume the availability of an RT design library that is characterized by parametrized generic components. The generator's parameters specify the size, functionality, design style and shape for a component instance. Let G be the set of component generators, P be the set of parameters and C_i be the set of components associated with a generator G_i:

- $G = \{G_i | G_i$ is a RT component generator.$\}$
- $P = \{P_i | P_i$ is a parameter.$\}$
- $C_i = \{C_{ij} | C_{ij}$ is a member of generator G_i.$\}$

We instantiate a particular component C_{ij} by specifying parameter values P_{ij} for the component generator G_i. For example, a 4-bit ripple-carry adder component with unity aspect ratio is generated by invoking the $ADDER$ generator with parameter values (bit-width = 4, style = *Ripple-carry*, AR = 1.0). Furthermore, with each such component C_{ij} we associate design attributes such as area (A_{ij}) and delay (D_{ij}).

The domain of parameter values P_{ij} associated with a generator G_i specifies the complete design space of component implementations for the generator G_i. Accordingly, by varying the parameter values for a generator, we obtain component designs with different area and delay attributes. Such a design space is typically visualized by the plotting the area and delay of the RT-component for varying bit-widths, resulting in a "family" of possible designs. Note that in the most general case, the parameter set P_{ij} for a component generator G_i includes not only the bit-width, functionality and design style, but also physical design information such as the component's aspect ratio. We define this general case to be the **comprehensive model** for an RT-component, where, for example, multiple design implementations (and hence area-delay attributes) can be generated for a fixed bit-width component by simply changing the component's aspect ratio.

The design space of a RT-component generator G_i can be constrained by holding constant the value of one or more of its parameters P_{ij}. For example, an adder component generated for a fixed shape (e.g., unity aspect ratio), restricts the space of all possible adders realizable from the ADDER generator. Traditionally, HLS tools have used one such restricted model that ignores the effect of aspect ratio variations in individual RT components as well on the complete design composed of several RT-components. In this report, we define a **point model** to refer to the restricted design space for a generator whose aspect ratio parameter is fixed to a constant value. For example, Figure 2 is representative of a point model for the design space for an n-bit adder, where each "point" defines an n-bit adder with a particular implementation style (i.e., ripple-carry, carry-lookahead, etc.). Similarly, for a complete design composed of several RT-components, we can use a "point model" that assumes a fixed aspect ratio, as well as a "comprehensive model" that reflects aspect ratio variations on the complete RT-level design.

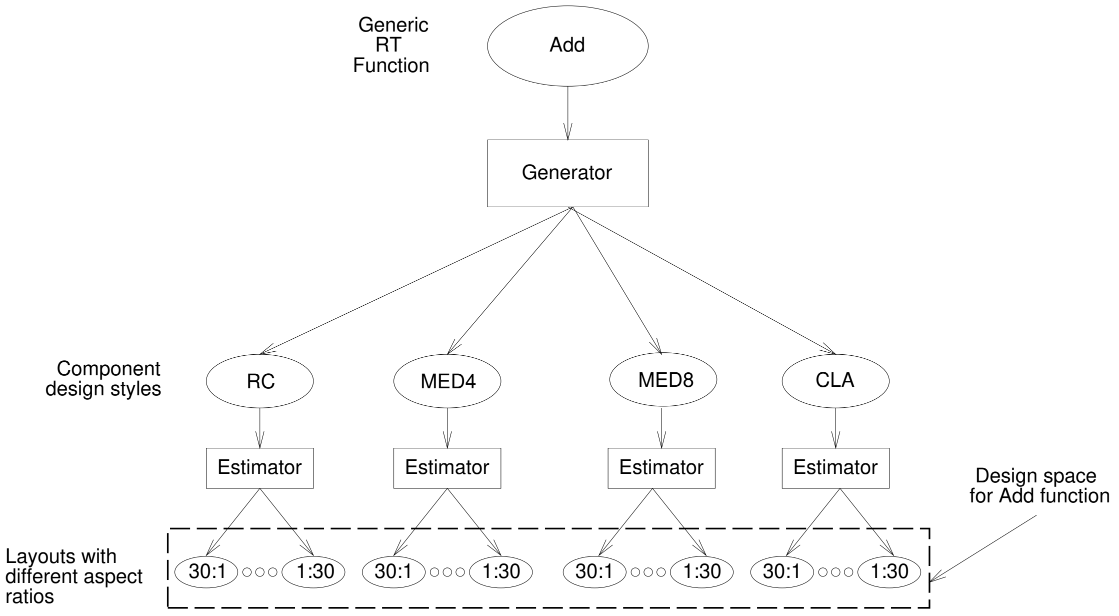
In this report we present the results of comparing the effect of the point model versus the comprehensive model on individual RT components, as well as on a complete RT-level design synthesized by HLS tools. In particular, we are interested in finding answers to the following questions. Given a particular RT-level component generator G_i,

1. What are the 1st order factors (parameters) that determine and/or influence the size, composition, and distribution of the "real" design space of a component generator G_i?

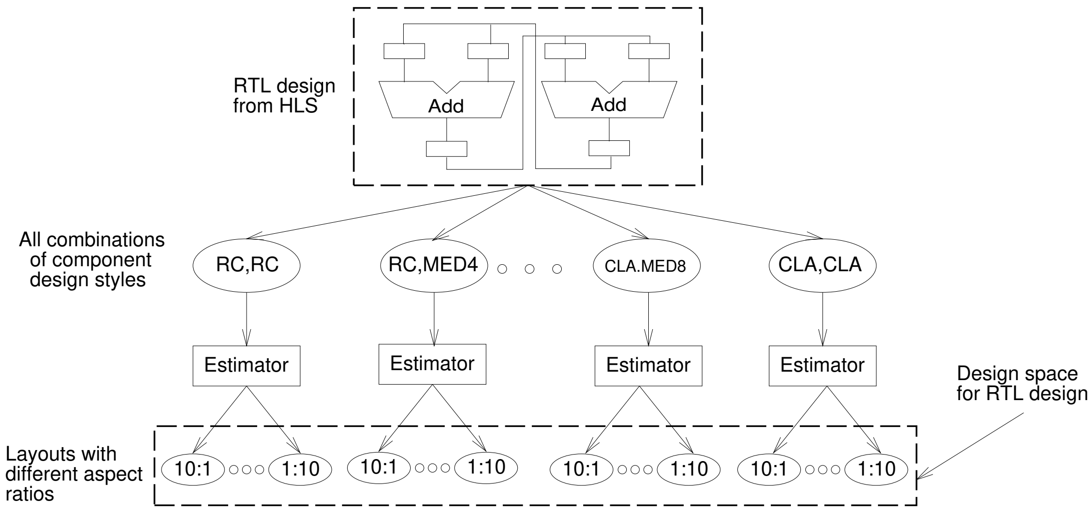
2. How would the nature of the design space of individual components (from 1) influence the overall design space of a "typical" complete design composed of several RT components?

