Standard Cell Library Optimization for Leakage Reduction

Saumil Shah
University of Michigan
Ann Arbor, MI

Puneet Gupta
Blaze DFM, Inc
Sunnyvale, CA

Andrew Kahng
Blaze DFM, Inc
Sunnyvale, CA

Abstract

Scaling device geometries have caused leakage-power consumption to be one of the major challenges of deep sub-micron design and a major source for parametric yield loss. We propose a library optimization approach involving generation of additional variants for each cell master, by biasing gate-lengths of devices. We employ *transistor-level* gate-length assignment to exploit asymmetries in standard cell circuit topology as well slack distribution of the design. The enhanced library is used by a power optimizer to reduce design leakage without violating any timing constraints. Such transistor-level optimization of cell libraries offers significantly better leakage-delay tradeoff than simple cell-level biasing (CLB) proposed previously. Experimental results on benchmarks show transistor-level biasing (TLB) can improve the CLB leakage optimization results by 8-17%. There is a corresponding improvement in design leakage distribution as well.

Categories and Subject Descriptors

B.7.2 [Hardware]: Integrated Circuits – *Design Aids*

General Terms: Design, Performance

Keywords: Gate-length biasing, Library optimization, Leakage reduction

1. Introduction

High power dissipation is a growing concern for high-performance circuit designers. Leakage power is soon becoming a dominant component of the total power and is projected to reach 54% of the total power at the 65 nm node [1].

There has been a large amount of work on leakage reduction [2-7]. Standby leakage reduction techniques focus mainly on reducing leakage of devices that are in an idle state, whereas runtime leakage reduction techniques focus on reducing leakage of active devices. Notable standby techniques are MTCMOS[2] and Substrate Biasing[3]. Among the previously proposed runtime techniques, a popular method is multiple threshold voltage assignment [4-6]. Multiple threshold voltages are used to generate variants of an existing standard cell. High V_{th} cells are placed on non-critical paths and Low-V_{th} cells on critical paths. Most of the prior work focuses on cell-level V_{th} assignment, where all devices in a cell have the same threshold voltage. The inherent asymmetries present in cells and circuit slack distributions motivate the need for fine-grained optimization beyond the realm of cell-level assignment.

Although reference [5] performs transistor-level V_{th} assignment, the authors use an exhaustive search method involving SPICE simulations, making the variant generation runtime unacceptably large. Also, the high cost of the extra masking steps required for different threshold implants limits the number of distinct V_{th}s to a maximum of 2 or 3, constraining the available design space.

In this work, we propose a new variant generation methodology to overcome these drawbacks. Reference [7] proposes the use of *small* biases to device gate-lengths for leakage reduction as well as leakage-variability reduction. We draw upon this idea to perform transistor-level gate-length biasing. Our method eliminates the high runtime, and limited design space constraints associated with the techniques proposed in [5] and [7].

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

The need for a transistor level assignment scheme is further motivated by the fact NMOS and PMOS devices have different Ioff/Ion dependencies on gate-length [7], and that different devices in a cell affect cell characteristics differently. [7] proposes biases of less than 10% of nominal length. However, process and area considerations constrain the bias even further and we are limited by bounds imposed by the technology[1].Taking these observations into consideration, we implement a cell-variant generation methodology in a module called Transistor-Level Biasing (TLB). Library optimization methods such as TLB are particularly attractive because the variant generation and characterization effort is amortized over multiple designs using the technology. Contributions of this work include:

1. Taxonomization of various cell-variant classes
2. Efficient algorithms to systematically augment a standard cell library to drive design power optimization.

The rest of the paper explains the variants as well as the generation algorithms. Section 2 motivates the need for, and describes the variant types. Section 3 describes transistor-level optimization for variant generation.. In Section 4, we describe the transistor-level delay and leakage models used in the biasing optimization. Section 5 describes the experimental setup and results and Section 6 concludes the paper.

