Optimized Power-Delay Curve Generation for Standard Cell ICs

Miodrag Vujkovic and Carl Sechen
Department of Electrical Engineering
University of Washington
Seattle, WA
miodrag@ee.washington.edu, sechen@ee.washington.edu

Abstract

An effective way to compare logic techniques, logic families, or cell libraries is by means of power (or area) versus delay plots, since the efficiency of achieving a particular delay is of crucial significance. In this paper we describe a method of producing an optimized power versus delay curve for a combinational circuit. We then describe a method for comparing the relative merits of a set of power versus delay curves for a circuit, each generated with a different cell library. Our results indicate that very few combinational functions need to be in a cell library, at most 11. The power-delay points achieved by Design Compiler from Synopsys using the state-of-the-art Artisan Sage-X library compare unfavorably to our approach. In terms of minimum energy-delay product, our approach is superior by 79% on average. Our approach yields the same delay points with a 107% savings in power consumption, on average. We also show that the specified V_{DD} for a process technology should only be used for the absolute fastest implementations of a circuit.

1. Introduction

The standard cell design style has been an important approach for application-specific integrated circuits (ASICs) for a long time. The quality of a synthesized design depends on three components: the synthesis tool, the place and route tools and the target cell library [1]. Choosing the right cell library can have a significant impact on the characteristics of a designed circuit.

General principles of cell library design, aimed to improve final circuit speed, have been proposed, such as providing each cell in a variety of drive strengths or including cells with dual polarity [1]. It was shown that relatively simple modifications in a cell library could lead to 20-30% speed improvements. Another study investigated the impact of library size on the quality of automated synthesis [2]. The results have indicated that an incrementally larger library size could considerably reduce area while meeting comparable timing requirements. On the other hand, the experimental study presented in [3] has confirmed that a great number of cells in typical libraries are not essential.

In high performance applications, the use of fixed, predefined cell libraries is becoming unattractive. Fixed libraries prevent device tuning for delay/power optimization [4]. Their physical features are rarely optimal for all applications, and therefore the performance of such designs is limited. The solution to these problems is the use of on-the-fly cell generation. These libraries are often referred to as *fluid* or *liquid* cell libraries.

A semi-custom design methodology, which exploits a fluid cell approach, has been reported in [5]. The basic building blocks used in this methodology are a set of parameterized static CMOS gates (around 10 different gates), suitable for circuit tuning. A cell generation tool was used to create a conventional library of discrete sizes. There were from 10 to 25 power levels and from 1 to 4 beta ratios for each gate type. The library consisted of approximately 1200 cells, and is referred to as a "tall thin" library.

A similar flow called the Power and Performance Optimization (PPO) flow was developed to achieve higher performance and reduce the power consumption in cell-based designs [6]. The PPO flow starts from an implemented design and optimizes the transistor sizes in each cell within a design to increase the performance and/or reduce the power dissipation. Also, in [7] a post-layout transistor sizing method for power reduction aims to reduce the redundancy of cell-based design and to obtain performance close to full-custom quality.

Transistor-level optimization can also be performed by restructuring. It can be combined with sizing of the individual transistors to optimize performance of a given structure. The proposed technique is called transistor-level resynthesis [8]. Results show that transistor-level resynthesis can achieve delay improvements up to 20%, along with a smaller transistor count and a power reduction. Efficient way to obtain the cost versus delay trade-off curve of combinational circuits by mapping of the gate-sizing problem onto piecewise linear model is presented in [9].

2. Motivation

The drawbacks associated with existing approaches to standard cell IC synthesis are as follows.

1) There has not been clear evidence as to which combinational cells should be in a library when optimized power versus delay trade-offs are the objective.
2) Typically only a single point in power-delay or area-delay space is produced when comparing cell libraries.

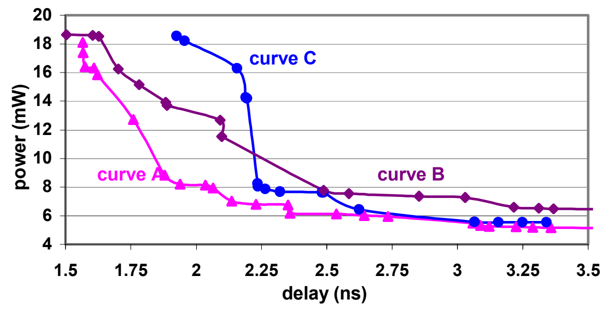

Figure 1. Comparison of power-delay curves

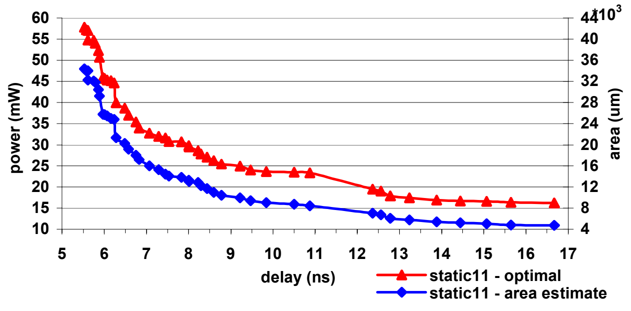

Figure 2. Power (area) vs. delay plot – each point represents one possible implementation

3) Fluid cell approaches have been used primarily to try to improve one aspect of the original power-delay point (typically to improve the delay).
4) Thus far different circuit synthesis approaches have been compared based on single points in power-delay or area-delay space.