3. The main question we wish to address is: are these "point models" of RT-components adequate for characterizing the resulting space of complete design implementations using these RT-components? In other words, how much penalty in accuracy is incurred by using the "point model" instead of the "comprehensive model"?

In order to answer these questions, we performed two sets of extensive experiments. In the first set of experiments, depicted in Figure 3(a), we varied the aspect ratios and styles for an individual RT-component and observed the resulting variations in the component's area and delay attributes. In the second set of experiments, depicted in Figure 3(b), we took the output of high-level synthesis applied to some benchmark descriptions, and studied the effects of varying component styles and



(a) Experiment 1 : Effect of aspect ratio variations on individual RT-component



(b) Experiment 2 : Effect of aspect ratio variations on complete RT-design

Figure 3: Description of the experiments of (a) Section 4, and (b) Section 5.

the design's overall aspect ratio on the area and delay of the complete RT-design. In both sets of experiments we observed large variations in the area and delay. We then generated several figures and tables to help analyze these variations.

Before describing these experiments, we first outline the experimental set-up used for our design runs. For the generator set G, we used the generic component library GENUS [12] that is used extensively by several high-level synthesis tools at UC Irvine. GENUS provides logic equations for every instantiated component from a class of generators. For example, consider the ADDER generator instantiated with parameters (num-bits=4, style=RC) and (num-bits=4, style=CLA). GENUS automatically provides the logic equations for both the ripple-carry and the carry-lookahead 4-bit adder. These logic equations are then synthesized by MIS and mapped to the GDT CMOS standard cell library.

Since the actual layout characterization process is extremely time-consuming, we used two fast but accurate layout estimators: LAST, a Layout Area and Shape function esTimator [13], and TELE, a Timing Evaluator based on Layout Estimates [14]. Both tools rely on a combined analytical/contructive layout model which estimates the layout area and delay for a range of layout aspect ratios. Both LAST and TELE have been benchmarked against actual layouts. LAST was benchmarked using standard cell layouts produced by commercial layout tools from VALID and Mentor Graphics and found to predict layout area with an average relative error of about 5% for standard cell design as large as 15,000 cells. Similarly, TELE was shown to achieve a 7% or smaller relative error in the worst case delay predictions for standard cell designs with up to 1800 cells. Since both LAST and TELE generate estimates within a few seconds of CPU time, we were able to accelerate the experimental run times without paying a penalty for accuracy.

4 Effect of Aspect Ratio Variation on RT-Components

This section explores the effect of varying aspect ratios for a RT-component on its area and delay attributes and attempts to determine if the "point model" for RT-components sufficiently covers the actual design space (i.e., the "comprehensive model") of component implementations.

The experiments, as illustrated in Figure 3(a), were organized as follows:

1. We chose the RT-level ADDER generator which is characterized by three parameters: (bit-width, style, aspect ratio). The adder is an interesting RT-component, since it has multiple style choices.

2. We generated several ADDER components by varying the styles of designs for each bit-width, and repeated this process for different bit-widths. In particular, we generated:

   - Ripple Carry adder (RC),

- MED4 adder, (i.e., 4-bit carry-lookahead blocks rippled)
- MED8 adder, (i.e., 8-bit carry-lookahead blocks rippled)
- Full carry-lookahead adder (CLA), and
- Carry save adder (CS).

3. For each implementation, we first estimated the area and delay of the corresponding design netlist assuming a unity aspect ratio (i.e. square layout). This essentially gave us a "point model" estimate. The estimation was done by first generating a standard cell netlist of each implementation, and then used the highly accurate predictors LAST and TELE (briefly described in Section 3) to obtain estimates of its layout area and delay.

4. We also used the "comprehensive model" to generate area/delay values for different aspect ratios of each RT-component. This capability is built into our model because it employs the constructive-analytical technique described in [13] [14] when estimating both area and delay.

5. To analyze the effects of aspect-ratio variation on each RT component, we plotted the results of the experiment in several ways and attempted to capture the overall effects with figures indicating percentage variations in area and delay for each RT component, with respect to different design styles (e.g., ripple-carry or carry-lookahead) and bit-widths.

The results of the experiments are summarized by the graphs in Appendix 6 which graphically shows the results for adders whose bitwidths range from 8 to 128 bits. In these graphs, the point models (corresponding to designs with unity aspect ratios) are represented by big circles. Specifically, these graphs show:

- Area vs. delay of the adder (with varying aspect ratios),
- Aspect ratio vs. area for the adder,
- Aspect ratio vs. delay for the adder, and
- Aspect ratio vs. the product of area and delay (AT – a relative overall figure-of-merit)[1].

Table 1 summarizes the percentage variations obtained by varying the aspect ratios of different adder implementations with different bit widths.

In analyzing the data, we note the following:

- we observe large variations in area and delay, as summarized by Table 1. In most cases, the variations exceed 30% in either area or delay by simply changing the component's aspect ratio.

[1]Assuming a linear tradeoff between area and delay for the sake of illustration – any other tradeoff model could also be used.

Style		*Bit_width*					
		8	*16*	*32*	*64*	*96*	*128*
Ripple carry	area	101.11	183.51	67.80	62.36	31.04	24.77
	delay	30.60	41.65	17.29	31.50	98.55	35.87
4-bit carry lookahead	area	108.02	131.15	87.82	52.59	29.01	20.11
	delay	23.37	37.53	31.57	62.60	61.42	77.14
8-bit carry lookahead	area	100.28	140.53	56.65	26.63	22.70	14.55
	delay	28.36	43.42	84.20	137.73	110.40	169.02
Full carry lookahead	area	100.28	70.29	69.70	18.45	19.11	23.24
	delay	28.36	51.92	50.61	98.77	99.86	125.22
Carry save	area	160.88	100.95	55.41	26.40	22.02	
	delay	36.92	45.71	70.73	69.89	99.89	

Table 1: Percentage variation in area and delay for ADDER design

- for smaller bit widths, the area variations are larger than the corresponding delay variations, but the trend is reversed for larger bit widths. This indicates that the areas of larger designs tend to change less at extreme aspect ratios while delay is still quite sensitive to the changes in the relative layout dimensions.

