Soft-Edge Flip-flops for Improved Timing Yield: Design and Optimization

Vivek Joshi, David Blaauw, Dennis Sylvester
University of Michigan, Ann Arbor, MI
vivekj@umich.edu

Abstract **Parameter variations cause high yield losses due to their large impact on circuit delay. In this paper, we propose the use of so-called soft-edge flip-flops as an effective way to mitigate these yield losses. Soft-edge flip-flops have a small window of transparency (ranging from 0.25-3 FO4) instead of a hard edge, allowing limited cycle stealing on critical paths, and thus compensating for delay variations. By enabling time borrowing, soft-edge flip-flops allow random delay variations to average out across multiple logic stages. In addition, they address small amounts of delay imbalance between logic stages, further maximizing the frequency of operation. We develop a library of soft-edge flip-flops with varying amounts of softness. We show that the power and area overhead of soft-edge flip-flops grows directly with the amount of softness. We then propose a statistically aware flip-flop assignment algorithm that maximizes the gain in timing yield while minimizing the incurred power overhead. Experimental results on a wide range of benchmark circuits show that the proposed approach improves the mean delay by 1.9-22.3% while simultaneously reducing the standard deviation of delay by 1.9-24.1% while increasing power by a small amount (0.3-2.8%).**

1 INTRODUCTION

Modern CMOS processes suffer from large variations in transistor parameters such as length and threshold voltage [1]. As the circuit delay has a strong dependence on these parameters, it shows a significant spread due to process variations. This can result in substantial yield losses due to timing failures in the circuit, as nominally sub-critical paths may now become critical. Scaling trends point to an overall increase in the variations with technology scaling. This increase can be attributed to a number of factors, such as increasing difficulty of manufacturing control, increase in atomic scale randomness like variation in dopant profile of the transistor channel, and introduction of new systematic variation generating mechanisms [2]. Hence, there is a need for extensive design and modelling in order to address the variations.

A significant amount of research has focussed on process variation aware analysis. One such approach is corner based static timing methodology [3]. However, traditional corner based design approach mostly leads to overly pessimistic guardbanding [4]. Moreover, while global variations can be approximated by considering appropriate corner cases, it does not provide a statistical way of modelling the variations across dies. Also, as variations are increasing, number of process corners to be considered for true accuracy is becoming too large for computational efficiency. A second approach, namely, statistical static timing analysis (SSTA) has emerged as an alternative analysis approach. While much work has been done on analysis using SSTA [5, 15], there has been a limited work to account for variability in circuit optimization[6, 7, 8, 13, 16]. In [7], the authors extend deterministic optimization using gate sizing approach to the statistical domain, while in [9], authors used geometric programming to address the problem. Most of the optimization in this area uses gate sizing as a solution.

Retiming and so-called *useful* clock skew assignment has been considered as an effective way to balance out the path delays and maximize the frequency of operation. In the past, this has been formulated as a linear programming problem [10]. However, with the exception of [11], these solutions are deterministic in nature and do not consider the effect of process variations. Useful skew assignment also results in a large number of paths pushed to the very edge of satisfying the timing requirements. Hence, even a slight variation in delay on these critical paths can cause a timing based failure. This makes useful skew assignment more effective in improving nominal delay than for improving yield. Furthermore, useful skew assignment is typically performed on an individual flip-flop basis, which is difficult in practice. In the results of this paper, we show that performing useful skew assignment for clusters of flip-flops can address clock skew, but is ineffective for addressing process variation and circuit imbalance.

Another approach to address delay variations in the circuit is to use latch based design. Latches have no hard boundary and are transparent for half a clock period. This means that there can be variations in the data arrival time and still the correct data would be captured by the latch. In [12], the authors present clock scheduling for latches in order to improve yield considering the delay variations. However, latch based design has its own limitations. The most important problem with latch based design is the need for two separate clocks, which means significant power and area overhead. Furthermore, generating two non-overlapping clocks can be difficult in high performance design. Other problems with latch based design include serious hold time violation issues due to the large period of transparency, and difficulties in designing with latches using standard EDA tools. The goal of our work is to use the concept of averaging and cycle stealing, while maintaining the conventional flip-flop based paradigm.

In this paper we propose the use of soft-edge flip-flops as a useful method to address process variation as well as limited circuit imbalance through time borrowing and averaging across stages. The key idea is to delay the clock edge of the master latch so as to create a window of transparency instead of a hard boundary for capturing the data. As shown in Figure 1, the transparency window allows cycle stealing in order to compensate for the difference in delays of the two paths resulting from either circuit imbalance or from local random process variations. Soft-edge flip-flops utilize the fact that

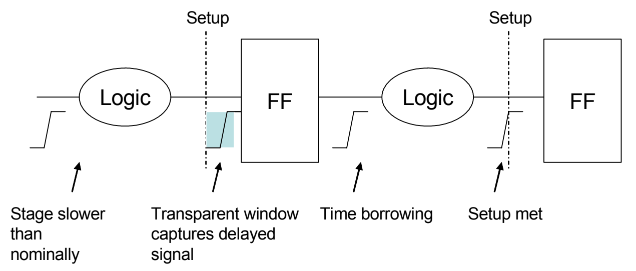
Setup
Setup
Logic
FF
Logic
FF
Stage slower than nominally
Transparent window captures delayed signal
Time borrowing
Setup met

Figure 1. Transparency window compensating for delay variation.

random variations average out along a circuit path, and by allowing averaging across stages they reduce the sensitivity of the design towards process variations. By creating a window of transparency, we increase the hold time of the flip-flop. However, as the window is small, the resulting increase in hold time is not large enough to cause hold time violations. This is verified for all the test circuits as discussed later in the results section.

