MTCMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Patterns

James Kao, Siva Narendra, Anantha Chandrakasan
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
{jkao, naren, anantha}@mtl.mit.edu

ABSTRACT

Multi-threshold CMOS is a popular circuit style that will provide high performance and low power operation. Optimally sizing the gating sleep transistor to provide adequate performance is difficult because the overall delay characteristics are strongly dependent on the discharge patterns of internal gates. This paper proposes a methodology for sizing the sleep transistor for a large module based on mutual exclusive discharge patterns of internal blocks. This algorithm can be applied at all levels of a circuit hierarchy, where the internal blocks can represent transistors, cells within an array, or entire modules. This methodology will give an upper bound for the sleep transistor size required to meet any performance constraint.

1. BACKGROUND

Multi-threshold CMOS is an emerging technology that provides high performance and low power operation by utilizing both high and low V_t transistors[1][2][3]. By using low V_t transistors in the signal path, the supply voltage can be lowered (while still maintaining performance) to reduce switching power dissipation. By reducing V_{dd}, the switching power can be reduced quadratically, but as V_t decreases to maintain performance, the subthreshold leakage current will increase exponentially. For ambitious scaling, the increased leakage power can actually dominate the switching power[4]. In many event driven applications, like a processor running an X-server, circuits spend most of their time in an idle state where no computation is being performed. During these "sleep" times, it is very wasteful to have large subthreshold leakage currents. Static power dissipation can be reduced in the sleep mode by using high V_t transistors with very low leakage currents to gate the power supply lines for the entire module.

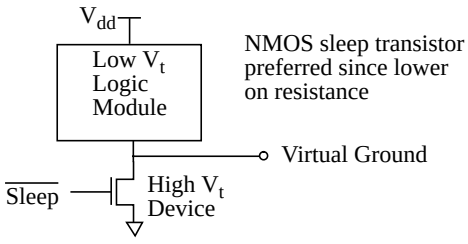


Figure 1. MTCMOS circuit structure.

Although it is easy to reduce leakage by using a high V_t gating device, it is difficult to size the sleep transistor large enough so that performance is maintained. Some initial work on MTCMOS circuits was presented in [5], and it was shown that the sleep transistor can be approximated very closely by a linear resistor that creates a finite voltage drop across the virtual ground node as gates are discharging. This virtual ground bounce causes the internal logic to slow down for two reasons: first, the gate drive is reduced and second, the internal transistor threshold voltages will increase due to the body effect. The worst case delay in an MTCMOS circuit is strongly dependent on the discharge patterns of internal gates, which will cause the virtual ground line to fluctuate depending on discharge patterns through this sleep transistor. The worst case input vector is difficult to predict and can even be different than a vector which exercises a critical path in an ordinary CMOS implementation. As a result, optimal sizing of the sleep transistor for an arbitrary circuit to meet a performance constraint can be difficult. A switch level simulator had been proposed to provide fast MTCMOS simulations to help narrow down this search space[5].

In this paper, we will explore another methodology for sizing the sleep transistor to meet a performance constraint. Rather than search for the worst case input vector to exercise the worst case discharge patterns in the MTCMOS circuit, we instead work from the bottom up, and synthesize a sleep transistor size based on mutual exclusive discharge patterns. Application of this sizing methodology will guarantee that the performance of a complex MTCMOS circuit will be within a chosen percentage of the original CMOS version for all possible inputs.

2. APPROACHES TO TRANSISTOR SIZING

The most straightforward (but difficult) way to correctly size the sleep transistor of an MTCMOS circuit is to exhaustively test for the worst case input vector and to ensure that the worst case delay meets a fixed performance constraint. However, individual gates within this critical path, and other paths within the circuit can degrade in percentage more or less than this fixed criteria. Figure 2b shows how individual gate degradations can vary (assuming both polarities of sleep transistor are used) while overall performance is maintained.