- The design space (area versus delay) is more "cluttered" for the smaller bitwidths (e.g., Figure 8(a)), and is sparser for larger bit widths (e.g., Figure 8(b)). Thus, for smaller bitwidths, there are no clear "boundaries" between the different design styles. For example, two implementations with different design styles (MED4 and MED8) can have similar areas (or delays) for some aspect ratio configurations, and hence there is no clear choice unless the component shapes are restricted by a floorplanner.

- the "bathtub" effect of extreme aspect ratios on area is less pronounced as bit width increases, i.e., the Aspect-vs-area curves are "flatter" for larger bit-widths. Note that this is not true for delay, i.e. delay is still sensitive to extreme aspect ratio variations for all the designs.

- using the A*T figure of merit, MED4 appears to beat the others consistently (i.e. it has the lowest AT product). This is also confirmed in the Area-vs-Delay curves where the MED4 design points generally appear closer than others to the origin.

- The Area-vs-Delay curves show that CS adders have the worst area and inferior delays in general. Moreover, using the A*T figure of merit, CS adders seem to be the worst buy. However, this conclusion may change if another figure of merit is used.

Based on these observations, one can easily see that the point models are not adequate for representing the RT-component attributes, and for use in High-Level Synthesis because:

1. RT components are building blocks that will be floorplanned together to form a design layout. Thus, the final aspect ratio (and hence the are-delay attributes) of each component cannot be known until physical design is started. However, point models implicitly assume a single aspect ratio configuration which yields a single area-delay value for each component – in other words, they assume that area and delay do not vary under different aspect ratio configurations.

2. Contrary to the assumption above, the experimental results in Table 1 clearly indicate that there are large area and delay variations with aspect ratio. Thus one cannot rely on a single set of attributes (area,delay) of one point per design style as a "representative" of the whole design space of that style.

3. Since point models are not adequate for modeling the metrics of RT component, they cannot be used as a basis for module set selection prior to, during, or after scheduling and allocation. Instead, one must examine the full design space of these components, or alternatively, include the physical design information into the synthesis process. The latter approach means that component style *and aspect ratio* must be determined at the same time in order to obtain reliable measures of the attributes of the components.

5 Effect of Aspect Ratio Variations on Complete RT-Designs Obtained from High-Level Synthesis

The previous section showed that we observe substantial variation in area/delay attributes for each component generator by simply varying the aspect ratio parameter of the implementation. However, a larger question is, how does this influence a complete RT design, and how can this be used in the process of generating a RT-design from High-Level Synthesis?

In an attempt to answer this larger question, we examined the combined effects of aspect ratio variation as well as component style variations on two popular examples from the HLS literature: MAHA [15] and Shift-multiplier [16], and one industrial timer example. The objective of these experiments was to study the overall RT area-delay design space as populated by different RT-component styles (e.g., CLA adder and RC adder), and by varying the overall aspect ratio of the final RT design, as shown in Figure 3(b). That is, our goal was to compare the effectiveness of the "point" model (using fixed aspect ratios) for covering the complete design space represented by the "comprehensive" model. The results are summarized using graphs that plot area vs. delay, aspect ratio vs. area, aspect ratio vs. delay, and aspect ratio vs AT, as well as tables that show percentage variations in area and delay attributes with respect to the "point" model for the complete RT designs.

The experiments were run in the following manner. For each RT design generated by a HLS tool ([17]) from a benchmark description,

1. We used the "point model" to generate the design space with a fixed aspect ratio using typical RT components, including layout and wiring which were estimated using LAST and TELE.
2. We then modified the aspect ratio of the final design and generated the RT design space for different combinations of RT component styles/implementations, and included the effects of wiring and PD. To make the comparisons realistic, we restricted the aspect ratio variations to between 1:10 through 10:1.

The results for the timer and shift multipler benchmarks are shown in Figures 4(a) and 4(b) respectively. The results for the MAHA benchmark using an allocation of one adder for 8 and 16 bits are shown in Figure 5. Figure 6 shows the Area-vs-Delay curve for the MAHA benchmark with an allocation of two adders. Note that the "point model" (with unity aspect ratio) is represnted by big circles in Figures timer+mult, 5 and 6. The percentage variation in area and delay with respect to the "point model" for the timer, shift-multiplier and Maha (one-adder and two-adder) designs are shown in Tables 2 and 3. By examining these figures and tables, we make the following observations:

- In all the cases, we observed a large size design space (as noted from the amount of variations in area and delay from Table 2). This design space tends to be densely-populated in some regions and sparse as area and delay increase. However, we note that the size of the densely populated region is large, indicating several not-so-obvious design alternatives for similar area/delay constraints. Furthermore, a seemingly inferior design point on the traditional Area-vs-Delay curve (i.e., a design with inferior area and inferior delay) may turn out to be the *only* feasible design due to the shape constraint in the final floorplan and layout.
- In comparing the point models to the overall design space, we observe that the space defined by the set of points in the "point model" is a small subset of the overall design space(Figures 5 and 6). In the case of MAHA with 2 adders (Figure 6), we can see that the range of delays of the "point model" space is only a small fraction of the possible variations in delay across the overall space.
- Given any target delay or area for the system within a certain range, it is possible to find *some* design point which is quite close to that target. This stems from the fact that the design space is quite dense for a wide range of area and delay values. More analysis is needed to be able to quantize this density in general.
- We observe more swing in delay than area (2-3 times or more), which indicates that in contrast to area, delay is quite sensitive to aspect ratio variations (this agrres with the results in [18]).
- The Aspect-vs-Delay graph indicates that designs using the MED* & CLA adders have close delays, while design using the RC adders are clearly slower.

