Efficient Computation of Discharge Current Upper Bounds for Clustered Sleep Transistor Sizing

A.Sathanur* A.Calimera* L.Benini◇ A.Macii* E.Macii* M.Poncino*

* Politecnico di Torino
10129 Torino, ITALY

◇ Università di Bologna
40136 Bologna, ITALY

Abstract

Sleep transistor insertion is a key step in low power design methodologies for nanometer CMOS. In the clustered sleep transistor approach, a single sleep transistor is shared among a number of gates and it must be sized according to the maximum current that can be injected onto the virtual ground by the gates in the cluster. A conservative (upper bound) estimate of the maximum injected current is required in order to avoid excessive speed degradation and possible violations of timing constraints. In this paper we propose a scalable algorithm for tightening upper bound computation, with a controlled and tunable computational cost. The algorithm leverages the capabilities of state-of-the-art commercial timing analysis engines, and it is tightly integrated into standard industrial flow for leakage optimization. Benchmark results demonstrate the effectiveness and efficiency of our approach.

1 Introduction

Leakage power has increased incessantly with the scaling of CMOS technology in the nanometer regime. Even though a number of new leakage sources are emerging as transistors shrink, the main contributor to overall leakage is still drain-to-source leakage through transistors that are nominally off. Drain-to-source leakage can be reduced by manufacturing slower high-threshold transistors, but the performance penalty associated with this solution is often unacceptable. Hence, a number of circuit, logic, architectural and system-level techniques have been devised in recent years to reduce leakage power [1].

One of the most popular approaches to leakage power reduction relies on the insertion of sleep transistors which are placed between the ground (or V_{dd}) terminal of the gates and the ground (or V_{dd}) distribution network. Several different embodiments of this technique have been proposed in the literature based on the granularity at which sleep transistors are inserted. In the *distributed* sleep-transistor approach, these devices are added on an cell-by-cell basis [1]. The main issues with this technique are: High area overhead and large number of sinks for the activation signal net, which becomes extremely hard to route. On the other side of the granularity spectrum, the approach known as *power gating* insert very large sleep transistors on the root of the power distribution networks of large sub-units (typically many thousands of gates). This is by far the most common sleep transistor insertion technique and it is supported by most state-of-the-art industrial design flows. The main shortcoming of this approach is the long transition delays between sleep and active states.

On a middle ground between the two approaches mentioned above, a very promising alternative is *clustered sleep transistor insertion*, where a subset of a gate network (i.e., a gate cluster) is connected to a single sleep transistor. This reduces overhead with respect to distributed sleep transistors, while reducing turn-on transition times with respect to power gating. This paper addresses one of the key steps in automated sleep transistor insertion, namely the estimation of the maximum current I_{max} produced by a gate cluster.

Estimation of maximum current is challenging for two reasons. First, exact computation of I_{max} is computationally unfeasible, because it requires the solution of a number of large-size weighted satisfiability problems. On the other hand, we do not want to grossly oversize the sleep transistor to avoid the ensuing overheads. Several attempts have been made in the past to compute *upper bounds* of I_{max} at an acceptable computational cost. This work presents a novel technique for obtaining an upper bound that can be computed efficiently. We leverage a state-of-the-art timing analysis engine, and embed it in an optimization loop that iteratively tightens an initial loose upper bound computed using basic min-max delay analysis. Experimental results on a set of benchmarks demonstrate that our solution achieves maximum current upper bounds of 12% tighter, on average, than a traditional approach based on the plain overlapping of switching windows.

978-3-9810801-2-4/DATE07 © 2007 EDAA

2 Previous Work

The estimation of the maximum current drawn by a circuit in a given clock cycle is a well-studied problem. Maximum current (and thus maximum power) is in fact important because it is related to the reliability of a VLSI circuit [3]. Nowadays, besides reliability issues, estimating the maximum current is also extremely important for properly sizing the sleep transistors. several techniques for sizing sleep transistors based on maximum current estimation have recently appeared in the literature.

In [4], the authors address the issues of sleep transistor sizing based on mutually exclusive discharge current pattern. They recursively apply their methodology to size a sleep transistor for an entire circuit consisting of smaller sub-modules. They show that grouping the gates which have mutually exclusive current discharge pattern can result in very optimal sleep transistor sizes. This method has the drawback of giving an upperbound estimate of the maximum current that may be too pessimistic, leading to an oversized sleep transistor.