The softness comes at a cost of power. Hence, we designed a library of soft-edge flip-flops with varying amounts of softness in order to analyze the power overhead involved. A variation aware algorithm was then used to statistically optimize yield with minimum power overhead. When applied to a wide variety of benchmark circuits synthesized, placed and routed with an industrial library, the resulting improvement in the mean delay ranged from 1.9-22.3%, for a power overhead of about 1.8% on average. The corresponding improvement in $\mu+3\sigma$ value for the delay distribution ranged from 4.8-23.6%, which gives an estimate of improvement in timing yield. The yield data was obtained using Monte Carlo simulation.

The remainder of this paper is organized as follows. Section 2 discusses the proposed methodology while Section 3 describes the soft-edge flip-flop assignment technique. The actual design of the soft-edge flip-flops is discussed in Section 4. Sections 5 and 6 give the experimental results and conclusions, respectively.

2 PROPOSED METHODOLOGY

Figure 2 shows a sample setup to demonstrate the effectiveness of soft-edge flip-flops in addressing process variations. The setup consists of two stages of logic(Logic1 and Logic2) separated by a soft-edge flip-flop whose softness is varied from 0 to 100ps in steps of 20 ps. Logic1 has a nominal delay of 1050 ps, and Logic2 has a nominal delay of 1000 ps. Each logic stage has a delay distribution with a global variation of 5%(σ/μ) and another 5% of random variation (σ/μ). In the absence of softness the critical delay is the maximum of the two delays. However, on introducing softness in the flip-flop, for cases where the path preceding the flip-flop (Logic1) is critical, the softness is used to borrow time from the next stage (Logic2) to average out the delays. This leads to the shift in the mean delay as softness is increased. If skew assignment was done, it would have assigned the flip-flop a skew of 25 ps, so as to balance out the nominal delays. And for each sample, the critical delay would have been the maximum of the two delays.

With standard D flip-flop, the distribution of critical delay has a mean of 1059.8 ps, and a standard deviation of 68.4 ps.

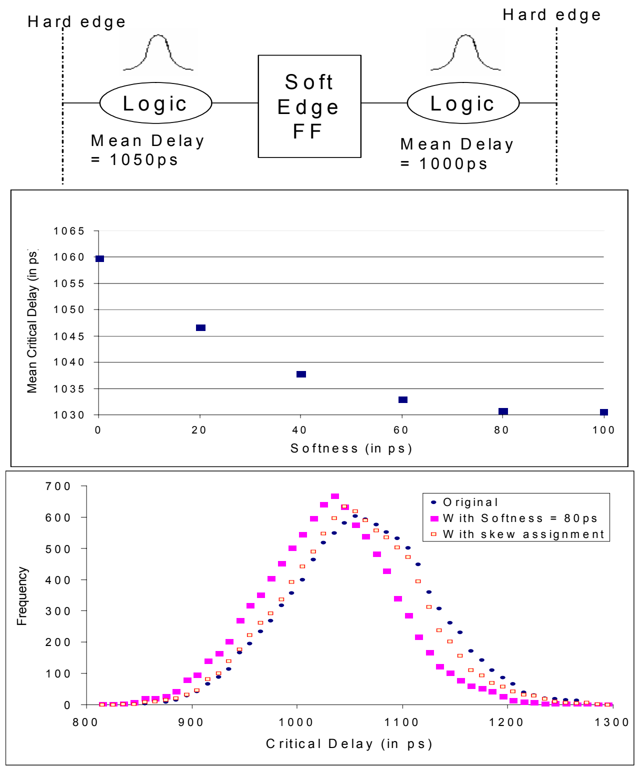

Figure 2. Setup for demonstrating the effectiveness of soft-edge flip-flops and the corresponding plots of mean v/s softness and delay distribution for softness of 0 and 80 ps and for skew assignment.

As shown in the Figure 2, the mean decreases as softness is increased and becomes almost constant after a softness of 80 ps. This is because almost all the cases involving the path preceding the soft-edge flip-flop being critical, have been averaged out by time borrowing to the maximum limit allowed by the slack available. Beyond this point, increasing softness has little effect on the mean. The value of mean for softness of 80 ps is 1030.7 ps with a standard deviation of 63.6 ps. This means that the mean of the delay improved by 29.1 ps which is a fairly large fraction of the initial standard deviation (about 42.5%). For the case of skew assignment, the mean delay is 1053.1 for an improvement of only 6.7 ps with a standard deviation of 66.7 ps. While skew assignment obtains a 25ps improvement in nominal delay, the statistical mean of the delay under process variation is improved by much less. This is caused by the balancing of delay paths in skew assignment, which makes the design more sensitive to process variation and counter acts the improvement in nominal delay. The delay distribution curves plotted for softness of 0 and 80ps along with the distribution for the case of skew assignment are shown in Figure 2. Note that skew assignment shifts the delay distribution curve slightly to the left, while the shift is more significant in the case with softness of 80 ps. This shift corresponds to the improvement in mean. The curve for the case of 80 ps softness is narrower than that for no softness, and this shows up as improvement in the standard deviation.