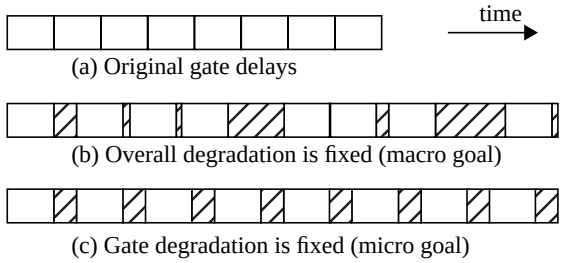


Figure 2. MTCMOS gate degradation scenarios to meet fixed specification.

1-58113-049-x-98/0006/$3.50

A different way to satisfy a macro performance criteria is to ensure that every individual gate meets a local performance constraint (Figure 2c, which assumes both the high to low and low to high transitions are degraded). This will ensure that any combination of gates in a path will also meet the performance requirements. Forcing every single gate to meet a nominal performance measure is a much more demanding constraint than simply achieving an overall performance goal. However, in the context of MTCMOS circuits, it is much easier to implement this sizing strategy because one does not need to determine the worst case input vector pattern for the whole circuit. Instead, each individual gate can be assigned it's own high V_t sleep transistor, whose size will be locally determined through exhaustive SPICE simulations.

Once an MTCMOS circuit is sized with individual sleep transistors then one can systematically merge the sleep transistors together, because they can be shared among mutually exclusive gates, where no two gates can be discharging current at the same time. Finally, these sets of sleep transistors can then be combined to make a single sleep transistor for the whole circuit that guarantees that for any input vector, the MTCMOS circuit performance will be within the specified range of the corresponding CMOS circuit.

3. SLEEP TRANSISTOR SIZING AND MERGING TECHNIQUE

A good way to describe the sleep transistor sizing and merging technique is through an example. Figure 3 to the right shows how an MTCMOS circuit can initially be sized using individual sleep transistors that can be merged together at later steps. The circuit consists of three chains of five low V_t inverters, and measurements are made for the input to output delay, the delay for inverter I_5, and the virtual ground bounce transients. Table 1 below summarizes some simulation results.

R ohms	I_5 (a) %	Total (a) %	I_5 (c) %	Total (c) %	I_5 (e) %	Total (e) %
0	0	0	0	0	0	0
100	1.9	1.2	2.0	1.1	4.9	2.9
200	3.0	2.3	3.5	1.9	9.3	5.7
300	4.2	3.4	4.9	2.9	13.7	8.5
400	5.8	4.6	6.1	3.7	17.9	11.0
500	7.5	5.7	7.5	4.6	22.1	13.4

Table 1. Percent degradation for gate I5 and total delay (for cases a, b, c) as function of sleep resistance.

3.1 Individual sleep transistor sizing

Figure 3a shows the first step in the transistor sizing procedure, where an identical sleep resistor (which models a sleep transistor in the on state) is placed in series with each gate. As can be seen in columns 2 and 3 of Table 1, the overall performance of the inverter chain will be satisfied if the internal gates meet the required speed (i.e. the % delay in column 3 is always less than or equal to that of column 2). The overall delay degradation is less than the individual gate degradation of gate I5 namely because the low to high transitions of inverters I2 and I4 are not degraded by an NMOS sleep transistor. Figure 3b shows how the virtual ground lines (V_1, V_3, and V_5) for this circuit will fluctuate as a result of a rising step function applied to the input.