2. Biasing Objectives and Cell-variants

Library optimization techniques are required to determine the best tradeoff between library size and design space. It is important to carefully determine the cells that would prove most useful in the circuit optimization process. The choice of variants is influenced by technology constraints, layout and design rule constraints, as well as by typical slack characteristics of designs.

To motivate the need for transistor-level biased variants, we consider the timing report statistics for a few benchmark designs shown in Table 1. The large discrepancy between rise and fall slacks is clear. In our sample set, this difference is found to be as large as 960 ps. A downstream power optimization engine will try to consume as much positive slack as possible to recover leakage power. We are able to identify several useful biasing objectives. The variants corresponding to the objectives are described in Table 2[2]. We distinguish between cell-level biased (CLB) variants, where all devices have equal bias and transistor-level biased (TLB) variants.

CLB Variants:

Maximum Leakage Reduction: Cells on paths with large positive slack can be replaced with variants (C_P*max*) which have all devices biased to the maximum positive limit.

Maximum Timing Improvement: Cells on paths with large negative slack can be replaced with cells (C_N*max*) which have all devices biased to maximum negative value.

We also (optionally) generate other CLB variants (C_P*n* and C_N*n*) where the biases are some fraction of the maximum bias, for paths with small positive or negative slack.

[1] The constraints are imposed primarily by the following design rules
   a. Polysilicon to polysilicon minimum spacing rules
   b. Polysilicon gate to active contact spacing rules

[2] Although the current work focuses primarily on leakage reduction (_P variants), negatively biased variants are included for the sake of completeness. Ongoing work on this tool involves the use of _N variants for timing optimization. All test circuits in Section 5 are initially timing-correct.

Table 1. Slack characteristics of circuit timing reports

Circuit	Avg. Slack (ps)	Max Slack (ps)	Avg. |R-F| (ps)	Max (R-F) (ps)	Max (F-R) (ps)
C5315	715	1570	17.7	50	80
C6288	127	470	29.6	50	90
S9324	501	1090	110	160	60
S13207	914	933	78.8	240	370
S38417	538	1540	68.7	300	300
AES	325	990	36.1	180	90
ALU	177	1370	21.1	40	100
Industry Case 1	1851	3240	69.5	960	320
Industry Case 2	667	2940	65.2	340	440

Table 2. List of Variants and polarity of biases

Variant	Objective	Bias assignment
C_P*max*	Maximum leakage reduction	All Positive Max
C_N*max*	Maximum delay reduction	All Negative Max
C_P*n*	Leakage Reduction: fraction of C_P*max*	Positive – Equal across devices
C_N*n*	Delay Reduction:fraction of C_N*max*	Negative – Equal across devices
A_P	Leakage reduction. Delay upper bound	Positive.
A_N	Delay reduction with bound	Negative
R_P	Leakage reduction. Only fall delay affected	Positive
F_P	Leakage reduction. Only rise delay affected.	Positive
R_N	Rise delay reduction	Negative
F_N	Fall delay reduction	Negative
D_P	Delay and Leakage Reduction. Emphasis on Leakage	Positive and Negative
D_N	Delay and Leakage Reduction. Emphasis on Delay	Positive and Negative

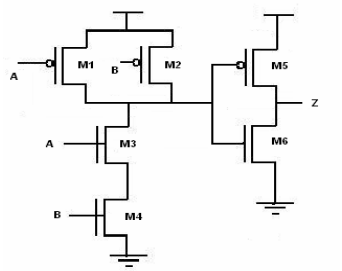

Figure 1. AND circuit diagram

Table 3. Distribution of states over different devices in AND gate

Input State		Device					
A	B	M1	M2	M3	M4	M5	M6
0	0	D	D	N	N	L	D
0	1	D	L	L	N	L	D
1	0	L	D	N	L	L	D
1	1	L	L	D	D	D	L

TLB Variants:

Leakage Reduction with Delay Upper Bound: Small positive slack can be exploited by TLB variants (A_P) that reduce leakage while maintaining the delay within a specified bound.