As seen in this example, there is little improvement in mean if we increase the softness beyond 80 ps. Softness comes at a cost of power overhead, thus, there is a need for an algorithm to intelligently assign softness so as to get best possible improvements in mean for minimum power. We dis-

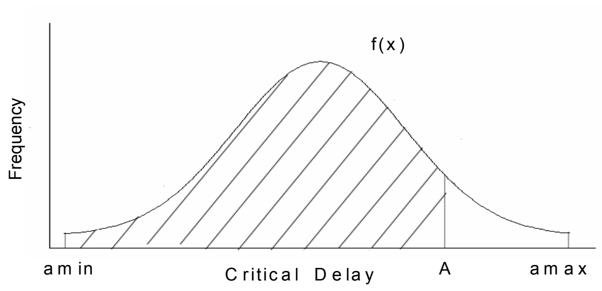

Figure 3. Calculation of yield for a given clock frequency from the delay distribution curve.

cuss the soft-edge flip-flop assignment technique to minimize power overheads in the next section. In order to analyze the power overhead involved, a library of soft-edge flip-flops was designed. The method to design a soft edge flip-flop is to delay the master's clock. This can be done in different ways, and the designs are described in section 4.

3 Soft-edge flip-flop assignment technique

Soft-edge flip-flops have a power overhead associated with them due to the additional delaying elements used to delay the clock to the master latch. The flip-flop assignment problem can thus be formulated as the minimization of power for a given yield constraint. In theory, for a custom design, it is possible to construct a soft-edge flip-flop with any arbitrary amount of softness, by proper sizing. However, for a typical standard cell based design, there will be a library of such flip-flops with varying amounts of softness, and so our goal in this case is to assign flip-flops available in a library. Thus, the design variable which is the amount of softness, is a discrete variable, while the yield constraint is statistical in nature.

The calculation of yield from the statistical data is illustrated in Figure 3. For a given frequency of operation, there is a certain value of critical delay beyond which a circuit fails to meet the timing. This value is shown as A in the figure. So for all values of delay between amin and A, the circuit meets the timing requirements and for delays beyond that it fails. The yield in this case is given by the area of the shaded portion divided by the total area under the curve. Thus, the yield Y is given by equation 1.

$$Y = \frac{\int_{amin}^{A} f(x)dx}{\int_{amin}^{amax} f(x)dx} \quad (1)$$

Clearly, any yield constraint of the form $Y>Y_0$ can be translated into an equivalent constraint on the critical delay A for that given clock period expressed as $A > A_0$. Now, we can express the flip-flop assignment problem as a simple robust integer linear programming (ILP) problem as shown below:

$$\text{Minimize power } P = \sum_{f=1}^{F} \sum_{k=1}^{n} P_k v_{fk} \text{ subject to:}$$

$$Y(x_1, x_2, \ldots, x_F) > Y_0 \quad (2)$$

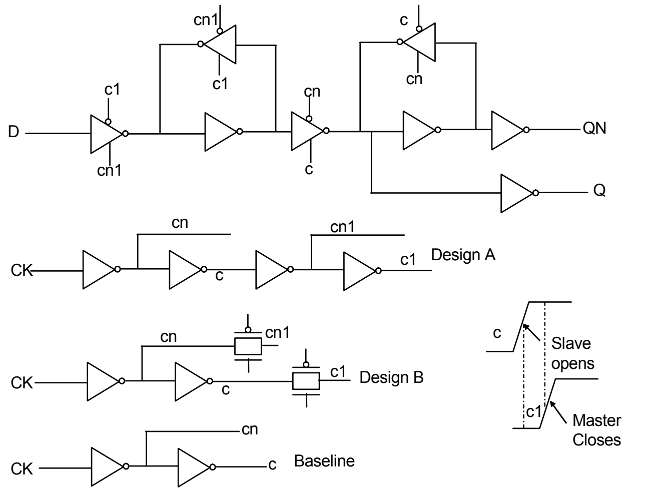

Figure 4. Designs for the soft-edge flip-flop.