An effective way to compare logic techniques, logic families, or cell libraries is by means of power (or area) versus delay plots, since the efficiency of achieving a particular delay is of crucial significance. In Fig. 1, it is apparent that whatever technique was used to produce curve A is clearly superior to the technique used to produce curve B. When two curves intersect, as do curves B and C in Fig. 1, it is possible to specify delay ranges (for example) where one technique outperforms the other.

Fig. 2 illustrates very typical power (and area) versus delay curves for a benchmark circuit. Note that a very wide range of power (factor of 4), area (factor of 9) and delay (factor of 3) are readily available.

In this work:

1. We identify a method to produce an optimized power versus delay curve for a combinational circuit.
2. We describe a method for comparing the relative merits of a set of power versus delay curves for a circuit, each generated with a different cell library.
3. We show which logic functions should be in a cell library to achieve the best power versus delay curves, analyzed over a set of benchmark circuits.
4. We show that the power-delay points achievable by Design Compiler from Synopsys using the state-of-the-art *Artisan Sage-X library* compare unfavorably to our approach.
5. We show that the specified V_{DD} for a process technology should only be used for the absolute fastest implementations of a circuit.

Our ultimate objective is to provide an optimized power (or area) versus delay plot for a circuit, in which the user can "click" on a particular point and the layout having those power (or area) and delay characteristics will be provided. Quite often the desired delays are either in fact not achievable, or are achievable but at a significant increment in cost (power or area). With power (and area) versus delay curves, feasibility and cost are readily available.

Another objective is to achieve full-custom circuit efficiencies (in terms of power, area and delay) while providing a fully automated design flow typical of semi-custom design.

The focus of this paper is on the steps leading up to the layout phase.

3. Design Flow

Our fluid cell design flow included the following steps: cell characterization, technology library generation, design synthesis, and transistor-level optimization. We used simulation parameters for the TSMC 0.18-micron technology, with lambda-based design rules, featuring a drawn channel length of 0.20 microns and a fanout-of-four inverter delay of 84ps. The initial set of transistor sizes was such that all nMOS widths were the same (1.4 µm), as were the pMOS widths (2.1 µm). The cells were characterized for input slews ranging from 50-400 ps, and fanouts ranging from one to ten (the unit load is *inverter1x*).

Design Compiler (DC) from Synopsys was used for synthesizing the benchmark circuits into optimized, technology-dependent, gate-level designs. The benchmark circuits were mapped to a given technology library – generated by Library Compiler. Four scripts with different logic-level optimization steps (flattening and timing-driven structuring) were used. We found that in all cases the best results were obtained using *Script1* (structuring enabled, flattening disabled). Consequently, only results obtained using *Script1* will be presented.

Based on prior layout experience, the wire load model we used was 17 µm per fanout. Since the wire capacitance for the target technology was 0.2 fF/µm, this implies 3.4 fF per fanout.

3.1 Transistor-Level Optimization

AMPS (Automatic Minimization of Power through Sizing) from Synopsys was used for transistor-level optimization within the three-dimensional optimization space delay/area/power [10]. AMPS takes a circuit netlist in *spice* format, in addition to a set of input patterns that will be used for power estimation, and runs a static delay analysis and a dynamic power simulation, based on the PowerMill/Nanosim and PathMill tools [11]. The power/delay optimization was performed using the three relevant optimization modes:

- Cost-function mode *(CFM)*
- Delay-requirement mode (*DM)*
- Power-requirement mode *(PM)*

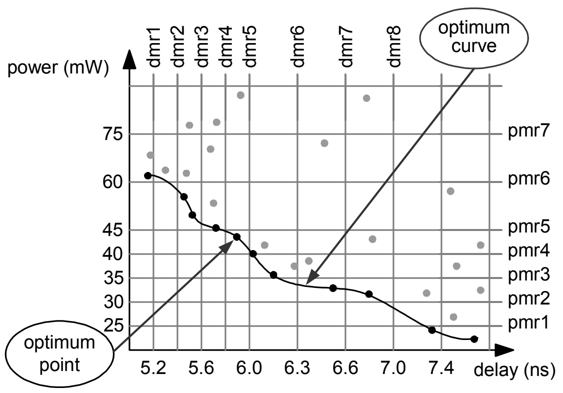

Figure 3. AMPS optimization process consists of target delay (dmr) and target power (pmr) optimization runs

First, AMPS was run for each benchmark in *CFM*. The weighting factors for the delay and power analysis were varied in range 1-32 by powers of two, for example, (delay cost, power cost) ∈ {(1,32), (1,16), ...(1,1), ...(32,1)}. The goal of this step was solely to obtain a range of delay/power values for each benchmark. After that, a group of target delay values was selected from the obtained delay range, and similarly, a set of target power values was chosen from the power range. The cost-function optimization rarely achieves the minimum power and delay values. Therefore, some target values were selected below the minimum of the range.

Our actual power vs. delay optimization is performed with AMPS in DM and PM, using the selected group of target delay and power values, respectively. In delay-requirement mode, AMPS resizes the circuit to achieve the specified delay and then continues to reduce the power while preserving the worst-case delay. In power-requirement mode, the specified power was achieved first followed by the attempt to reduce the delay while not exceeding the required power. The AMPS optimization process consists of a series of DM runs (dmr1, dmr2…) and PM runs (pmr1, pmr2…) for a wide range of target delay and power values, as shown in Fig. 3. Each AMPS run consists of 10 iterations. While it is possible to do more iterations, we found no advantage in doing so.