Delay Reduction with Bound: When there is small negative slack, it is useful to have variants (A_N) with delay reduction that is some fraction of the maximum possible reduction, to avoid excessively large leakage overhead.

Leakage Reduction with Transition-dependent Delay Overhead: These variants are for paths with large slack for fall transitions and little slack for rise transitions (R_P) or vice versa (F_P).

Transition-dependent Delay Reduction: These variants (R_N, F_N) are for paths with negative slack for a transition in one direction and zero or positive slack for the other.

Leakage and Delay Reduction: Finally, we propose special variants which we refer to as dominant (D) variants. Dominant variants are those that are superior in both delay and leakage to the nominal cell, or superior in one and equal in the other. These variants do not exist for all cells, and are possible only for technologies that allow both positive and negative biases. We motivate the existence of dominant variants by taking a simple example of an AND gate.

The circuit diagram for an AND gate is shown in Figure 1. Table 3 shows the state of each device in the circuit for different input states. The states are Delay Dominant (D), Leakage Dominant (L), Neither Delay nor Leakage Dominant (N).

A device is considered as delay dominant for a transition if it is in a charging/discharging path. It is considered as leakage dominant if it is turned off and series connected to other *off* devices. We draw the concept of dominant states from [4].

From the table, M3, M4 and M5 contribute to the same transitions while contributing differently to average leakage (M5 leaks for 3 states while M3 & M4 leak for only one). We can expect that intelligently increasing the bias of M5 and reducing that of M3 or M4 creates a variant with lower average leakage than the nominal cell, while maintaining similar delay characteristics.

As another example, we consider a cell where multiple input stages feed a single output stage. Assigning negative bias to devices in the output stage speeds up all transitions. The available slack can be used to reduce leakage by positively biasing devices in all input stages.

The examples suggest that dominant variants should be more common with multi-stage gates and this hypothesis is corroborated by experiment.

In the next section we describe methods of pruning the variant list under runtime/characterization constraints.

2.1 Variant List Pruning

Due to runtime constraints for SPICE characterization of variants, and for optimization runs, it is sometimes required to prune the cell-variant list. We investigate the biasing benefits of different cells based on their usage statistics and topologies.

Cell Usage Statistics:

Number of variants to be assigned to every cell can be based on observation of cell usage statistics over a few sample circuits. Maximum variants should be assigned only heavily used cells, whereas for sparsely used cells, the large characterization and optimization effort would not be justified.

Topology:

Heavily stacked devices usually have a small number of leakage dominating states, and their contribution to total cell leakage is small. It is not very useful to assign positive bias to these stacked devices, as the leakage gains are small.

This observation suggests that cells that have NAND topology (NMOS stacks) are not highly suited to R_P variants, as the biases are exclusively on the stacked devices. Similarly, cells with NOR topology are not suited to F_P variants.

For inverters and buffers, the pull-up and pull-down networks are exactly the same. If, for a particular technology, PMOS and NMOS delay/leakage tradeoffs are similar, A_P and A_N variants would not perform much better than C_Pn and C_Nn variants, and should be omitted. Also, for several cells, it is not possible to create useful dominant variants.

Having described all variants, in the next section we detail the generation of the variants.

3. Biasing Methodology

In this section, we describe our heuristic for generating the variants described in Table 2. We first introduce an important concept, the biasability of a device.

```
Algorithm: generateTLB
1. computeBiasability();
2. For all i, bias[i] = minBias;
   x=0;
3. Iterate
   x=x+1
   bias[i] = x*biasability[i];
   Snap bias to grid;
   computeDelayOverhead();
4. If Delay overhead > Delay Upperbound
   solution = previous bias;
   return solution;
   else
   goTo Iterate;
```