$$\sum_{k=1}^{n} v_{fk} = 1 \quad (3)$$

$$\sum_{k=1}^{n} S_k v_{fk} - x_f = 0 \quad (4)$$

$$v_{fk} \in \{0, 1\} \quad (5)$$

where F is the total number of flip-flops in the circuit, and n is the total number of quantization levels of softness that a flip-flop can have (softness is a discrete design variable and can be varied in steps). v_{fk} is the kth quantization level variable associated with the fth flip-flop, S_k is the softness associated with the kth quantization level, and x_f is the value of softness for the fth flip-flop. Note that softness associated with the first quantization level is 0, and this corresponds to the standard D flip-flop from the library. P_k is the power consumed by a flip-flop with softness S_k. The power for the rest of the circuit remains the same, and the only overhead is due to the assignment of softness to the flip-flops. Constraint (2) is the simple yield constraint while constraints (3), (4) and (5) together make sure that each flip-flop has only one value of softness assigned to it out of the n possible values corresponding to the n levels of quantization. Setup and hold time constraints are essentially captured in the yield constraint, and hence they need not be expressed independently. The yield constraint makes the problem variation aware, as yield depends on the delay distribution caused due to process variations. There have been many advancements in efficiently solving robust integer linear programming problems. However, a typical industrial circuit consists of several thousands of flip-flops and the runtimes for solving the corresponding ILP will most likely be very large. Hence, we use a less complex greedy heuristic to assign softness, while the ILP is considered for possible future extensions.

We use a greedy algorithm 'GREEDY_SOFTNESS( )' to solve this problem of soft-edge flip-flop assignment. The idea is to search for the most critical path in the circuit (considering process variations), and increase the softness of the flip-flop following the path by one step. This is to be repeated till the yield requirements are just met, or when there ceases to be any appreciable improvement upon increasing softness. The second criterion ensures that we stop assigning further softness if the improvement gained by that step is not appreciable, and that we stop in case the yield target is not achievable for the given constraint. The pseudo code for GREEDY_SOFTNESS( ) is as follows:

```
GREEDY_SOFTNESS( )
1 continue <-- true
2 while continue == true
3    find most critical path P
4    find current yield Y
5    check <-- false
6    if P ends at a flip-flop f's input
7       then increase softness of the flip-flop f by one step
8          check <-- true
9    find new yield $Y_{new}$
10   if $Y_{new} > Y_0$
11      then continue <-- false
12   else if ($Y - Y_{new}$) is not appreciable
13      then continue <-- false
14         if check == true
15         then decrease the softness of flip-flop f by one
                     step
```

As we show later in the results section, we obtain near maximum delay improvements with small power overheads (less than 3%), by using this greedy algorithm for assigning soft-edge flip-flops.

4 Soft-edge flip-flop design

Figure 4 shows the design of a soft-edge flip-flop. Like a standard D flip-flop, it is constructed using back to back master/slave latches. Each latch consists of a tristate inverter feeding into the latch followed by a cross-coupled inverter pair. For a standard D flip-flop, at the positive edge of the clock, the tristate feeding the master closes while the tristate forming the feedback loop opens thereby latching the value into the master latch. At the same time, the tristate feeding the slave opens, while the feedback tristate closes, making the slave transparent. This allows the outputs Q and QB to be evaluated based on the value latched in the master. So, in order to be displayed correctly at the output, the data should arrive sometime before the rising edge of the clock. This time is called the setup time and it defines the hard edge before which the data should arrive for a standard D flip-flop.

In order to create a window of transparency (soft edge), we need to delay the clock supplied to the master latch by a small amount. The amount of delay is the period for which both the master and the slave are transparent together. This is because when the slave's clock makes a transition causing it to become transparent, the tristate feeding the master is also open as its clock is a delayed version of the slave clock. In this way, even though slave is transparent, a change in data

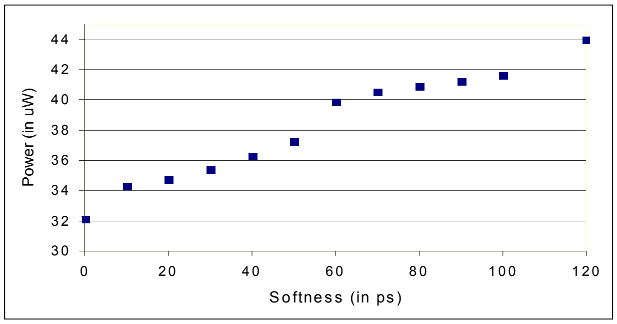

Figure 5. Power consumption for the designed soft-edge flip-flops.

value can still change the state of master which, in turn, will change the value of output. As shown in Figure 4, signal c1 which is fed to the master latch is the delayed version of signal c which controls the slave latch. For the period between the two vertical lines, slave has opened while the master has not closed and this is the period of transparency or the amount of softness. This softness allows time borrowing if the stage following the flip-flop has a slack.

As in the original D flip-flop c was one inverter delayed version of cn, in our new design c1 and cn1 (supplied to the master) should be such that c1 is more delayed than cn1. This ensures that our design is as close to the original as possible, if we match the rise time for the new signals to the original ones. Keeping this in mind, there are two ways of delaying the clock. One is to simply insert two more inverters after c as shown in design A. Here, we need to size the inverter chain such that cn1 and cn have the same rise time, while c and c1 have the same rise time and this should be close to the rise time of the corresponding signal in the standard D flip-flop. Along with the risetime we need to ensure that the delay between cn1 (c1) and cn (c) is the same as the value of softness required. For higher values of softness, more inverters might be added to the chain of inverters.