During the optimization process AMPS generates a large number of points in (power, delay) space. Among them, only the dominant points are retained to define the optimized power vs. delay curve. A point ($x1$, $y1$) dominates a point ($x2$, $y2$) if and only if $x2$ is greater than or equal to $x1$ and $y2$ is greater than or equal to $y1$. In other words, both points are retained if and only if each of the points has exactly one attribute (power or delay) that is superior to the other point's corresponding attribute.

PS was provided with a discrete set of allowable transistor sizes. The allowable transistor sizes for our 0.18-micron process were between 0.7 µm and 14 µm, in steps of 0.7 µm, and between 0.3 µm and 0.7 µm in steps of 0.1 µm.

The transistor-level optimization can be performed using two different sizing schemes: *tx* and *pn.* In the *tx* scheme, every transistor in a cell can be independently sized, while in the *pn* scheme, all *n*-devices for a cell are sized as a group, and all *p*-devices are sized as a separate group. We found that the *pn* scheme produced equally good power-delay curves in far less time, and produced far fewer cell instances.

Cell name	static7	static8a	static8b	static9a	static9b	static11	static25
inverter	X	X	X	X	X	X	X
nand2	X	X	X	X	X	X	X
nand3			X	X	X	X	X
nand4			X	X	X	X	X
nor2	X	X	X	X	X	X	X
nor3				X	X	X	X
nor4							X
xor2		X	X	X	X	X	X
xnor2							X
aoi21	X	X			X	X	X
aoi22	X	X	X	X	X	X	X
oai21	X	X				X	X
oai22	X	X	X	X		X	X
aoi31-aoi33 (3)							X
aoi211-aoi222 (3)							X
oai31-oai33 (3)							X
oai211-oai222 (3)							X

Table 1. Overview of standard cell libraries

3.2 Overview of Cell Libraries

In our research we have investigated a very large number of libraries, where a given library is characterized by the different combinational logic functions it provides. Our experience suggests that the *static25* library (Table 1), containing 25 different logic functions and including most of the functions that have up to 3 transistors in series, represents approximately one of the most complex (in terms of function count) libraries that is in use today.

While we have investigated a very large number of subsets of this 25-cell library, Table 1 contains the best performing six subsets (where a subset with fewer cells is considered better than a larger subset that produces essentially the same power-delay curves). Furthermore, we observed that libraries containing multi-level cells and/or pass-transistor-based cells, such as XOR and MUXs, consistently yielded worse results than the libraries in Table 1 (due to space limitations we are not able to show this data).

4. Library Evaluation

For a given circuit, we generate a power vs. delay curve for each of the cell libraries shown in Table 1. The complete set of power-delay curves for benchmark circuit *C6288* is shown in Fig. 4. For each library in Fig. 4, among all power-delay points generated, only the dominant points are retained to produce the shown curve.

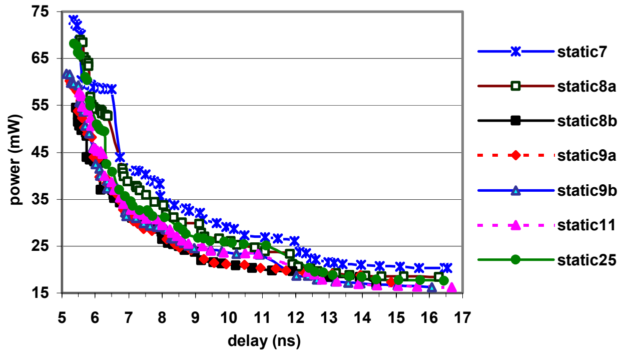

Figure 4. Power-delay curves for 7 static libraries (whole range) – *C6288* benchmark

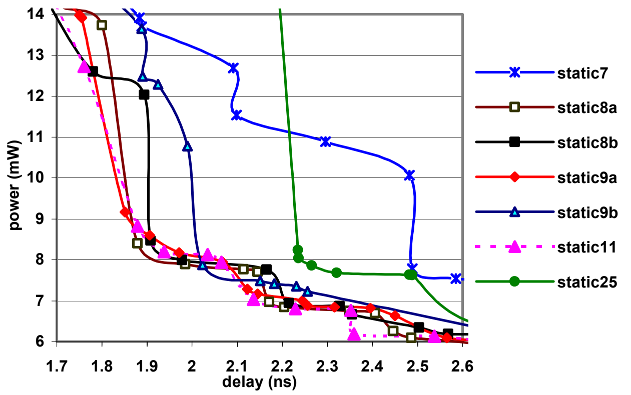

Figure 5. Power-delay curves for 7 static libraries (partial range) - *C7552* benchmark

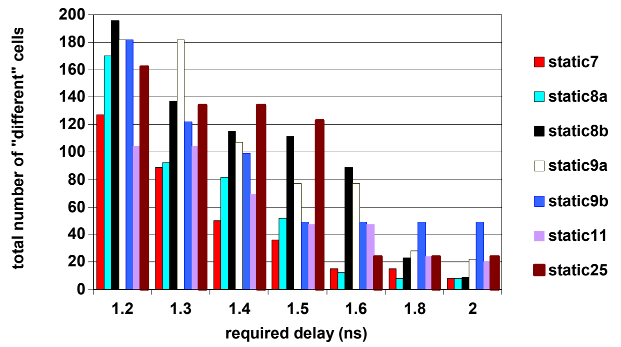

Figure 6. Cell count vs. target delay – for various libraries - *des* benchmark

A (zoomed in) portion of the power-delay curves for the*C7552* benchmark is shown in Fig. 5. Note that the largest library, *static25*, did not produce the best results for either benchmark circuit in Figs. 4 and 5.