In [5], [6] and [7], the authors propose a method of constructing a current envelope for the entire cluster of gates, which will share a sleep transistor, by computing the expected discharge current of the gates. They multiply the maximum discharge current of each gate with its switching activity and refer it as expected discharge current of each gate. This method may not be suitable for designing sleep transistors because, since they compute the expected discharge current instead of peak current, their value signifies the average or mean value of the peak current distribution of the gate cluster. Since the sleep transistor sizing phase has to rely upon extreme value of the discharge current instead of just the average current, this method can lead to very inaccurately sized sleep transistors for very complex and large circuits. In [8], the authors use the same methodology as in [5] to build a current envelope based on expected discharge current of the gate cluster. But they show that sizing based on timing criticality can lead to lower sleep transistor sizes as compared to [5] and [4]. Among other techniques, in [9] the authors propose a heuristic for estimating the maximum discharge current based on artificially setting a gate to switch. This is based on the correlation of the gate with the other gates. Even though this method performs an upper bound peak current estimation, it is a simulation based methodology and hence, for complex circuits with a large number of inputs, the number of simulations runs to achieve a good upper bound value might be large and hence could be very time consuming. Our technique on the contrary uses static timing analysis to compute the current upper bound and thus is completely independent of the population of the simulation data needed. This is very important because as the circuits grow in size the technique in [9] requires more simulation data points to achieve accuracy which requires very high computation time. In [10], the authors show that considering not only topological aspect of the circuit, but also considering the functionality of the circuit, can help in optimal clustering and hence sizing of the sleep transistor. This method has the disadvantage to be too time consuming since it performs functionality based relation graph construction.

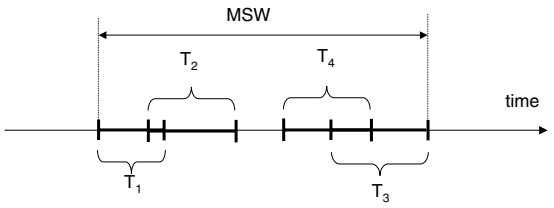

Figure 1. Switching Windows and MSW.

3 Maximum Current Estimation

The problem of maximum discharge current estimation can be stated as follows. Given a set of gates in a cluster sharing the same sleep transistor, we need to compute an upper bound to the current value this set of gates will discharge (in order to size the sleep transistor accordingly). The difficulty in computing an upper bound for that value is that, for a very complex circuit having thousands of gates and many input ports, it is infeasible to find the input vector transition that discharges the peak current value using dynamic simulation. Our technique does not require any dynamic simulation, in contrast to other techniques and it only utilizes static timing analysis to achieve the task (thus being very fast). We first propose our iterative algorithm and follow it up with a simple example.

3.1 Preliminaries and Definitions

We define the *switching window* of a gate under an input pattern as the interval between the arrival time and the output transition of the gate. There is one switching window for each path through a particular gate. The width of the switching window of a gate for a given path is equivalent to the propagation delay of the gate for a rising or falling transition.

For a given gate in the cluster, the *maximum switching window (MSW)* is defined as the time interval encompassing all possible switching time intervals of the gate.
Figure 1 shows an example of MSW, where for simplicity only 4 switching windows are shown. Here $T1, T2, T3$ and $T4$ denote the possible switching time intervals for the gate.

If a gate has n such switching windows, the set of values $T1, \ldots, Tn$ is called the *full switching window (FSW)* of the gate.

3.2 Maximum Current Estimation Algorithm

Figure 2 shows a high-level pseudo-code of our maximum current estimation algorithm. The first step (Line 1)

```
1.  Compute MSWs for all the gates
2.  Construct current plot
    while not convereged {
3.      Identify gates contributing to maximum current
4.      Perform gate ordering
5.      Extract FSWs
6.      Update current plot
    }
```

Figure 2. Algorithm for Maximum Current Estimation.

consists of computing the MSW for each gate in the cluster. The motivation behind extracting the MSW and not all the switching times of a gate is that, for a complex circuit and for a gate very deep in the logic, there are possibly a very large number of switching time intervals, each one corresponding to a path being activated through a gate. Moreover, it is very time consuming to extract all these windows for all the gates, which form a cluster. Conversely, it is very fast to extract the MSW for each gate. In fact, it only requires the calculation of the first (earliest) switching window $T1$ and of the last (latest) one Tn; the MSW is simply obtained as the time interval encompassing $T1$, Tn and the time between them as well as shown in the Figure 1. We thus need to extract only two switching time intervals for each gate in the cluster, which can be accomplished very fast.