However, for design A, cn1 (c1) is two inverter delayed version of cn (c). This means that there is a lower limit to the softness that we can assign which is determined by the minimum delay of two inverters. For lower values of softness, we use design B, which uses pass gates for delaying the signals. In design B as shown in the Figure 4, c1 and cn1 are delayed versions of c and cn respectively, which have been delayed using pass gates. By appropriate sizing much lower values of softness are possible. In our designed library, we use design B for getting the softness of 10 ps and 20 ps, while design A is used for all other values.

Figure 5 shows the power values for flip-flops with different values of softness. Note that softness of 0 corresponds to the standard D flip-flop. Power has been measured using Hspice, assuming the switching activity of the data to be 0.1. The sudden increase in power when we go from softness of 50 ps to 60 ps, is due to the fact that two more inverters have to be added to the inverter chain in order to get values of softness greater than 50 ps. A similar increase can be seen when increasing the softness from 100ps to 120ps. Table 1 gives the power and area overheads for the library of soft-edge flip-flops. Area overheads are computed by laying out the soft-edge flip-flops and comparing it to the area of standard cell. The power and area overheads for the flip-flop with the largest softness (120ps) are 36.8% and 45%. However, we find that only a small fraction of flip-flops need this level of softness. Hence, the power overhead was found to be a very small percentage (0.3-2.8%) of the overall circuit power, as reflected in the experimental results which are discussed in the next section.

5 Experimental results

The proposed soft-edge flip-flop assignment technique was implemented based on an industrial $0.13\mu m$ CMOS technology. All the test circuits were synthesized, placed and routed using commercial tools. The test circuits ranged in size from 119-36544 gates. The physical placement details were used to cluster the flip-flops for the purpose of clustering based skew assignment. We compare our results against skew assignment for cluster sizes of 30 (large), 15 (moderate), and 5 (small). Based on industrial estimates [14], we use a switching activity of 0.1 for the primary inputs of the circuit, while measuring power. In this work, we consider channel length as the source of variability. We consider inter-die, spatially correlated intra-die as well as random components of variation (5% σ/μ due to each component).

Table 2 summarizes the main results for the proposed approach and compares them with clustering based skew assignment for ISCAS89 benchmark circuits and two DSP circuit implementations ('Viterbi1' and 'Viterbi2'). The first three columns give the name and size of each test circuit. The fourth and fifth columns give the mean and standard deviation of delay for the original test circuit without any optimization. The next three pairs of columns give the percentage improvement in mean and standard deviation of delay, by using clustering based skew assignment technique for cluster sizes of 30, 15 and 5. The next pair of columns gives the percentage improvement in mean and standard deviation by using soft-edge flip-flops. Note that this is the best possible improvement which can be achieved by using soft edge flip-flops. The last two columns provide an estimate of how this improvement compares with that for skew assignment technique for a cluster size of 5. Essentially, it is the ratio of the improvement achieved by using soft-edge flip-flops, to that achieved by using clustering based skew assignment for cluster size of 5. A cluster size of 5 is difficult to achieve. This is because typically at least a dozen flip-flops are driven by a single clock regenerator.

The results clearly show that our approach gives significantly better improvements in mean and standard deviation even when compared to skew assignment for a cluster size of 5, which is difficult to achieve. We get improvements of up to 22.6% (8.9% on average) in the mean and up to 24.1% (10% on average) in the standard deviation of the delay, over the original circuit. As compared to the improvement using skew assignment with a cluster size of 5, our approach gives improvements of up to 8.8X (2.6X on average) in the mean and up to 9.9X (2.6X on average) in the standard deviation of the circuit delay.

Figure 6 shows sample delay distribution curves of one of the larger circuits (Viterbi1) for the original circuit without any optimization, for the case of skew assignment (cluster size =5), and for the case of using soft-edge flip-flops. As shown in Table 2, the mean delay improves by 2.3% for skew assignment, and by 6.1% for the case in which soft-edge flip-flops are used. This change is depicted in the shifting of the curves towards left. The shift is less for skew assignment (lesser improvement), and a greater shift is observed for soft-edge flip-flops. Also, the curves become narrower as we move from the original circuit to skew assignment and to soft-edge flip-flop assignment. This corresponds to the improvement in the standard deviation by using the two approaches.

Table 3 shows results for soft-edge flip-flop assignment using the greedy algorithm. Columns two and three give improvement achieved in mean and express it as a fraction of the best possible improvement that we can get using soft-edge flip-flop assignment (reported in Table 2). The next column gives the improvement in $\mu+3\sigma$ value for the delay distribution, which provides an estimate for the improvement in timing yield. It shows improvement of 11.9% on average. Last two columns give the percentage power overhead and the percentage of flip-flops that were assigned some softness. Power overhead is computed based on the extra power used for making the flip-flops soft, as compared to the power for the original circuit, assuming a switching activity of 0.1 for the primary inputs. On an average, we get 0.97 of the best possible improvement in mean by using the greedy algorithm for a power overhead of 1.8%. The results show that we get very close to the best possible improvement by using greedy algorithm, with a small power overhead. For a more complex

Table 1. Area and power overheads for soft-edge flip-flops.