A potentially unpleasant aspect of using a transistor and/or gate sizer on a random logic block is that it might touch a significant percentage of the cell instances in the block. We therefore measured the number of *different* library cells (unique instances) used in a design implementation. Cells are considered *different*, if they differ in size or functionality. Fig. 6 shows the different cell count versus target delay for various libraries for circuit *des*. The number of different cells generated during the optimization process generally increases with the number of functionally different cells in the starting library. As shown later, the *static11* library yields good power-delay performance, and yet as indicated in Fig. 6, uses a modest number of different cells.

4.1 Library comparison

Although a manual inspection of a family of power-delay curves can often lead to the identity of the best performing library for a given circuit, this is not always easy. Further, it is difficult to manually ascertain the relative behavior over a wide range of circuits. We therefore developed a method for determining the relative performance of the libraries.

After the family of power-delay curves is generated for a circuit, the next step is to identify the overall optimal power-delay curve. This is accomplished by retaining only the dominant points among all points produced by all libraries.

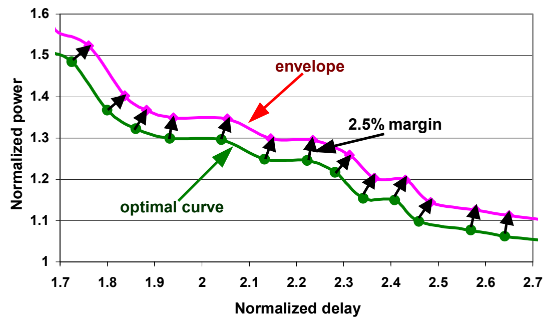

Figure 7. Envelope curve lies above the optimal curve by some margin

One possible measure of library quality is the number of points on the overall optimal curve due to that particular library. However, with this approach, non-dominant points do not contribute even if they lie in the vicinity of the optimal curve. This would be unfortunate since the accuracy of the tools used to obtain the points is at best within a few percent.

We therefore defined an *envelope* curve that lies above the optimal curve by a small margin and parallel to it (as shown in Fig. 7). In order to measure Euclidean distance on a plot with two types of units (power and delay), the power and delay values were normalized to their minimum values. The small margin we happen to use is 2.5%, but our conclusions were invariant for margins at least as high as 5%.

For a given circuit, a possible quality metric for a library would then be the total number of points from its power-delay curve that lie on or below the envelope. However, this may not be a good metric if groups of points tend to lie close to one another. In other words, five points that are grouped together in a particular portion of power-delay space would not be of the same quality as the case in which the five points are more separated and span a greater portion of the power-delay space. We believe that a better metric of library performance is the percentage of a library's power-delay curve that lies at or below the envelope. To accomplish this, a library's power-delay curve is decomposed into a piecewise linear representation. Note that each power-delay point is associated with two linear segments, as illustrated in Fig. 8. The curve length assigned to a power-delay point is defined as one-half of the length of each of the two linear segments (one to the left l_{i-1} and one to the right l_i) associated with the power-delay point.

$$\texttt{curve\_length(i)} = l_{i-1}/2 + l_i/2 \qquad (1)$$

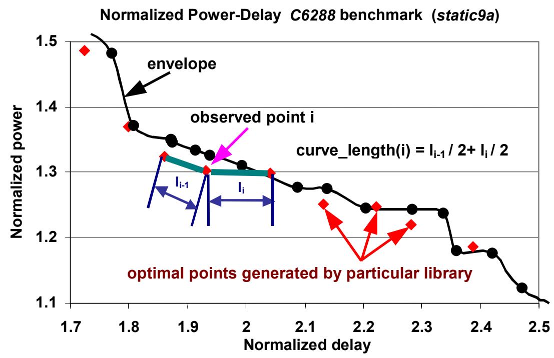

Figure 8. Curve length calculation

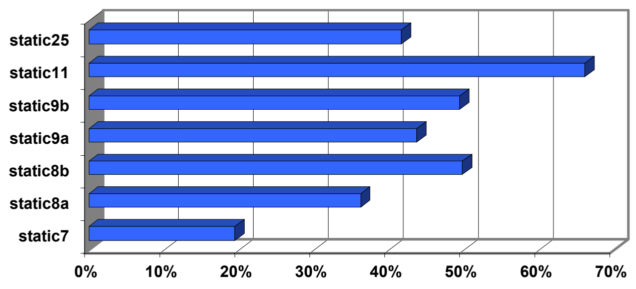

Figure 9. ***Quality*** **scores for 7 different libraries**

In order to include the influence of points that lie a very small distance outside the envelope, we developed a weighting function `weight(i)` for each power-delay point. A point *i* on or inside the envelope gets a weight of one, and the weight decreases quadratically with distance `d(i)` from the envelope. We used the following analytical formula for the weight function for a power-delay point *i*:

$$\text{weight(i)} = \begin{cases} 1 & \text{if } d(i) \le 0 \\ \left(\dfrac{1}{1 + \text{coef} \cdot d(i)}\right)^2 & \text{if } d(i) > 0 \end{cases} \quad (2)$$

Higher values of *coef* create a steeper curve for the weight function, which lowers the impact of points above the envelope. A reasonable value of *coef* is 15, and the subsequent results comparing the relative qualities of the different libraries are unchanged if *coef* is within a factor of 2 or 3 from our choice of 15.