- The designs with MED adders are generally the best for medium-to-high performance, while RC-based designs dominate for area-efficiency, however,
- The A*T figure of merit gives almost equal scores to all implementations, except RC which is higher in most cases. What this indicates is that it may not be worth the small savings in area to go with RC-based designs because of the drastic degradation in performance which would result from such a choice.

Based on these observations, the experimental results show conclusively that aspect ratio variations cannot be ignored during HLS, since variations are quite large and that the "point model" is indeed restricting the design space. We need to consider the "comprehensive model" that factors in aspect ratio variations in order to capture a more realistic design space.

6 Summary

In this report, we attempted to explore the effects of varying aspect ratios on the overall RT design space. Although it is intuitively obvious that aspect ratio variations of RT-components affect the area-delay attributes of individual components and complete RT-level designs, we provided conclusive data derived from a fairly extensive set of experiments to verify this fact. We presented experimental data for aspect ratio variations at the RT-component level, as well as aspect ratio variations for complete RT-level designs generated by HLS tools from some benchmarks. The variations in area and delay were observed to be significantly large to warrant early inclusion in HLS decisions.

Indeed, we believe that the results we obtained have some deep implications on the "traditional" flow of HLS design tasks where scheduling and allocation are subject to some area and/or delay constraints. Our results raise two interesting questions related to this approach:

1. Given these large variations in the final area/delay, how can one ensure that the area/delay constraints initially imposed on the overall design are indeed satisfied when design is laid out? This is especially relevant when the synthesized design is part of a chip which contains other blocks, thus the shape of the final design is not exactly known in advance.
2. Does it make sense to assume "point models" during HLS? Why constrain the HLS to assume a certain (area, delay) point for each component type? Why not have HLS optimize (or select) preferred (area,delay) for each component type since we can find *some* implementation which can be very close to those specs? This argument can be extended to more recent synthesis systems (e.g. [17] and [19]) which incorporate a mix-and-match strategy of component selection, except that here only the aspect ratio is the source of variation in results.

Style		*Timer*	*Shift-add Mult*	*Maha(1 Adder)*
Ripple	area	13.33	14.02	13.54
carry	delay	51.68	27.68	50.96
4-bit carry	area	23.51	13.43	10.32
lookahead	delay	59.36	35.79	47.79
8-bit carry	area	20.88	16.71	17.30
lookahead	delay	53.88	32.95	45.53
Full carry	area	17.28	11.70	17.49
lookahead	delay	51.97	33.12	46.57
Over all	area	27.86	26.58	22.08
designs	delay	92.37	64.47	70.14

Table 2: Percentage variation in area and delay by varying aspect ratios for some benchmark designs

Clearly, these and other interesting questions remain to be answered in light of the results we presented. Future work needs to address the issue of how exactly to abstract out this information for use early in the design (for example, during High-Level Synthesis).

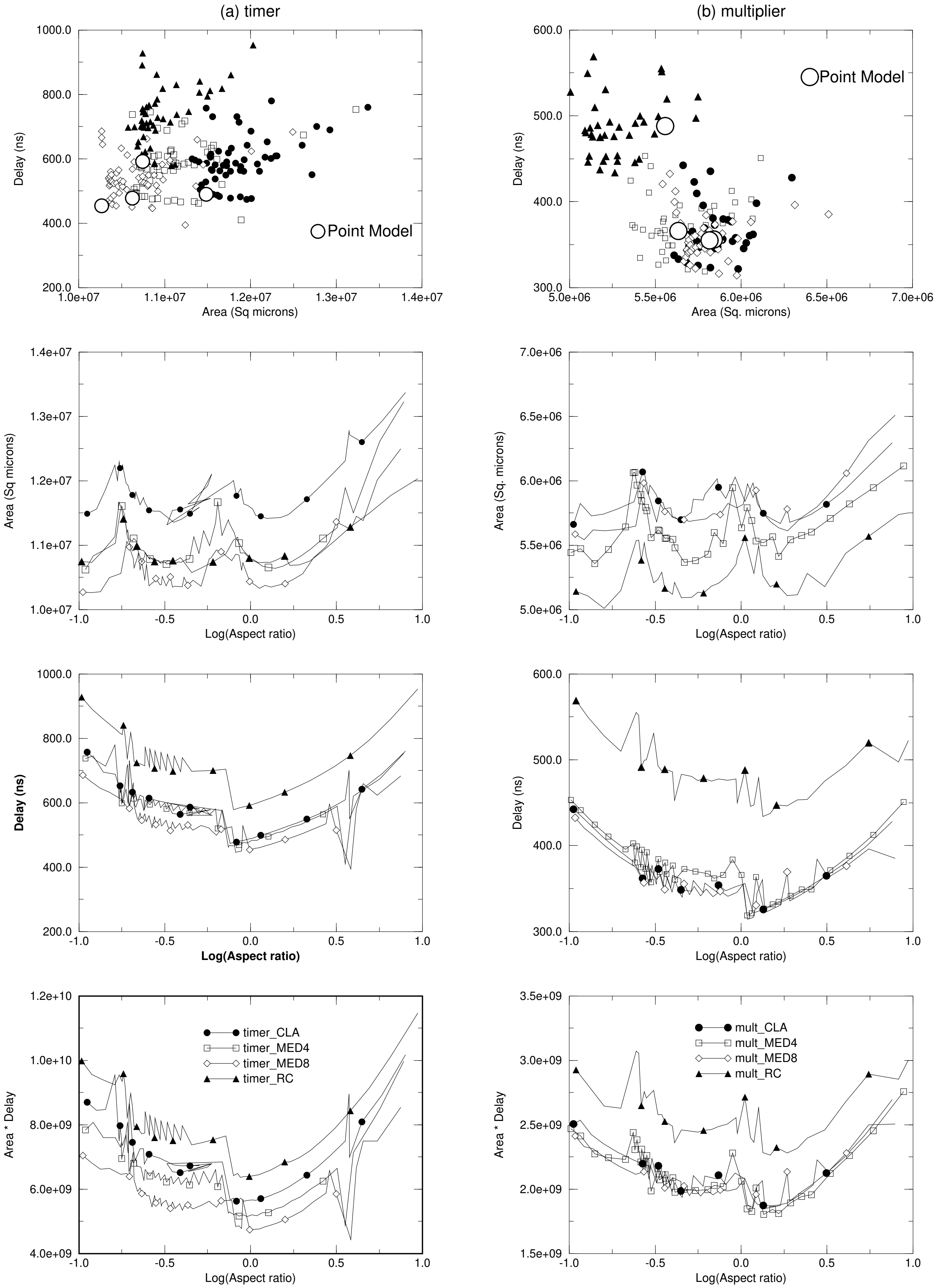


Figure 4: area and delay variations of (a) timer and (b) mutiplier (with varying aspect ratios). Both designs assume 16 bit words.