Softness	Percentage power overhead	Percentage area overhead
10	6.8	18
20	8.2	18
30	10.2	18
40	12.9	18
50	16.0	18
60	24.1	32
70	26.2	32
80	27.2	32
90	28.3	32
100	29.5	32
120	36.8	45

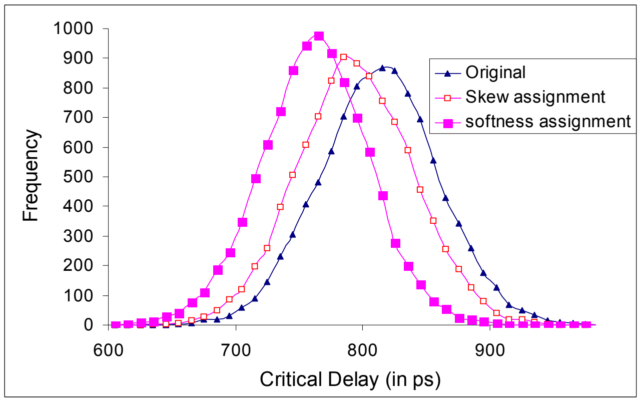

Figure 6. Delay distributions for the circuit Viterbi1- for the original, skew assignment and softness assignment cases.

Table 2. Improvement in mean delay for several circuits by soft-edge flip-flop assignment as compared to skewing.

Delay (ps)	No. of gates	No. of flip-flops	Values for Original Circuit		Percentage Improvement by skew assignment (Cluster Size = 30)		Percentage Improvement by skew assignment (Cluster Size = 15)		Percentage Improvement by Skew assignment (Cluster Size = 5)		Percentage Improvement by Softness		Improvement by softness compared to that by clustering (size=5)	
			μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
s298	119	14	345.5	15.2	0.0%	0.0%	0.0%	0.0%	1.6%	2.8%	14.3%	6.3%	8.8X	2.2X
s344	160	15	567.5	30.8	0.0%	0.0%	0.0%	0.0%	5.6%	3.5%	17.1%	11.1%	2.9X	3.2X
s349	161	15	573.5	31.3	0.0%	0.0%	0.0%	0.0%	7.4%	4.4%	17.2%	14.0%	2.3X	3.1X
s386	159	6	351.5	15.6	0.6%	3.2%	0.6%	3.2%	0.6%	3.2%	2.6%	4.1%	4.6X	1.2X
s400	164	21	490.5	25.4	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	11.5%	15.6%	NA	NA
s420	218	16	731.4	42.4	0.0%	0.0%	4.8%	7.4%	5.9%	7.5%	6.8%	15.2%	1.2X	2.0X
s510	211	6	521.5	27.6	1.2%	0.4%	1.2%	0.4%	2.0%	0.9%	6.5%	8.7%	3.3X	9.9X
s526	194	21	488.5	25.3	0.0%	0.0%	0.0%	0.0%	15.6%	13.3%	22.6%	22.8%	1.4X	1.7X
s820	289	5	531.2	28.3	1.1%	4.2%	1.1%	4.2%	1.1%	4.2%	4.1%	5.5%	3.8X	1.3X
s832	287	5	537.5	28.7	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	3.9%	5.1%	NA	NA
s838	446	32	1144.4	71.5	4.4%	0.9%	4.4%	0.9%	4.8%	1.1%	6.7%	3.6%	1.4X	3.4X
s1196	529	18	644.5	36.3	4.9%	2.8%	6.4%	8.1%	6.8%	10.7%	16.9%	24.1%	2.5X	2.2X
s1238	508	18	683.4	39.0	3.7%	4.6%	3.7%	4.6%	3.7%	4.6%	9.5%	10.3%	2.5X	2.3X
s1423	657	74	1688.5	99.2	0.3%	0.1%	0.4%	0.3%	0.5%	0.3%	3.1%	1.9%	6.8X	5.9X
s1488	653	6	579.4	31.7	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	2.1%	2.7%	NA	NA
s15850	9812	534	1456.0	92.5	0.7%	0.6%	1.8%	1.9%	1.8%	1.9%	6.0%	5.9%	3.4X	3.0X
s35932	16065	1728	480.4	24.0	3.7%	5.1%	3.7%	5.1%	3.7%	5.1%	8.9%	12.2%	2.4X	2.4X
s38417	15106	1636	1046.6	62.8	1.8%	2.2%	3.1%	4.6%	5.0%	6.4%	8.1%	11.4%	1.6X	1.8X
Viterbi1	12024	4215	812.3	46.8	0.3%	0.3%	0.3%	0.3%	2.3%	3.5%	6.1%	6.1%	2.7X	1.8X
Viterbi2	36544	5788	1744.3	70.0	0.2%	0.7%	0.3%	1.7%	0.5%	2.8%	3.3%	12.8%	6.4X	4.5X
Average					**1.1%**	**1.2%**	**1.6%**	**2.1%**	**3.4%**	**3.8%**	**8.9%**	**10.0%**	**2.6X**	**2.6X**

approach, the small gains are likely to be overweighed by the runtime overheads.