The second step (Line 2) consists of the construction of a plot of current over time (hereafter, the current plot), which records the gates that switch at a particular time interval and hence the total discharge current in that time interval. This plot is the superposition over time of rectangles whose base corresponds to the MSWs of the individual gates of the cluster (which will in general partially overlap), and whose height corresponds to the current drawn by the gate (e.g., derived by a technology library). From this current plot, we obtain the time interval during which the maximum current discharge occurs and the gates that contribute to this current discharge.

Figure 3 shows an example of current plot, in which five MSWs (a to e) are shown. The interval in which the current drawn is maximum is shown, corresponding to the overlapping of the c, d, and e MSWs.

Here, we emphasize the usefulness of the conservative

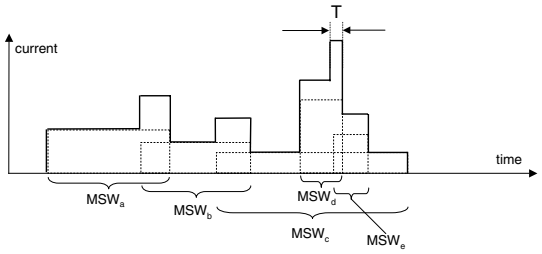

Figure 3. Example of Current Plot.

approach to quickly obtain the MSW for all the gates in the cluster. The reason behind this is that it allows us to identify (based on MSWs) the time interval during which the maximum current discharge occurs. We are only interested in extracting the detailed FSW information for those gates which contribute to this maximum value (disregarding those that do not contribute to the maximum discharge current).

From the current plot we find the gates, which contributes to this maximum current value (Line 3). Gate ordering (Line 4) on the set of gates identified in Line 3 is done to maximize the probability of non-overlapping switching of the gates that contribute to this maximum current value. This part is explained in more detail in the next section.

After this ordering of the gates, we perform the FSW extraction (Line 5) gate by gate or for a set of gates. We assume that only a subset of these gates, which contribute to the maximum current, is sent for FSW extraction. Once we extract all the possible switching time intervals for the subset of gates, we update the current plot (Line 6) and then we repeat the process of obtaining the gates, which contribute to the new maximum current.

The process is repeated until we converge. In the worst case, we have to extract the FSW for all the gates in the cluster and compute the maximum current. Hence we guarantee the convergence of our algorithm.

3.3 Gate Ordering

We use gate ordering to speed up the process of FSW extraction of a set of gates that contribute to maximum current value. In the algorithm, the interval that exhibits the current peak moves as the iteration progresses, each time reducing or maintaining a constant value of the maximum discharge current. Figure 4 shows this effect. Consider the time interval during which a current peak occurs; there will typically be a set of gates that have switching times that extend on both sides of this time interval. For instance, in Figure 3, the MSW of gate c extends both to the left and the right of the time interval T (whereas d extends on the left side of T only, and e to the right only).

For gates such as c, there is a good probability that it will not switch at T which we call as the *virtual switching window*, which will disappear when we extract the complete FSW for that particular gate. Then, the previously found current peak actually collapses and might move onto a new time period (*peak current hopping*). Since it is desirable that

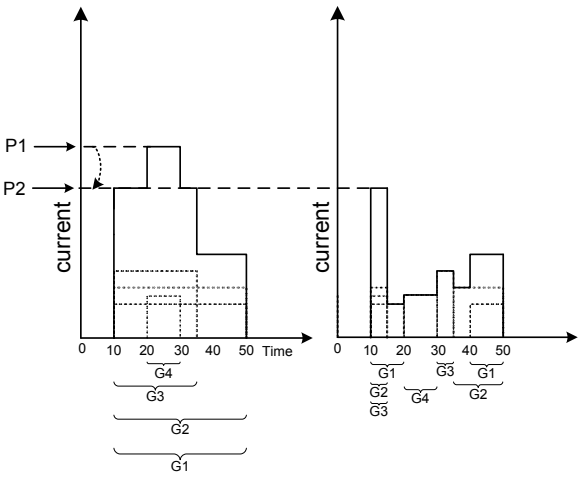

Figure 4. Peak Current Hopping through Algorithm Iteration.

we hop to new current peak as quickly as possible to converge on the final maximum discharge current, it is essential that we send gates having this virtual switching window for FSW extraction. This technique speeds up the algorithm since we do not perform FSW extraction for all the gates that contributes to this maximum current.