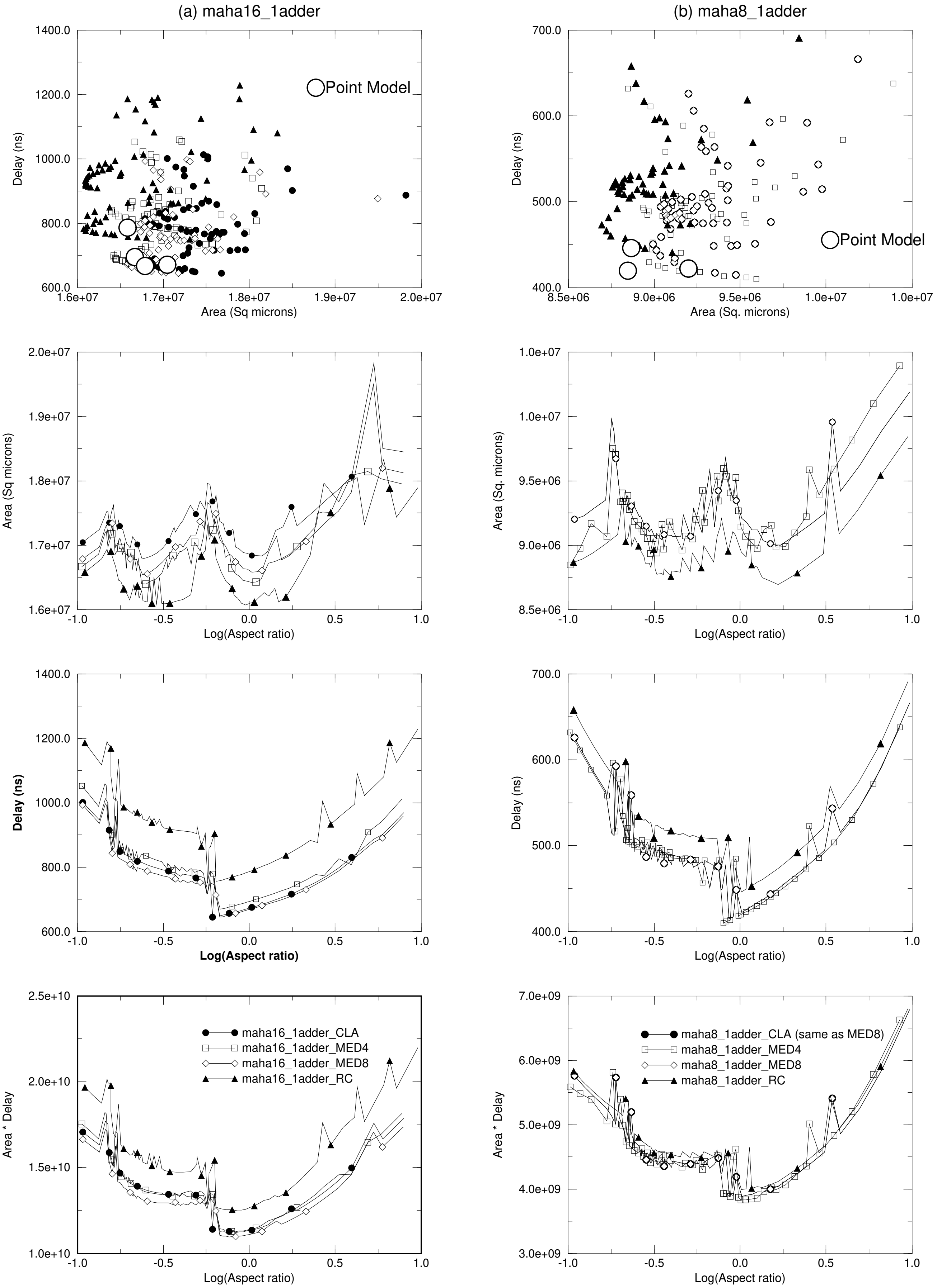


Figure 5: area and delay variations of MAHA design with one adder (with varying aspect ratios). Designs assume 8 and 16 bit words.