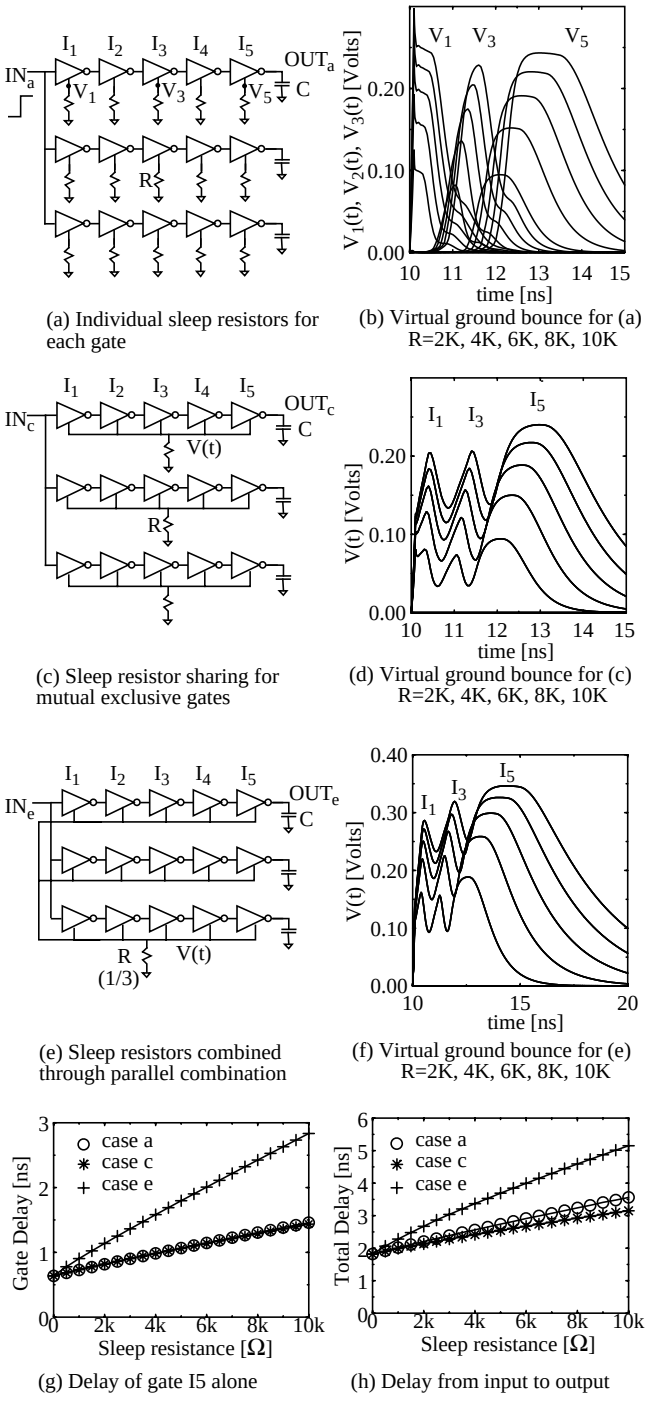


Figure 3. Inverter chain example showing the 3 steps for merging sleep resistors. Circuits use V_{dd}=1.0v, V_t=0.2v, C=50fF, l_{min}=0.7μm.

3.2 Sleep transistor merging based on mutual exclusive discharge patterns

Although it is relatively simple to develop an MTCMOS sizing strategy by individually adding high V_t transistors to each gate in a circuit, this can result in large overestimates in sleep transistor area

and large overheads in wiring area. However, since not all gates in the circuit will switch at the same times, it is possible to merge sleep transistors together from mutual exclusive gates and thereby reduce circuit complexity. For a set of n such gates with equivalent sleep resistances r_1, r_2,... r_n, the sleep resistors can be combined and replaced by a single r_{eff} = min (r_1, r_2,... r_n). These mutually exclusive gates will discharge currents through the sleep transistor at different times so that the virtual ground bounce that each transitioning gate experiences will still be the same or smaller than before. As a result, the delay of each gate sharing the common sleep transistor should also be the same or smaller than in the original circuit. An added benefit of replacing n sleep resistors with a single one is that the subthreshold leakage current will decrease by a factor of n, and also the increased parasitic capacitance on the virtual ground line can improve performance.

Figure 3c shows how the Figure 2a inverter tree's sleep resistors can be replaced by only 3 resistors by utilizing the same high V_t switch for mutual exclusive gates. Inverters I_1, I_2, I_3, I_4, and I_5 for example will never transition from high to low at the same times, and as a result can share a common sleep transistor. Figure 3g shows how the delay of inverter I_5 remains the same for both cases, which is to be expected. Upon closer inspection, one can see that the overall performance of the inverter chain actually improves when the transistors are merged together as in Figure 3h. This is because there is a larger parasitic capacitance on the virtual ground line for the merged case, which will tend to low pass filter the virtual ground bounce. As a result, inverter I_1 will be faster because the virtual ground bounce rises more slowly. As the parasitic capacitances charges up though, later gates will not see these beneficial effects since the capacitance does not have time to discharge again, as can be seen in Figure 3d.