3.3.1 Speeding up FSW extraction

As already mentioned, for a gate in a complex circuit at a considerable logic depth, there are a very large number of switching windows. Some of these are overlapping windows and hence do not constitute unique switching times for the gate. In general, there is a perticular distribution of the number of switching windows for each gate in a circuit. It is important to point out here that there tend to be a large number of switching windows in the middle than those towards the extremes. To achieve convergence in extracting these non-overlapping switching windows for each gate, we do a two-way extraction of the switching windows as shown in Figure 5.

First, we extract, for each gate, a certain number of early switching windows $\mathcal{W}_{min}$ and a certain number of late switching windows $\mathcal{W}_{max}$. Then, we check if the latest switching window of $\mathcal{W}_{min}$ and the earliest switching window of $\mathcal{W}_{max}$ overlap. If they do, then we stop at this point. Then we merge the overlapping switching windows and construct unique switching windows from $\mathcal{W}_{min}$ and $\mathcal{W}_{max}$. There is the possibility that even after extracting a few hundred $\mathcal{W}_{max}$ and $\mathcal{W}_{min}$ switching windows, they do not converge.

In that case, we make a check to see if the extracted $\mathcal{W}_{max}$ windows reach a threshold θ_{max} and if the extracted

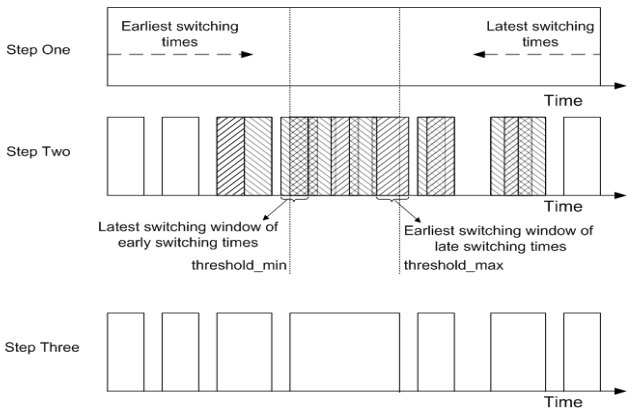

Figure 5. Full window extraction method.

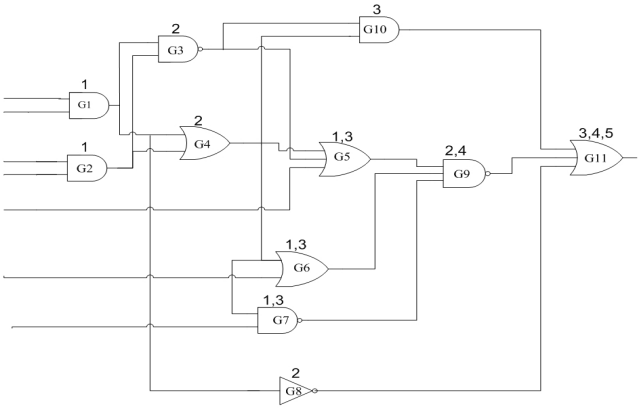

Figure 6. A Simple Gate Network.

$\mathcal{W}_{min}$ windows reach another threshold θ_{min}. If this happens, then we construct a big switching window, which has a starting time at θ_{min} and ends at θ_{max}.

We can tradeoff accuracy for execution speed by taking different values of θ_{min} and θ_{max}.

3.4 Example

Let us consider a simple example of a gate network as shown in Figure 6. For simplicity, we assume that each gate has a switching time of 1 unit and it discharges 1 unit of current in that time interval. As outlined in the algorithm, we first construct a current plot from *MSW* of all the gates in the network, shown in Figure 7. Values above each gate indicate its corresponding switching window.