The *Quality* score (or goodness) of a particular library *lib* can therefore be expressed as:

$$\text{Quality(lib)} = \frac{\sum_{\text{all points i generated by lib}} \text{weight(i)} \cdot \text{curve\_length(i)}}{\sum_{\text{all optimal points j generated over all libraries}} \text{curve\_length(j)}} \quad (3)$$

The *Quality* scores were computed for 7 libraries, averaged over 11 ISCAS combinational benchmark circuits, as shown in Fig. 9. We selected the largest available benchmarks, plus random selections among the remainder. Note that the highest *Quality* score was turned in by library *static11*, while *static8b* was ranked second.

Note that *static11* represents a subset of the *static25* library, and yet the average *Quality* score for *static25* is not as high as for *static11*. The presence of complex AOI/OAI cells (with 3 transistors in series) in the *static25* library affects the synthesis. Although the minimum achieved delay for both libraries is comparable, the *static25* implementation typically consumes more power and hence does not produce as good of a power-delay curve.

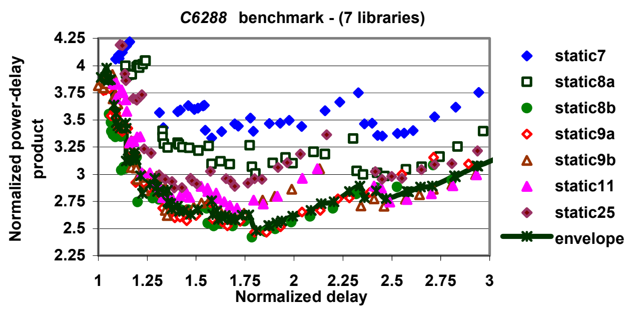

Figure 10. Normalized power-delay product curves

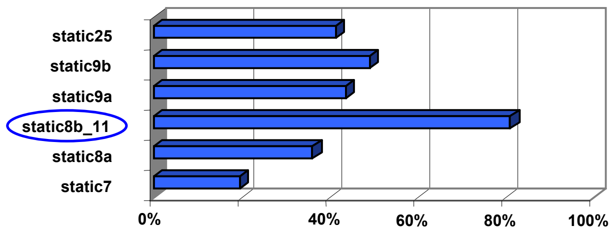

Figure 11. ***Quality*** **scores (*****static8b*** **and** ***static11*** **combined)**

We also plot the power-delay product (PDP) versus delay for benchmark circuit *C6288* in Fig. 10 for the various libraries. The PDP points were taken from the optimal points on the power vs. delay curve.

Since libraries *static11* and *static8b* were the two top performing libraries across all benchmarks, we therefore suggest merging the optimal points generated by the two libraries to obtain a new power-delay curve for the approach we call *static8b_11*. We perform separate synthesis and sizing runs for both libraries *static8b* and *static11*. Even though the *static8b* library is a subset of *static11*, its power-delay curve may be superior to *static11*'s curve for some benchmarks. In this way, the user is not limited to the set of power-delay points generated by a single library.

The *Quality* scores, given in Fig. 11, suggest that the *static8b_11* approach is almost twice as effective as any other library and in fact captures an average of 80% of the envelope.

To confirm that our *Quality* metric provides meaningful results, we performed two additional measurements averaged over 11 benchmarks. Fig. 12 shows the average power deviation from the overall optimal curve for 7 libraries and the *static8b_11* approach. (Note that the average power deviation is

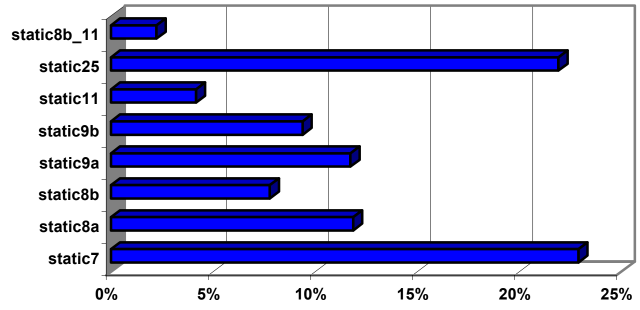

Figure 12. Average power deviation for 7 different libraries and static8b_11 approach

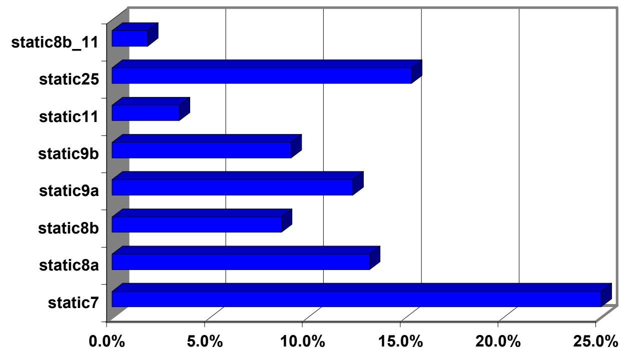

Figure 13. Minimum energy-delay product deviation for 7 different libraries and static8b_11 approach

in fact the average power increase for a library versus the overall optimal curve, and that this is essentially a measure of the area between the curves for the delay span of that particular library.) The drawback of this metric is that it doesn't capture the entire optimal power-delay curve like our *Quality* score. In other words, two libraries may have the same deviation, while one of them actually captures more of the optimal power-delay curve. For example, the *static25* library has a similar average power deviation compared to *static7*, while its *Quality* score is twice as good. This is because the delay-span of static25's power-delay curve is larger, and is in fact similar to the overall optimal curve.