3.3 Merging through parallel combination

Having separate sleep resistors for different groups of mutually exclusive gates can be cumbersome for the circuit layout. In many cases, it is possible to lump these sleep transistors together as a parallel combination, and performance will still be maintained. Although total transistor area will be the same, wiring and layout area can be reduced. To quantify this point, consider the circuit in Figure 4.

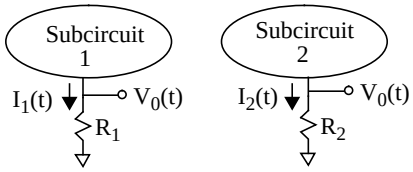


Figure 4. Circuit showing how sleep resistors can be combined in parallel.

If the virtual ground voltages for two different subcircuits is similar, then they can be modeled as two current sources, $i_1(t)$ and $i_2(t)$, connected to resistors r_1 and r_2 to give a voltage waveform $v_0(t)$ for both cases. However, if $i_1(t)$ and $i_2(t)$ are summed together and r_1 and r_2 are placed in parallel, then the new voltage over the resistor is:

$v(t) = (i_1(t) + i_2(t)) * (r_1 // r_2)$
$= (i_1(t) * r_1 * r_2 + i_2(t) * r_1 * r_2) * (1 / r_1 + r_2)$
$= (v_0(t) * r_2 + v_0(t) * r_1) * (1 / r_1 + r_2)$
$= v_0(t)$

which is the same as before. Thus, for two subcircuits with very similar virtual ground transient behaviors, combining the two systems together will result in unchanged virtual ground characteristics, so the overall performance should be unchanged. In general, if voltages $v_1(t)$ and $v_2(t)$ are very different, then the resistors should be combined such that v(t) will not exceed the minimum of $v_1(t)$ or $v_2(t)$. In this case, r_{eq} = min ($v_1(t)$,$v_2(t)$) / ($v_1(t)/r_1 + v_2(t)/ r_2$).

In Figure 3e, the three separate sleep resistors from Figure 3c can be replaced by a single resistor with three times the conductance that now gates the entire circuit. Figures 3g and 3h show comparisons of the delay vs. sleep resistor size for these two cases, and that the resistance must be lowered by one third in order to achieve the same performance. Another way to appreciate this relationship is to examine the virtual ground transient response shown in Figure 3d and 3f. By scaling the resistance by 1/3 for the case with a single global sleep transistor, the virtual ground bounce shown in Figure 3f can be matched to the that of Figure 3g, which would give the same delay behavior.

In general, combining separate sleep transistors into a single common one will be beneficial. The increased parasitic capacitances will tend to speed up the circuit during the capacitor charging stage. More important, the worst case scenario where the subcircuits will all discharge simultaneously is not common. Because the larger resistances used in the original subcircuits are replaced by a smaller resistance applied to the combined circuit, in many cases individual gates will be faster than before. In some degenerate examples using pure parallel combination, it may be possible that two subcircuits with separate sleep transistors might have very different virtual ground transient responses. In such a case, combining sleep transistors by a simple parallel combination will speed up one case, but could possibly slow down the other (the one with a much smaller virtual ground bounce). However, this is most likely not going to affect the overall performance of the circuit as a whole.