Figure 2. Basic Biasing Algorithm

Table 4. Average Delay and Leakage overheads for all variants

Variant	Rise Delay Overhead (%)	Fall Delay Overhead (%)	Leakage Overhead (%)
A_P	7.42	4.48	-31.92
A_N	-4.76	-4.57	34.81
R_P	-1.32	10.84	-17.67
F_P	9.11	-0.98	-23.60
R_N	-6.20	-1.80	60.97
F_N	-1.06	-6.24	89.88
D	-0.62	-0.99	-3.23
C_P4	5.77	4.66	-26.73
C_N4	-5.39	-3.97	36.79
C_P6	8.26	6.71	-36.27
C_N6	-7.99	-5.98	54.76

3.1 Biasability computation

The biasability of a device is a figure of merit for assigning a bias to a device. The basic definition of biasability for the leakage reduction objective is

$$B = \Delta Leak / \Delta Del \quad (1)$$

This definition is modified slightly to account for the different objectives described in Section 2.

C_P and C_N variants do not require biasability computation as all devices are unconditionally pushed to the same bias. The definition of biasability in Equation (1) is used for generating variants of the A_P type and also for the A_N type.

For the generation of R_P and F_P type variants, we use a modification of the biasability equation. The example given is for R variants; the F variants are analogous. Here,

$$B = \Delta Leak /(\Delta Del_{rise} + k) \quad (2)$$

Here ΔDel_{rise} is the average delay overhead for all rise transitions. The constant *k* is chosen such that the biasabilities of devices that contribute to a charging/discharging path for rise transitions are nearly zero. Other transistors also affect the rise transition by appearing as a load in a charging/discharging path. These transistors have intermediate biasability values This ensures that all the devices that significantly affect the rise transitions are not biased, the remaining devices get small or zero bias. R_N and F_N variants use this definition too.

For 'dominant' variants, we use the maximum delay overhead number in the denominator, as opposed to the average overhead.

$$B = \Delta Leak / \Delta Del_{\max} \quad (3)$$

This is motivated by the fact that a truly 'dominant' variant is meant to *replace* the nominal cell, and therefore should be superior in leakage and *per-arc* timing to the unbiased cell.

An optional practical consideration in this step is the handling of fingered devices. Typically, each finger is treated as an individual device by TLB. However, physical verification tools merge fingers into a single device for runtime considerations. This functionality is hampered by assigning different gate-lengths to each finger. If required, TLB can force the biasabilities (and therefore biases) of all fingers to be equal.

3.2 Biasing algorithm

In this section we describe the variant generation algorithm. The core algorithm is as described in Figure 2. We note that in step 3, we snap the biases to a pre-defined grid based on technology parameters at every iteration. The changes to the algorithm for different variants are outlined below.

- *A_P*: Identical to Figure 2.
- *A_N*: *minBias* is set to maximum allowed negative bias. Exit condition is failing to meet the required delay improvement.
- *R_P (F_P):* Exit condition changed to whenever it is found all the primary fall (rise) transition affecting devices have reached their maximum bias values.
- *R_N (F_N)*: Similar to *A_N* with exit condition being all devices that do not affect rise (fall) transitions (either directly or through loading) are 'unbiased'.
- *D*: Similar to *A_N* with exit condition as finding a biasing solution that has lower leakage than the nominal cell. A *D* variant is found if the variant delay is less than nominal.

Table 4 shows the average delay/leakage tradeoffs for the variants observed after characterization. C_P4 and C_P6 indicate cells where all devices are biased at 4nm and 6 nm respectively. The A variants[3] clearly show the tradeoff improvements achieved by using TLB over CLB. We compare the A_P variants with the C_P variants using the $\Delta Leak / \Delta Del(avg)$ metric. The value of this metric for A_P variant is 5.36, while for C_P4 it is 5.13 and C_P6 it is 4.84.

Similarly we compare the A_N metric with C_N4 and C_N6 through the $\Delta Del(avg) / \Delta Leak$ metric. The value is 0.134 for A_N, while it is 0.127 for both C_N4 and C_N6. Clearly, the TLB variants have a more favorable bias assignment compared to CLB variants, for both slack utilization and timing optimization.

4. Delay and Leakage Models

For the algorithms in the previous section, we need to compute delay and leakage overheads at every biasing step. We implement a fast and accurate transistor level modeling algorithm (TLM) similar to [8].