We see that during iteration 1, the time at which maximum number of gates are switching is at time period 2. The gates which contribute to this current are G3, G4, G5, G6, G7, G8 and G9. We can see that G5, G6 and G7 have switching times on either side of the common overlap time, which is the time period 2. In our *MSW* extraction, we

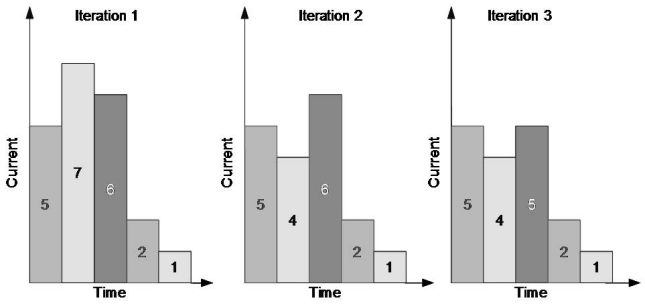

Figure 7. Current Graphs Showing 3 Iterations of the Algorithm.

extract the earliest and the latest switching windows only. Thus, the probability of G5, G6 and G7 having an empty space or virtual switching window in the time interval 2 is very high as compared to G9. G9 has a very high probability of having a void space in 3 but not in 2 since 2 is a switching time interval already extracted and G9 will definitely switch in this time period.

So after gate ordering, we have ordered gates as G5, G6, G7, G9, G3, G4 and G8. Now sending only G5, G6, G7 for FSW extraction and updating the current plot, we see that the current peak moves to time interval 3 which is equal to 6 units of current. Now at time 3, we apply the gate ordering and we obtain G9, G10 and G11. Since G5, G6 and G7 already have their FSW extracted and it is certain that G5, G6, G7, G10 and G11 will switch during the time period 3, we send only G9 for FSW extraction. After FSW extraction of G9 and updating the current plot, we have the maximum current at time periods 1 and 3 equal to 5 current units. The iteration stops here since at both times 1 and 3 there are no gates which have a virtual switching window in them. Since we use gate ordering, we save a lot of time by not extracting the FSW for all the gates that contribute to the maximum current value at a particular time period.

4 Experimental Results

4.1 Experimental Flow

We have used a set of benchmark circuits taken from the ISCAS and MCNC benchmark suites for testing our methodology. Each benchmark was synthesized using a reduced set of gates taken from a 65nm technology library from STMicroelectronics and using Synopsys DesignVision. Using this reduced set of library gates makes the characterization process easier, without loss of generality. For these reduced set of library gates, we perform SPICE simulation to extract peak discharge current. In the experiments we have used a FO4 as the output capacitive load and characterize the gates for a input transition time of 300ps.

Extraction of switching windows for a given gate was done by using static timing analysis; in our experiments we use PrimeTime from Synopsys, which can report all possible data arrival times at the inputs of a specified gate. So the data arrival time at the input constitutes the starting point of the switching window. The ending point of the switching window is just the starting point plus the propagation delay through the gate. Here we try to emphasize the importance of using a static timing analysis, which allows to compute the switching window very quickly, at the expense of some conservative approximation on delay computations. This gives our algorithm an added value since it can be used on large designs.

4.2 Clustering

Our methodology estimates the peak discharge current given a cluster of gates of a larger design. Even if clustering is not the subject of this work, and our algorithm can be applied to any set of gates, the clusters used in the experiments have been built using a simple clustering technique, in order to make experiments more meaningful. In particular we have used a traditional path-driven algorithm to build clusters. In fact, since gates that lie on a path have mutually exclusive switching times, grouping them to form a cluster minimizes the simultaneous current discharge.

4.3 Experimental Data

Table 1 show the results obtained on a set of ISCAS and MCNC benchmarks of different sizes. For each design, two different cluster sizes are considered. Results report the peak discharge current (Column *Initial Current*) obtained by considering only the superposition of MSW versus the estimate achieved by our algorithm (Column *Final Current*). Our algorithm provides an average 11.73% improvement of the upper bound on average (23% maximum). We clearly notice that the savings are strongly dependent on the benchmarks which is a function of the topology of the circuit.

The efficiency of our algorithm can be measured by comparing its execution time against a solution based on the complete extraction of FSWs for all gates in the cluster. Notice that our method and full FSW extraction will yield the same result, the difference being only the number of FSW extractions performed.