Figure 7 shows the plot of power overhead versus improvement in mean for the circuit Viterbi1. The point corresponding to the improvement obtained by the greedy algorithm has been circled in the figure. Note that beyond the circled point, increase in power is more significant as compared to the further improvement in mean that we can obtain. Figure 8 shows the corresponding distribution of softness after running the greedy algorithm on the circuit Viterbi1. Maximum softness used in this case is 80 ps.

Soft-edge flip-flops have a window of transparency instead of a hard edge. So, it is necessary to check for possible hold time violations. Figure 9 shows the short path slacks for all test circuits after soft-edge flip-flop assignment. Short path slack is the amount of time by which the minimum path delay meets the hold time violation constraint. As shown in the figure, the hold time constraints are met comfortably, and there is substantial slack for the circuit to meet the constraints after variations. This is expected because the window of transparency is very small even for the largest value of softness (about 3FO4), and so the resulting increase in hold-time is not large enough to cause hold time violations.

6 Conclusions

In this paper, we proposed the use of assigning soft-edge flip-flops as a process variation tolerant technique for fine grained balancing of a circuit, in order to improve the timing yield. Soft edge flip-flops allow time borrowing and averaging across stages, making the design less sensitive to process variations by taking advantage of the fact that random variations average out. We constructed a library of soft-edge flip-flop variants, with softness of up to 120 ps (about 3FO4). As soft-edge flip-flops have a power overhead associated with them, we used a statistically aware greedy algorithm to intelligently assign these flip-flops in order to minimize the power overhead. Experimental results show that as compared to clustering based skew assignment technique, our approach provides improvements of up to 22.3% (8.6% on average) in the mean and up to 24.1% (10% on average) in the standard deviation of the delay, with a very small power overhead (0.3-2.8%).

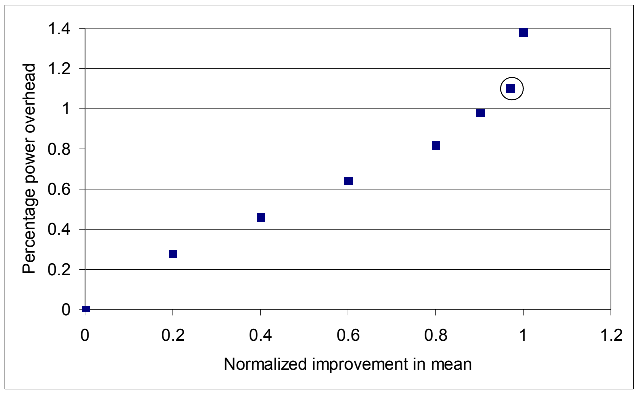

Figure 7. Power overhead versus improvement in mean delay for the circuit Viterbi1. Circled point corresponds to the improvement obtained by greedy algorithm.

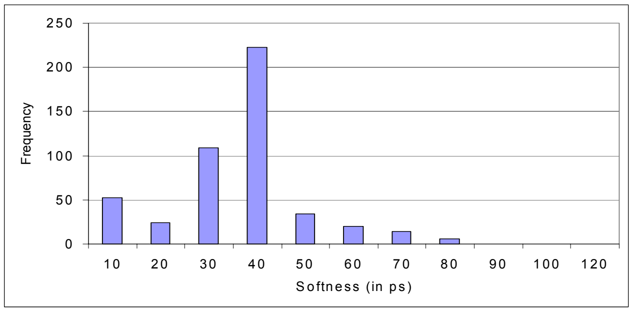

Figure 8. Distribution of softness for the circuit Viterbi1.

ACKNOWLEDGMENT

The authors acknowledge the support of Gigascale Systems Research Focus Center, one of the five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. This work was also supported in part by the NSF and STARC.

Table 3. Results for softness assignment using greedy algorithm

Circuit	Percentage improvement in μ	Improvement as a fraction of best possible improvement	Percentage improvement in $\mu + 3\sigma$	Percentage overhead in power	Percentage of flip-flops assigned softness
s298	13.8%	0.96	16.5%	1.2%	64.3%
s344	16.9%	0.98	17.5%	1.8%	20.0%
s349	16.5%	0.96	19.7%	2.2%	33.3%
s386	2.4%	0.96	7.0%	1.9%	50.0%
s400	11.4%	0.98	13.3%	1.8%	23.8%
s420	6.4%	0.94	13.2%	2.0%	25.0%
s510	6.2%	0.96	10.9%	0.9%	33.3%
s526	22.3%	0.99	23.6%	2.8%	66.7%
s820	3.9%	0.94	9.6%	2.6%	80.0%
s832	3.7%	0.95	8.7%	2.6%	80.0%
s838	6.4%	0.95	11.1%	2.1%	60.0%
s1196	16.6%	0.98	19.3%	1.1%	27.8%
s1238	9.3%	0.98	10.9%	2.7%	27.8%
s1423	3.1%	0.98	4.8%	2.6%	23.0%
s1488	1.9%	0.93	8.6%	0.3%	16.7%
s15850	5.9%	0.99	7.2%	2.7%	32.2%
s35932	8.7%	0.98	11.5%	1.2%	9.3%
s38417	8.0%	0.99	9.7%	1%	9.5%
Viterbi1	6.0%	0.98	7.7%	1.1%	11.4%
Viterbi2	3.3%	0.98	6.1%	1.1%	9.6%
Average	**8.6%**	**0.97**	**11.9%**	**1.8%**	**35.2%**