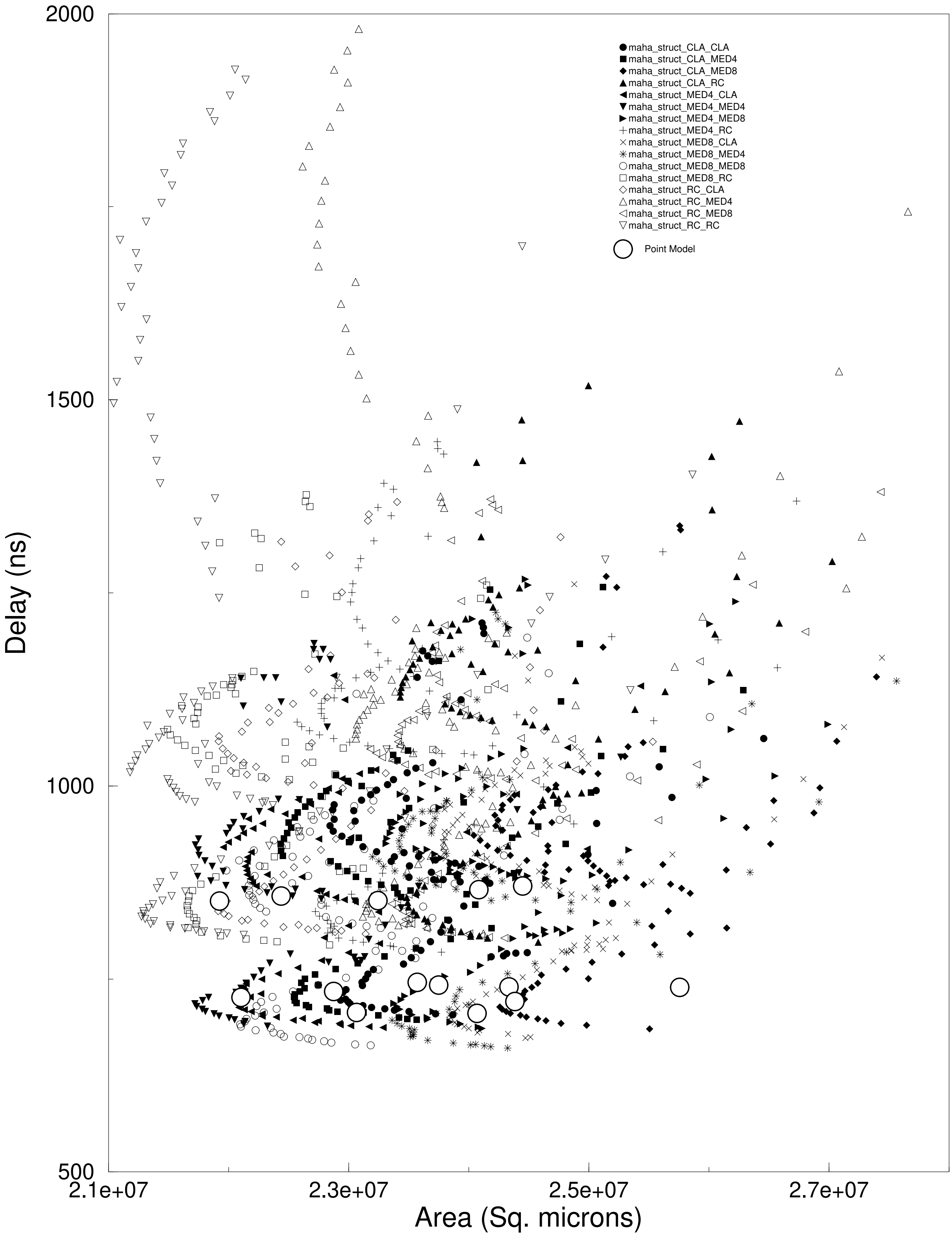


Figure 6: area and delay variations of MAHA design with two adders (with varying aspect ratios). Design assumes 16 bit words.

Style		*Percentage variation*	
Adder#1	*Adder#2*	*Area*	*Delay*
Ripple carry	*Ripple carry*	21.93	104.31
Ripple carry	*4-bit CLA*	21.23	107.24
Ripple carry	*8-bit CLA*	17.67	56.08
Ripple carry	*Full CLA*	15.86	54.88
4-bit CLA	*Ripple carry*	17.74	61.94
4-bit CLA	*4-bit CLA*	15.78	56.22
4-bit CLA	*8-bit CLA*	15.19	64.50
4-bit CLA	*Full CLA*	15.32	53.43
8-bit CLA	*Ripple carry*	12.29	56.44
8-bit CLA	*4-bit CLA*	18.12	64.88
8-bit CLA	*8-bit CLA*	17.60	63.14
8-bit CLA	*Full CLA*	17.38	66.67
Full CLA	*Ripple carry*	14.81	66.92
Full CLA	*4-bit CLA*	16.49	63.18
Full CLA	*8-bit CLA*	13.41	72.94
Full CLA	*Full CLA*	15.28	56.90
Over all designs		29.85	148.26