For none of the benchmarks reported in the Table 1 has been possible to complete the full extraction the FSWs (using a 50,000s bound). Conversely, selectively extracting FSWs as done in our approach allows to compute maximum current (the longest execution time is below 2000s).

Circuit	*#Gates*	*Clus*	Current [uA]		*% red*
		Size	*Initial*	*Final*	
c499	426	85	1211.6	1035.4	14.5
		169	2291.1	2082.4	9.1
c1355	942	70	973.3	759.1	22.0
		255	3018.3	2701.5	10.5
c1908	990	50	1120.3	923.5	17.6
		432	9353.9	7300.6	22.0
c2670	1178	67	2354.5	2181.3	7.4
		850	19717.6	15119.9	23.3
c3540	1620	294	8564.6	7761.0	9.4
		79	2131.0	2111.5	0.9
apex6	879	145	4170.8	3609.6	13.5
		454	12392.5	10532.2	15.0
x4	1596	250	6062.6	4996.0	17.6
		501	13232.9	11509.7	13.0
term1	1482	845	14569.4	12464.3	14.4
		1026	16057.7	13914.8	13.3
frg2	4323	152	4610.2	4267.5	7.4
		398	12111.1	11661.8	3.7
Average					11.73

Table 1. Maximum Current Estimation Results.

To realize a direct comparison, we have designed a small 4-bit array multiplier, for which full FSW extraction is feasible. Results are in this case less evident than for larger benchmarks (Table 2), because of the relatively small number of resulting MSWs, as well as the cost of building the current plot, which is comparable to a single FSW extraction.

Cluster Size (#Gates)	*Our Algo. (s)*	*Complete FSW (s)*	SpeedUp
50	297	347	1.17X
86	496	589	1.19X
104	600	717	1.20X
125	675	820	1.21X
140	813	1337	1.64X

Table 2. Complete FSW Extraction vs. Our Algorithm.

5 Conclusions

We have proposed an effective algorithm for the computation of tight upper bounds of the maximum discharge current of a circuit, with application in the problem of the sizing of sleep transistors in a clustered MTCMOS-based leakage power optimization scenario.

The algorithm iteratively refines current peak estimates by carefully selecting the time intervals in which to extract the switching windows of the gates in a cluster. Our approach provides the same accuracy as that achievable with a full extraction of switching windows for all the gates, yet in much shorter time. Other fine-grain optimizations allow to further speed up the computation allowing faster convergence.

Results show that our solution achieves maximum current upper bounds of about 12% tighter, on average, than a traditional approach based on the plain overlapping of switching windows. The efficiency of our algorithm is also very good since we achieve the same approximation quality provided by a full extraction of all switching windows, yet with much shorter execution times.

References

[1] K. Roy et al. "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," *Proceedings of the IEEE*, Vol. 91, No. 2, pp. 305-327, 2003.

[2] F. Fallah, M. Pedram, "Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits," *IEICE Transactions on Electronics*, Vol. E88-C No 4. pp. 509-519, 2005.

[3] N. E. Evmorfopoulos et al. "A Monte Carlo approach for maximum power estimation based on extreme value theory," *IEEE TCAD*, Vol. 21, No. 4, pp. 415-432, 2002.

[4] J. Kao et al. "MTCMOS hierarchical sizing based on mutual exclusive discharge patterns," *DAC-35*, pp. 495-500, 1998.

[5] M. Anis et al. "Design and optimization of multithreshold CMOS (MTCMOS) circuits," *IEEE TCAD*, Vol. 22, No. 10, pp. 1324-1342, October 2003.

[6] M. Anis et al. "Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique," *DAC-39*, pp. 480-485, 2002.

[7] W. Wang et al. "Fast techniques for standby leakage reduction in MTCMOS circuits," *SOCC-04*, pp. 21-24, 2004.

[8] A. Ramaligam et al. "Sleep transistor sizing using timing criticality and temporal currents," *10th ASPDAC*, pp. 1094-1097, 2005.

[9] C. Long, L. He, "Distributed sleep transistor network for power reduction" *IEEE TVLSI*, Vol. 12, No. 9, pp. 937-946, 2004.

[10] T.W. Chang et al. "Functionality directed clustering for low power MTCMOS design," *10th ASPDAC*, pp. 862-867, 2005.