3.4 Comparison with optimal sleep transistor size

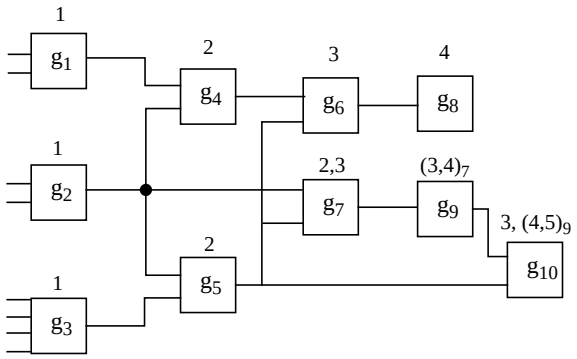
As a concrete example, we simulated the MTCMOS inverter network where the sleep transistor was designed to provide only a 5% degradation in performance over a conventional CMOS implementation. By simulating a single inverter with a sleep resistor in SPICE, we discovered that a sleep transistor with an equivalent resistance of less than 340Ω was required for less than 5% individual degradation. When applied to the inverter chain network and merged together, the sleep transistor equivalent resistance was 113Ω, with a 3.13% degradation in delay. The predicted sleep transistor required was actually an overestimate, because direct simulation shows that one only needs a resistance of 180Ω in order to achieve a 5% degradation in performance. By using this transistor sizing methodology, the transistor width was overestimated by 60%. One major cause for this discrepancy is that in MTCMOS circuits with NMOS sleep transistors, typically only half the gates, those switching from high to low, are actually degraded. Thus even if the high to low transition degrades by 5%, the overall chain will degrade on average by only 2.5% if pulldown and pull up transitions are balanced. Although this inverter chain circuit is easy enough to size through brute simulation, the resistor synthesis approach can be applied to more complicated circuits where exhaustive simulation is not possible.

4. SLEEP TRANSISTOR ALGORITHM

The previous example demonstrated how MTCMOS sleep transistors can be sized individually for each gate and then shared among mutually exclusive gates, where no two gates can be discharging

current at the same time. The primary value of this technique is in the sleep transistor reduction step, because area of the sleep transistor is of primary concern in MTCMOS circuits. One approach to develop a mutual exclusive set of gates in a circuit, is to use a criteria based on the structural interconnections in the network graph. Assuming a unit delay model for each gate, then one can tabulate all the possible times that any particular gate can switch. Mutually exclusive gates can then be grouped together whenever there is no intersection between the corresponding sets of times. In order to minimize total sleep transistor sizes, the number of these groupings of mutually exclusive gates should be minimized, and the sleep transistors chosen to be the largest transistors in each respective group.

Figure 5 shows a random logic circuit with arbitrary gate interconnections, where it is assumed that each gate has a corresponding sleep transistor (modeled as a resistor). Each gate is annotated using a unit delay model with all possible time slots that a transition can occur. Gates that do not have a time period in common will thus be mutually exclusive, and can be grouped together with a common sleep transistor. In cases where a gate can switch at multiple times, we further annotate the set of transition times by a subscript indicating the reference gate, because these two gates are also mutually exclusive even though they share a time slot. For example, gates g_7 and g_9 both show possible transitions at time 3, but this will never happen simultaneously because g_9 is always one time unit behind g_7. Ideally, the groupings should be selected to minimize the overall sleep transistor widths such that gates with very large sleep transistors should be lumped together.



Grouping #1 = $\{g_1, g_4, g_6, g_8\}$ $R_a = \min(r_1, r_4, r_6, r_8)$
Grouping #2 = $\{g_2, g_7, g_9\}$ $R_b = \min(r_2, r_7, r_9)$
Grouping #3 = $\{g_3, g_5, g_{10}\}$ $R_c = \min(r_3, r_5, r_{10})$

$$R_{equivalent} = R_a // R_b // R_c$$

Figure 5. Logic gates annotated with all possible transition times, so that sleep resistors can be merged.

This merging technique based on mutual exclusive gate discharge patterns is most effective for balanced circuits with minimal glitching. Fortunately, a large class of circuits fall into this category, especially since less glitching is attractive from a low power point of view [6]. For circuits with more complicated interconnections and glitching, the merging technique can still be used, although the compression ratio would probably be lower. To further improve the sleep transistor reduction, we can also use more rigorous criteria to determine mutual exclusivity that is based on logic rather than the structural connections in a circuit. We are currently working on such an approach that utilizes boolean manipulation[7].