Fig. 13 shows the minimum energy-delay product deviation from the overall minimum energy-delay product for 7 libraries and the *static8b_11* approach. Libraries *static11* and *static8b* remain the top two performers for both comparisons. The average power deviation was 4.17% and 7.79% respectively, whereas the *static8b_11* approach had a 2.24% deviation. Meanwhile, the minimum energy-delay product deviation was 3.44% and 8.66% respectively, whereas the *static8b_11* approach had 1.82% deviation.

5. Results versus Design Compiler

Thus far we have determined that *static8b_11* yields (on average) the best power-delay curve by a significant margin. Of interest now is how this power-delay curve compares to a power-delay curve obtained using Design Compiler (DC) from Synopsys and a commercial library.

We used the state-of-the-art Artisan *SAGE-X* standard cell library for the TSMC 0.18um, 1.8V process, [12]. The total number of cells (including sequential cells) in the Artisan library was 478. The library had 51 different base combinational functions and some complex arithmetic functions for adder and multiplier design. Most of the cells were provided in four different drive strengths. Inverters and buffers were available in two versions (symmetrical for clock signals and non-symmetrical) and 9 drive strengths. The total number of combinational cells in the library was 228.

Benchmark circuit	Minimum energy-delay prod. static8b_11 (mW *ns^2)	Minimum energy-delay prod. Artisan lib. (mW*ns^2)	Minimum energy-delay prod. improv.	Average pow. savings (Whole del. range)
des	22.954	53.259	132.02%	132.37%
C7552	29.765	51.950	74.54%	102.96%
C3540	29.411	39.321	33.69%	44.72%
C5315	23.256	34.537	48.51%	97.56%
k2	2.618	5.016	91.56%	68.75%
C6288	1280.850	1677.853	31.00%	33.93%
adder_16	0.467	0.971	107.82%	96.01%
mult16x16	44.826	96.772	115.88%	230.54%
comp_mult16	466.962	872.531	86.85%	149.32%
NR FIR filter	2095.513	1233.165	69.93%	25.46%
Average improvement:			79.18%	107.09%

Table 2. Comparison of minimum energy-delay products and average power savings for our optimization (*static8b_11* approach) vs. Design Compiler (Artisan *SAGE-X* library)

To obtain a reasonable set of points in (power, delay) design space using DC, two constraints were altered: maximum fanout and maximum delay. DC tries to achieve the target delay while minimizing the area. We performed about 25-30 synthesis runs using DC, whereas our optimization flow includes only two synthesis runs, each followed by 20-25 AMPS optimization runs for the *static8b_11* approach.

Fig. 14 shows representative comparisons for two of the larger benchmark circuits (*des* and *C7552*) of the power-delay curves generated by our optimization flow for the *static8b_11* approach and the power-delay curves generated using DC for the Artisan *SAGE-X* library. Table 2 summarizes results for six

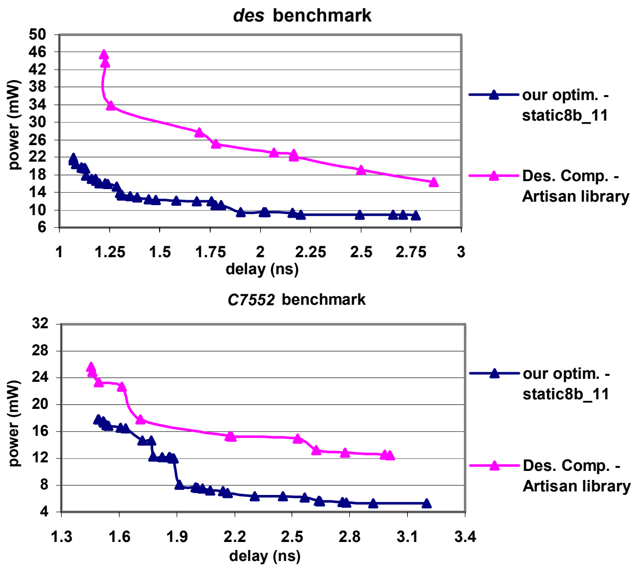

Figure 14. Our optimization (*static8b_11* approach) vs. Design Compiler (Artisan *SAGE-X* library) for *C7552* and *des* benchmarks

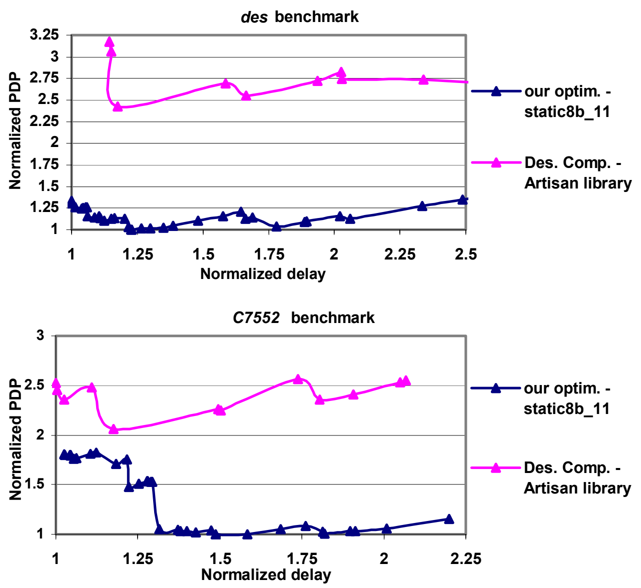

Figure 15. Normalized power-delay product vs. delay for our optimization vs. Design Compiler