Delay Modeling:

At the core of our delay modeling routine is an RC Delay model. The delay is recomputed for every target input-state. Currently the model does not distinguish between different input transitions leading to the same output state.

A set of channel connected devices is referred to as a stage. The modeling routine is explained with the help of the two-stage AND gate described in Section 2. Using Table 3, we determine the delay-dominant devices corresponding to each input state. Performing series-parallel reduction on the dominant devices, each stage is reduced to an RC pair. Both gate and junction capacitances are considered. As an example, for transitions leading to input-state '11', the delay is expressed as

$$D = (R3+R4)(C5+C6+CJ1) + R5\times(CL+CJ2) \quad (4)$$

Here Ri and Ci are, respectively, the resistance and capacitance of device i and CJi is the effective junction capacitance of stage i. TLM obtains these values from look-up tables generated by SPICE pre-characterization.

Leakage Modeling:

To estimate the leakage of the cell, we again refer to Table 3. The leakage dominant states are determined as in Section 2. The total cell leakage for a state is the sum of the off-currents of dominant devices, again obtained from a lookup table.

Model accuracy is shown in Table 5. We note that the absolute delay and leakage values are not of interest here and only the relative overheads due to biasing are required to be accurate. The accuracy suffers due to layout dependent effects such as well

[3] Here the A variants are generated such that the delay change for these variants is 75% of the maximally biased C variants.

proximity, stress, etc. as well as the "lumped" nature of the delay model. However, the characterization results in Table 4 and optimization results discussed in Section 5 show that this level of accuracy is sufficient for optimization purposes.

5. Optimization setup and Results

Some or all of the variants described above are added to the existing standard cell library to generate a new library to be used within an optimization flow. The optimizer selectively replaces existing cell masters in a circuit with new variants to generate a leakage-optimized design.

For our tests, we use an industrial 90nm technology with BSIM 4.3 SPICE models. The optimization is carried out by a sensitivity based optimizer similar to [7]. For correctly using TLB variants a slack-aware sensitivity function is essential. This enables the optimizer to choose the appropriate variant for the particular value of available slack. This is especially important for R/F variants, where a slack-unaware sensitivity function may incorrectly prefer a variant with lower average delay overhead ignoring any transition dependent slack discrepancy.

Results:

We test our implementation on designs from the ISCAS-89 suite[9] and the Opencores[10] suite. Optimization results are shown in Table 6. The tests were carried out on three libraries:

1. CLB-only library containing only CLB variants.
2. CLB+TLB library with some of the CLB variants replaced with TLB variants while maintaining the same library size
3. Complete library with all available variants.

In the first two cases, the number of variants and hence the library size was maintained the same. Results show that new libraries achieve significant leakage reduction over the existing design. Since the library size is the same, the runtimes are comparable. Library 3 has a larger number of variants, improving the leakage reduction slightly, while increasing total runtime. The results show that we achieve, on an average, 36 % leakage improvement over unoptimized designs and 12% leakage improvement over CLB optimized designs. [7] also shows that increasing gate-length reduces leakage uncertainty caused by gate-length variation. Therefore, apart from reducing the mean of the leakage, we also expect to reduce its standard deviation. The plots in Figure 3 show the distribution of gate-leakage obtained by Monte-Carlo simulation for unoptimized, CLB optimized and TLB optimized designs for the AES benchmark. The distribution for the TLB optimized circuit is not only considerably shifted to the left, it is also much tighter compared to the other designs. Here, the standard deviation is 66% less than the unoptimized design and 39% less than the CLB optimized design. The increasing power-limited yield loss in scaled technologies makes this reduced sensitivity to line-width variation highly desirable.