4.1 Hierarchical Transistor Sizing Methodology

Although the MTCMOS transistor sizing algorithm has been presented at the gate level, in fact it can be applied at many hierarchical levels of a circuit. The algorithm simply operates on generic circuit blocks that are elements within a larger module, and each block is assumed to have a local high V_t sleep transistor that is used for gating the power supply rails. The algorithm is applied to the network by combining the sleep transistors for mutual exclusive blocks. Thus, the blocks that the algorithm operates on can represent individual gates, cells within an array (like an adder cell in a multiplier), or even a module within a chip (like an ALU). In all these cases, a gating sleep transistor can be shared among several different blocks if those blocks have activity patterns that do not overlap in time.

In order to achieve the best results, one should initially use a detailed simulator like SPICE to simulate as large a block as possible and to exhaustively determine the optimal sleep transistor size. Next, the hierarchical merging technique can then be applied to these existing blocks to synthesize an overall sleep transistor for a larger module, where determining a worst case input vector would have been exceedingly difficult. Applying this hierarchical methodology too early can result in unnecessary overestimates for sleep transistor sizes however.

Using the hierarchical sizing methodology again, it is also possible to further apply this transistor merging technique on these existing modules into a larger system. However, by applying this nested algorithm at several levels of abstraction, we will tend to overestimate the minimum sleep transistor size required again, mainly because the granularity of our interactions between blocks will be much larger. For example, applying the algorithm at the cell level within an array might give a larger estimate for the sleep transistor size than if the algorithm had been pushed down in the hierarchy and applied to the gates directly. However, utilizing a hierarchical approach to sizing the sleep transistors is very attractive because detailed circuit complexity can be abstracted away at the expense of accuracy, a tradeoff which is very often desirable.

5. PARITY CHECKER EXAMPLE

As a practical example, the hierarchical sizing methodology was applied to a 32 bit MTCMOS parity checker circuit. The circuit consists of 31 XOR gates which are connected as a tree with 5 levels. Figure 7 shows a smaller 8 bit version of this circuit.

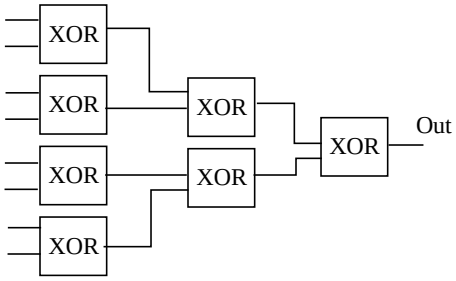


Figure 6. 8 bit parity checker.

The XOR gate was simulated by itself to determine the local sleep resistance needed for a single gate to meet performance requirements. For 20% degradation, the sleep transistor needs a resistance less than 4800Ω, and for 10% degradation the resistance must be less than 2400Ω. With an application of the merging algorithm based on mutual exclusive discharging gates, the total number of sleep transistors required could be reduced from 31 (one for each

gate) to only 16. The resulting sleep transistor for the entire 32 bit parity checker was then calculated to be less than 300Ω for 20% degradation and less than 150Ω for 10% degradation.

Since there are too many vector pairs (2^{64}) to test exhaustively, Table 2 below shows simulation results for a subset of 5 input vectors. Each of these vectors was chosen to exercise a critical path through the top row of the parity checker. Furthermore the critical 2-input XOR gates each transition with the worst case inputs (x=0->1,y=0->0).

Input Vector	CMOS [ns]	R =150Ω [ns], %degr	R = 300Ω [ns], %degr
1	9.08ns	9.14ns, 0.7%	9.21ns, 1.4%
2	9.07ns	9.34ns, 3.0%	9.60ns, 5.5%
3	9.07ns	9.46ns, 4.3%	9.87ns, 8.8%
4	9.08ns	9.44ns, 4.0%	9.81ns, 8.0%
5	9.08ns	9.34ns, 2.9%	9.60ns, 5.7%

Table 2. Parity generator performance as function of sleep transistor width for different input vectors.