REFERENCES

[1] S. Nassif, “Delay variability: sources, impacts and trends,” Proc. of ISSCC, pp. 368-369, 2000.
[2] Y. Taur et al., “CMOS scaling into nanometer regime,” Proc. of the IEEE, no. 4, pp. 486-504, 1997.
[3] S.R. Nassif, A.J. Strojwas and S.W. Director, “A methodology for worst-case analysis of integrated circuits,” IEEE Trans. Computer-Aided Design, vol. CAD-5, pp.104-113, Jan. 1986.
[4] C. Visweswariah, “Death, taxes and failing chips,” Proc of DAC, pp. 343-347, June 2003.
[5] H. Chang and S. Sapatnekar, “Statistical timing analysis considering spatial correlations using a single pert-like traversal,” Proc. of ICCAD, pp. 621-625, 2003.
[6] E. Jacobs and M. Berkelaar, “Gate sizing using a statistical delay model,” Proc. of DAC, pp. 283-290, 2000.
[7] P. Seung et al., “Novel sizing algorithm for yield improvement under process variation in nanometer technology,” Proc. of DAC, pp. 454-459, 2004.
[8] M. Mani and M. Orshansky, “A new statistical optimization algorithm for gate sizing,” Proc. of ICCD, pp. 272-277,2004.
[9] J. Singh, V. Nookala, Z. Q. Luo, and S. Sapatnekar, “Robust gate sizing by geometric programming,” Proc. of DAC, pp. 315–320, 2005.
[10]X. Liu, M.C. Papaefthymiou, and E.G. Friedman, “Maximizing performance by retiming and clock skew scheduling,” Proc. of DAC, pp. 231-236, 1999.
[11]V. Nawale and T.W. Chen, “Optimal useful skew scheduling in the presence of variations using robust ILP formulation,” Proc. of ICCAD, pp 27-32, 2006.
[12]A. P. Hurst and R. K. Brayton, “Latch based design under process variation,” IWLS 2006.
[13]A. Srivastava, D. Sylvester and D. Blaauw, “Statistical optimization of leakage power considering process variations using dual V_{th} and sizing,” Proc. of DAC, pp. 773-778, 2004.
[14]N. Magen, A. Kolodny, U. Weiser, N. Shamir, “Interconnect power dissipation in a microprocessor,” SLIP 2004.
[15]H. Chang, V. Zolotov, S. Narayan, and C. Visweswariah, “ Parameterized block-based statistical timing analysis with non-gaussian parameters, nonlinear delay functions,” Proc. of DAC, pp. 71-76 , 2005.
[16]K.Chopra, S. Shah, A. Srivastava, D. Blaauw and D. Sylvester, “Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation,” Proc. of ICCAD, pp. 1023-1028, 2005.

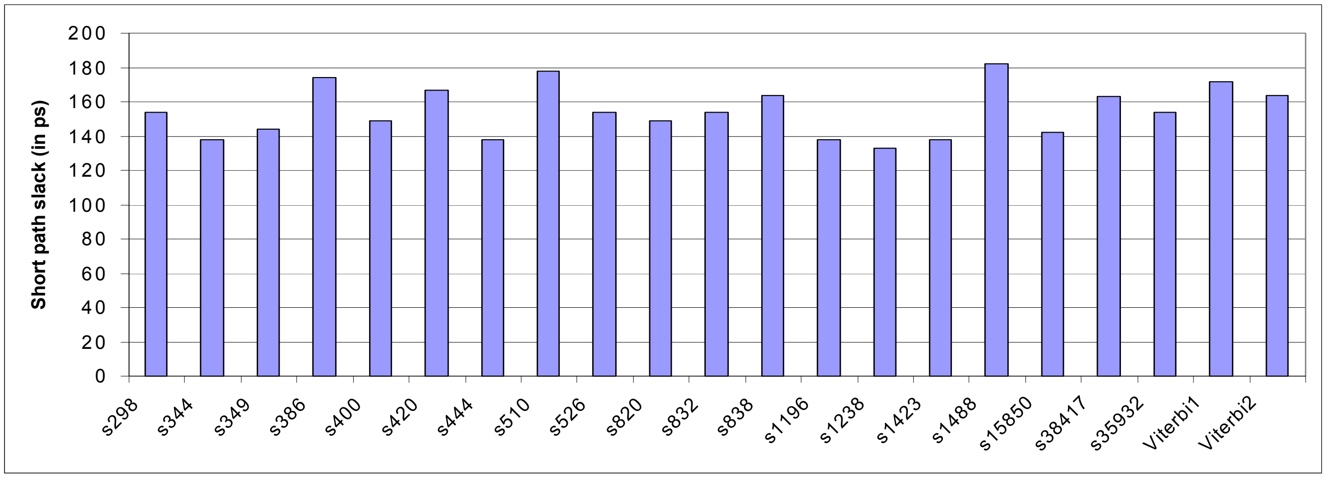

Figure 9. Short path slack after soft-edge flip-flop assignment.