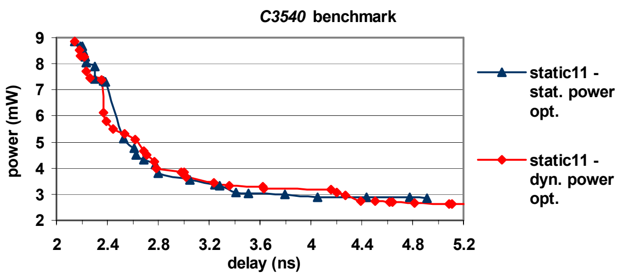

Figure 16. Power-delay curves for static and dynamic power optimization – *C3540* benchmark

larger benchmark circuits and four DSP macros (16-bit adder, 16x16 multiplier, 16-bit complex multiplier and a 32-tap FIR filter). The DSP macros were synthesized using the DesignWare library from Synopsys.

The average power savings for a circuit is computed for the intersection of the delay ranges produced by the two power-delay curves, where one of the curves is due to our optimization and the other is due to DC/Artisan. We also show the minimum energy-delay products for both curves and the average improvement achieved using our optimization flow.

The results for these benchmarks indicate that our optimization flow yields the same delay points with an average savings of 107% in power consumption. Furthermore, the energy-delay product is on average 79% better for our optimization. We have also noted, as shown in Fig. 15 for two circuits, that the average improvement in PDP is about 2X better for our optimization.

The advantage of transistor-level optimization (TLO) over gate-level optimization during synthesis is due to several factors. Perhaps the most important one is that the TLO tool has a global view of the netlist while doing the optimization, unlike the synthesis tool. Also, in TLO, it is possible to generate new cells with a much larger number of drive strengths and beta ratios. Finally, delay and power estimation at the gate level is less accurate than at the transistor level.

5.1 AMPS computation time

The AMPS computation effort consists of two components: static delay analysis and dynamic power simulation. The dynamic power simulation, performed using 400 test vectors, takes 20-30 times more computation time than the static delay analysis. The overall computation time for the larger benchmarks can be very significant (*e.g.* as high as 400 minutes for the largest *C6288* benchmark). One way to speed up the optimization is to execute the dynamic power simulation with a reduced set of test vectors.

Another approach that can significantly reduce the computation time while yielding comparable results is to perform a static power simulation during the optimization. This approach takes toggling activity information into account. Prior to optimization, the toggling count information was generated for each node based on the set of 400 test vectors. AMPS uses this information to estimate the power during the optimization. We ran AMPS for the same set of target delay and power values as before, and generated a number of points in (power, delay) space. Using static power simulation, the reported power values are only estimates (compared to dynamic power simulation). There-

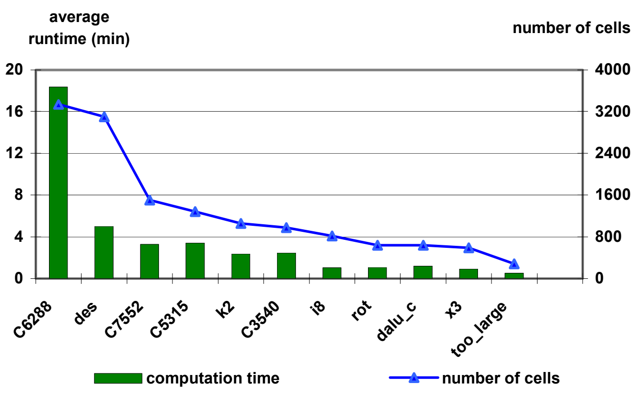

Figure 17. Average computation time vs. number of cells for one AMPS run using static power simulation

fore, after the extraction of the power-delay curve we performed dynamic power simulation (using NanoSim) only for points on the curve to obtain more accurate power values. The actual power values obtained from the dynamic power simulation can be higher or lower than the estimated values. Thus, some dominant points may become non-dominant, so they can be discarded.

Fig. 16 shows that the power-delay curves generated using dynamic and static power simulation are comparable. However, the computation time is drastically reduced by using static power simulation, in fact by a factor of about 20X. Fig. 17 shows the relationship between the average computation time (including both delay and power computations) for one power/delay run and the number of cells. The average runtime depends not only on the cell count, but also on the design structure. Benchmark circuits *des* and *C6288* have a similar number of the cells, but the *C6288* optimization takes almost 4X longer.

6. Supply Voltage Scaling

We now consider the use of supply voltages less than the specified 1.8 V for this process. The procedure begins with the

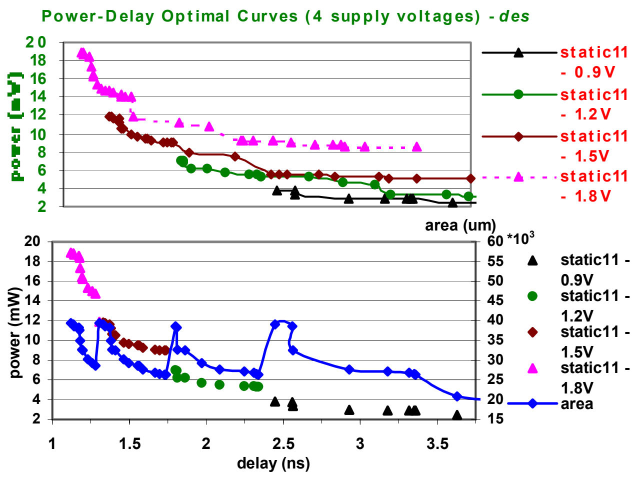

Figure 18. Single optimization method – (a) power-delay curves for different V_{DD} (b) power-delay and area-delay curves