Table 3: Percentage variation in area and delay for 16 bit Maha with two adders

A Results of the Adder experiments

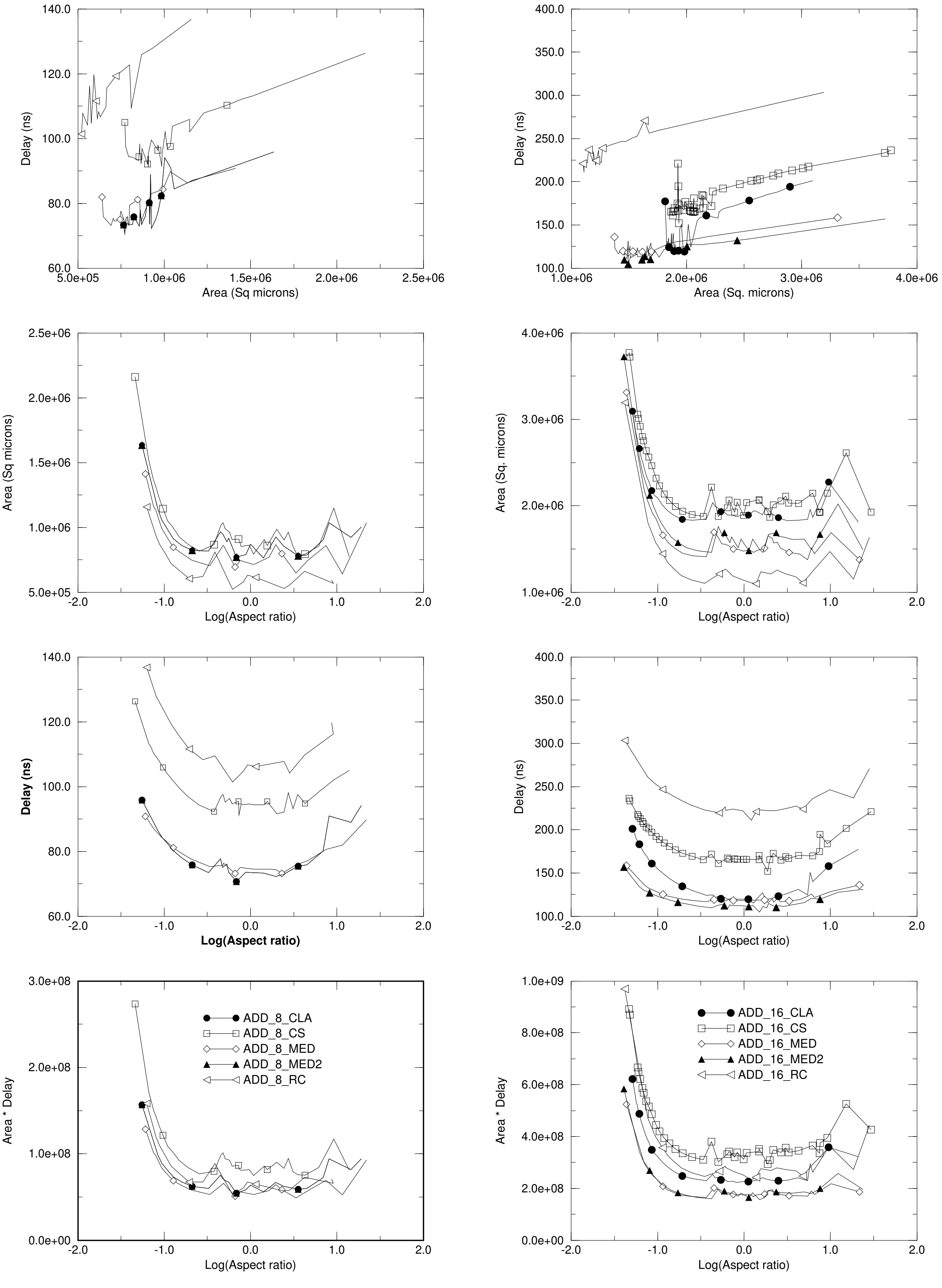


Figure 7: area and delay variations of (a) 8 and (b) 16 bits adders (with varying aspect ratios)

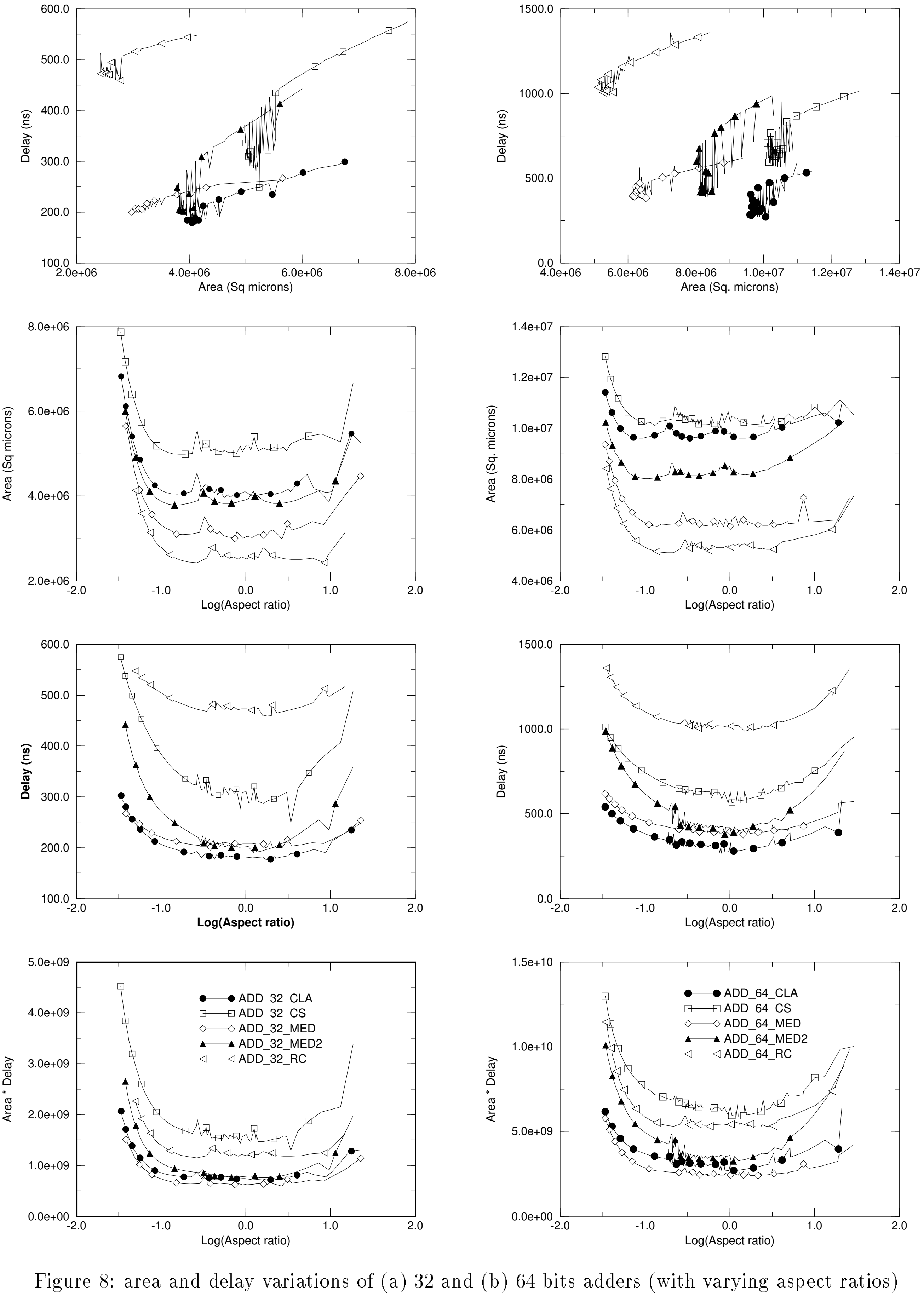


Figure 8: area and delay variations of (a) 32 and (b) 64 bits adders (with varying aspect ratios)