Table 5. TLM Matching Accuracy

Cell	Delay Overhead (%)		Leakage Overhead (%)	
	SPICE	TLM	SPICE	TLB
INV	-4.76	-8.1	42.68	49.37
NAND	-7.3	-11.2	53.93	60.69
AND	-6.8	-7.75	50.02	56.56
AOI	-6.3	-6.5	51.82	57.49
MUX	-5.8	-4.7	50.94	56.77

Table 6. Optimization results for TLB & CLB based libraries

Circuit	Instance Count	% Imp CLB	% Imp TLB	% Imp All Variants
C5315	1681	27.66	41.69	42.09
C6288	3041	16.99	26.17	27.25
AES	30991	22.68	38.05	38.66
ALU	15880	15.68	32.56	33.13
S9234	1212	24.38	31.41	32.46
S13207	3464	30.83	40.15	40.43
S38417	11620	25.98	38.44	38.78

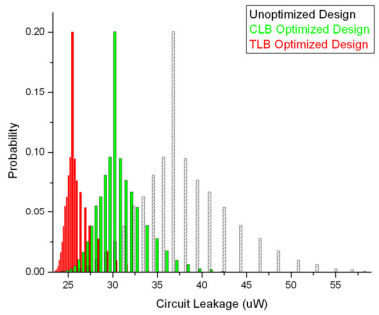

Figure 3. Pre and Post Optimization Leakage Distribution for AES

6. Conclusions

In this paper, we have proposed a new standard-cell library optimization method for leakage reduction. Existing standard cells are modified by performing transistor-level gate-length biasing, to change their leakage-delay characteristics. The enhanced library thus generated is used by a power optimizer to generate a leakage-optimized circuit from a given design. This method also considerably reduces the sensitivity of leakage to gate-length variation. Overall, we obtain leakage reduction of up to 42% and leakage variability reduction of 66% by applying our algorithm to an unoptimized design. Compared to a design optimized with only cell-level biased variants, we achieve up to 17% additional reduction in the mean and up to 39% reduction in the standard deviation of leakage with no runtime overhead. It is also interesting to note that, compared to CLB, TLB significantly reduces the total number of devices biased for comparable delay and leakage improvement, thus minimizing the risk of design-rule violations due to length biasing.

Ongoing work on this project is primarily in the following areas:

1. Use of negatively biased variants for timing optimization and enhanced leakage optimization using hill-climbing algorithms.
2. Improvement of delay/leakage modeling accuracy
3. Added variant generation flexibility by incorporating threshold voltage assignment.

7. Acknowledgements

We would like to thank Mr. Puneet Sharma, Mr. Maogang Wang and Prof. Dennis Sylvester for useful discussions.

8. References

[1] S. Narendra *et. al*., "Leakage Issues in IC Design: Trends, Estimation and Avoidance", *Tutorial, ICCAD,* 2003.

[2] S. Mutah, T. Douseki Y. Marsuya, T. Aoki and S. Shigematru. "I-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS, *JSSC* 1995. Vol. 30. No. 8.00. 847-854.

[3] Y. Oowalti et al.. "A sub-0.1um Circuit Design with Substrate-Over-Biasing". ISSCC. 1998, pp. 88-89.

[4] S. Sirichotiyakul *et. al*., "Duet: An accurate leakage estimation and optimization tool for dual-Vt circuits", *IEEE Transactions on VLSI Systems*, pp. 79-90, April 2002.

[5] P. Gupta et al, " A practical transistor-level dual threshold voltage assignment methodology", *ISQED*, 2005, pp 421-426.

[6] L. Wei et al, "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits", *DAC*. 1998. pp. 489-494.

[7] P. Gupta et al, "Selective gate-length biasing for cost-effective runtime leakage control", *DAC*, 2004. pp. 327-330.

[8] A. Salz, M. Horowitz, "IRSIM: an incremental MOS switch-level simulator", *DAC*, 1989. pp. 173-178

[9] F. Brglez, D. Bryan, and K. Kozminski, "Combinatorial Profiles of Sequential Benchmark Circuits," *Proc. International Symposium on Circuits and Systems (ISCAS)*, pp. 1229–1234, IEEE, 1989.

[10]http://www.opencores.org/project