optimized power-delay curve (*e.g.*, for approach *static8b_11*) generated for a circuit for 1.8 V as described previously. We then run PathMill (for delay) and NanoSim (for power) for each point on the original power-delay curve using a different supply voltage. We used 1.5 V, 1.2 V and 0.9 V. In this manner we produce 3 additional power-delay points for each point on the original power-delay curve, as shown in Fig. 18(a). The overall dominant points are then extracted, as shown in Fig. 18(b), where the area-delay curve is also shown. Each supply voltage proves to be optimal for a particular delay or power range. The highest supply voltage (1.8 V) covers the narrowest delay range. Conversely, the delay range for the lowest supply voltage is relatively large. We observed similar behavior for all of the benchmarks. Notice that the highest V_{DD} should be reserved only for the absolute fastest desired implementations.

Note the area rise at the boundary points of two different V_{DD}'s. It is evident that a higher V_{DD} point should be chosen if the target delay is at the V_{DD} boundary. An optimized curve with less area variation can be generated if smaller V_{DD} increments are used (*e.g.,* 0.1 V instead of 0.3 V).

7. Conclusion

An effective way to compare logic techniques, logic families, or cell libraries is by means of power (or area) versus delay plots, since the efficiency of achieving a particular delay is of crucial significance. An approach was developed to produce an optimized power versus delay curve for a combinational circuit. A method was developed for comparing the relative merits of a set of power versus delay curves for a circuit, each generated with a different cell library. We showed that the *static8b_11* curve, obtained by merging data points from the best two curves, *static11 and static8b*, performs almost twice as well as any other library curve. This suggests that very few combinational functions need to be in a cell library (8 to 11).

We also showed that the power-delay points achievable by Design Compiler from Synopsys using the state-of-the-art Artisan *SAGE-X* library compare unfavorably to our approach. We demonstrated that the specified V_{DD} for a process technology should only be used for the absolute fastest implementations of a circuit.

Our ultimate objective is to provide an optimized power (or area) versus delay plot for a circuit, in which the user can "click" on a particular point and the layout having those power (or area) and delay characteristics will be provided. Quite often the desired delays are either in fact not achievable, or are achievable but at a significant increment in cost (power or area). With power (and area) versus delay curves, feasibility and cost are easy to determine.

Another objective is to achieve full-custom circuit efficiencies (in terms of power, area and delay) while providing a fully automated design flow typical of semi-custom design.

Acknowledgements

We are grateful for the financial support provided by the National Science Foundation (NSF), the Semiconductor Research Corporation, MARCO, the NSF Center for the Design of Digital and Analog ICs (CDADIC), Boeing/DARPA, and Intel Corporation.

We also acknowledge Larry McMurchie for his useful suggestions.

References

[1] K. Scott, and K. Keutzer, "Improving Cell Libraries for Synthesis", *Proc. of Custom Integrated Circuit Conference (CICC)*, pp. 128-131, 1994.

[2] K. Keutzer, K. Kolwocz, and M. Lega, "Impact of Library size on the Quality of Automated Synthesis*", Proc. IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 120-123, 1987.

[3] J.L.Noullet, and A. Noullet, "Do We Need So Many Cells for Digital ASIC Synthesis?", *Int. Conf. on Mixed Design of Integrated Circuits and Systems (MIXDES)*, Lodz, Poland , 1998.

[4] K. Keutzer, and E. Girczyc, "Panel: Cell libraries - build vs. buy; static vs. dynamic", *Proc. of Design Automation Conference (DAC)*, pp. 341-342, 1999.

[5] G. Northrop, and P.F. Lu, "A Semi-Custom Design Flow in High-Performance Microprocessor Design", *Proc. of Design Automation Conference (DAC)*, pp. 426-431, 2001.

[6] E. Yoneno, and P. Hurat, "Power and Performance Optimization of Cell-Based Designs with Intelligent Transistor Sizing and Cell Creation", *IEEE/DATC Electronic Design Processes Workshop*, Monterey CA, April, 2001.

[7] M. Hashimoto, and H. Onodera, "Post-Layout Transistor Sizing for Power Reduction in Cell-Base Design", *IEICE Trans. Fundamentals*, Vol.E84-A, pp. 2769-2777, November 2001.

[8] S. Gavrilov, A. Glebov, S. Pullela, S.C. Moore, A. Dharchoudhury, R. Panda, G. Vijayan, and D.T. Blaauw, "Library-Less Synthesis for Static CMOS Combinational Logic Circuits", *Proc. IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 658-662, 1997.

[9] M.R.C.M. Berkelaar, P.H.W. Buurman, and J.A.G. Jess, "Computing the entire active area/power consumption versus delay tradeoff curve for gate sizing with a piecewise linear simulator*", IEEE Transactions on Computer-Aided Design of Integrated Circuits*, Vol. 15, pp. 1424-1434, November 1996.

[10] O.Coudert, R. Haddad, and K. Keutzer, "What is the state of the art in commercial EDA tools for low power?", *Proc. of Int. Symp. on Low Power Electronics and Design (ISLPED)*, Monterey CA, 1996.

[11] AMPS User Guide 5.4, Synopsys, 2000.

[12] TSMC 0.18um Process 1.8-Volt SAGE-X Standard Cell Library Databook, Artisan Components, Inc., 2001.