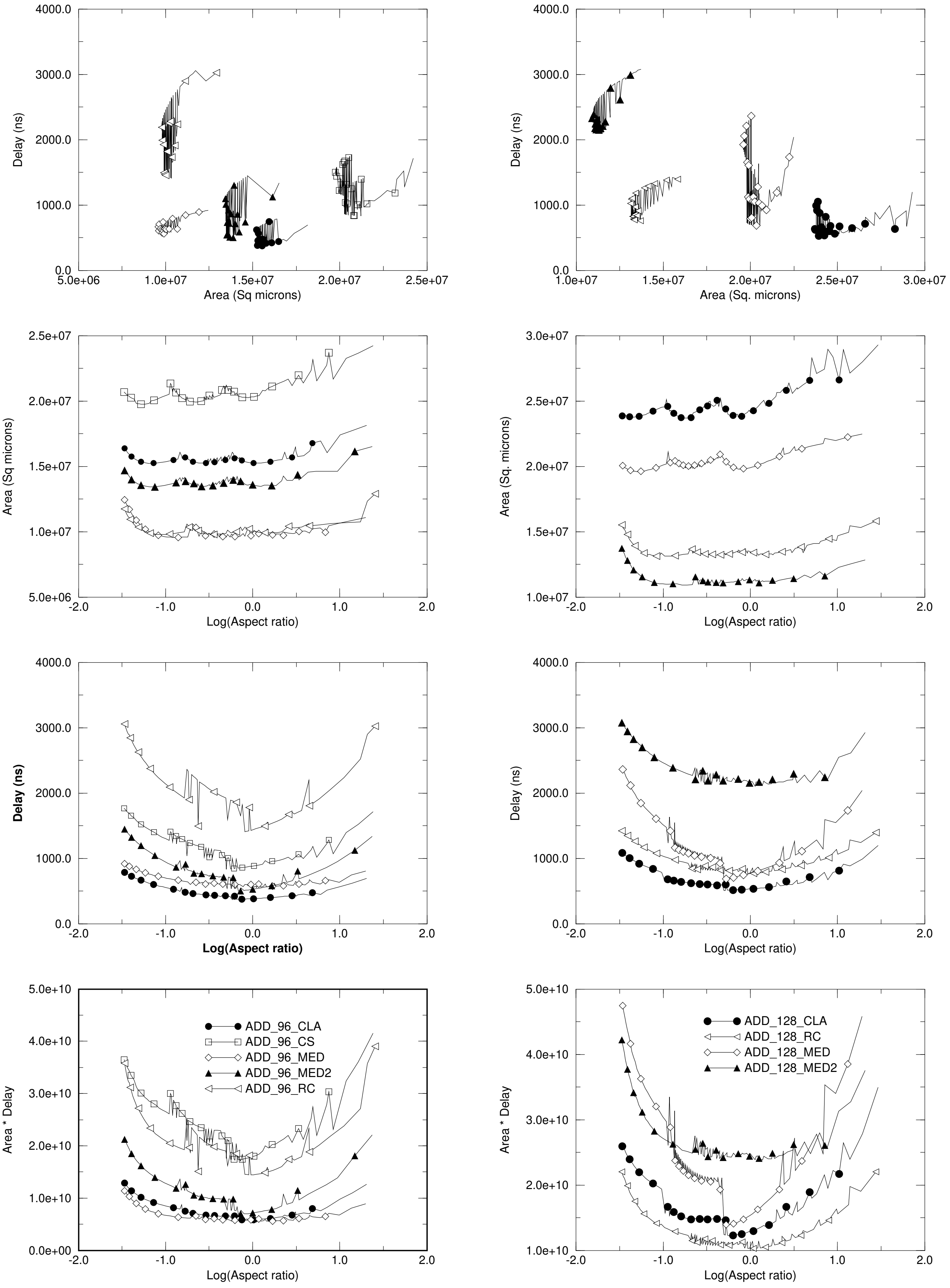


Figure 9: area and delay variations of (a) 96 and (b) 128 bits adders (with varying aspect ratios)

References

[1] J. Granacki and A. Parker, "The effect of register-transfer design tradeoffs on chip area and performance," in *Proceedings of the 20th Design Automation Conference*, 1982.

[2] M. McFarland, "Using bottom-up design techniques in the synthesis of digital hardware from abstract behavioral specifications," in *Proc. 23rd Design Automation Conf.*, pp. 474–480, IEEE/ACM, 1986.

[3] M. McFarland, "Reevaluating the design space for register-transfer hardware synthesis," in *Proc. ICCAD-87*, pp. 262–265, 1987.

[4] F. Brewer and D. Gajski, "Chippe: A system for constraint driven behavioral synthesis," *IEEE Trans. CAD*, vol. CAD-9, pp. 681–695, July 1990.

[5] B. M. Pangrle and D. D. Gajski, "Design tools for intelligent silicon compilation," *IEEE Trans. CAD*, vol. CAD-6, no. 6, pp. 1098–1112, 1987.

[6] D. W. Knapp, "Datapath optimization using feedback," in *Proc. EDAC-91*, pp. 129–134, Feb. 1991.

[7] D. W. Knapp, "An interactive tool for register level structure optimization," in *Proc. 26th Design Autiomation Conf.*, pp. 589–601, June 1989.

[8] J. Weng and A. C. Parker, "3-D scheduling: high-level synthesis with floorplanning," tech. rep., Dept. of EE-Systems, USC, 1991.

[9] A. C.-H. Wu, V. Chaiyakul, and D. D. Gajski, "Layout-area models for high-level synthesis," in *Proc. ICCAD-91*, pp. 34–37, Sept. 1991.

[10] V. Chaiyakul, A. Wu, and D. Gajski, "Timing models for high-level synthesis," in *Proc. EuroDAC-92*, 1992.

[11] C. Ramachandran, F. J. Kurdahi, D. Gajski, V. Chayakul, and A. Wu, "Accurate layout area and delay modeling for system level design," in *Proc. ICCAD-92*, Nov. 1992.

[12] N. D. Dutt and J. R. Kipps, "Bridging high-level synthesis to RTL technology libraries," in *Proc. 28th Design Automation Conference*, 1991.

[13] F. J. Kurdahi and C. Ramachandran, "LAST: A layout area and shape function estimator for high level applications," in *Proc. Second European Conf. on Design Automation*, Feb. 1991.

[14] C. Ramachandran and F. J. Kurdahi, "TELE: a timing evaluator using layout estimation for high level applications," in *Proc. EDAC-92*, 1992.

[15] A. Parker, J. Pizarro, and M. Mlinar, “Maha: A program for datapath synthesis,” in *Proceedings of the 23rd Design Automation Conference*, pp. 461–466, IEEE and ACM, 1986.

[16] F. Brewer and D. D. Gajski, “An expert system paradigm for design,” in *Proc. 23rd DAC*, 1986.

[17] L. Ramachandran and D. Gajski, “CHASSIS: A combined hardware selection and scheduling technique for peformance-driven synthesis,” Tech. Rep. 91-20, Dept. of ICS, Univ. of California, Irvine, Feb. 1991.

[18] C. Ramachandran and F. J. Kurdahi, “A combined topological and functionality based delay estimation using a layout-driven approach for high level applications,” in *Proc. Euro-DAC 92*, Sept. 1992.

[19] P. Gutberlet, J. Muller, H. Kramer, and W. Rosenstiel, “Automatic module allocation in high level synthesis,” in *Proc. EuroDAC-92*, 1992.