The SPICE simulation shows how the sleep transistor sizes (150Ω and 300Ω) ensure performance within 10% (9.99ns) and 20% (10.90ns) of the CMOS critical delay of 9.08ns. Vector #1 does not cause large currents to flow in adjacent gates, so its degradation in performance is not large (0.7% and 1.4%). However, vector #3 creates significant currents through adjacent gates, and as a result is more susceptible to degradation (4.3% and 8.8%). In all cases however, the delays are significantly faster than predicted. Although there are other vector combinations that will result in larger delays, typically the sleep transistor sizing from the algorithm will still be a conservative overestimate of the required sleep transistor size. This is due mainly to three factors. First, only one half of the gates, those switching from high to low, are actually degraded as described in section 3.4. As a result, ensuring all NMOS transistors degrade by no more than 20%, will likely cause only a 10% degradation in overall performance. Second, our gate partitioning can be further improved by using more sophisticated algorithms to determine mutual exclusivity, as only a structural logic independent grouping algorithm was used. Finally the requirement that each gate's high to low transition degrade by no more than a fixed amount is overly stringent, and also contributes to a conservative estimate of sleep transistor size.

6. WALLACE TREE MULTIPLIER EXAMPLE

As another example, the hierarchical sizing methodology was applied to a 6x6 bit Wallace tree multiplier circuit shown in Figure 7. This is a type of circuit is well suited for this algorithm because there are many mutually exclusive gates that cannot transition at the same time[8].

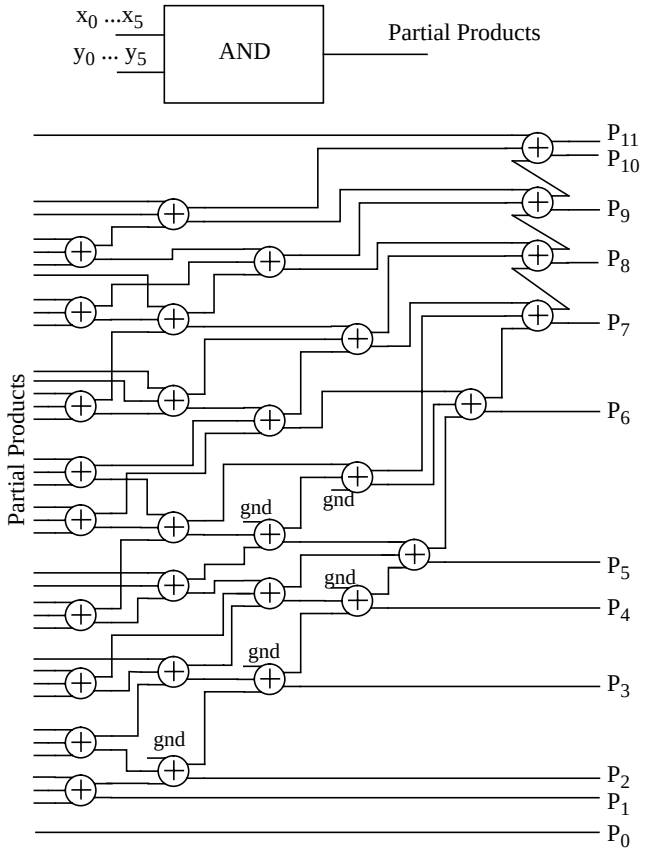


Figure 7. 6x6 Wallace Multiplier.

Initially, the AND gates and the carry save adder units (with representative loadings) were simulated in SPICE to determine optimal high V_t sleep transistor sizes (actually equivalent resistances) for each unit to give rise to a fixed degradation in performance. To achieve a degradation of 20% and 10%, the CSA required sleep transistors with equivalent resistances of 1600Ω and 800Ω, respectively. Likewise 20% and 10% degradation of the AND gates requires equivalent resistances of 2700Ω and 1350Ω, respectively.

Next, the sleep transistor reduction and merging steps were performed to give rise to an equivalent resistor that could gate the entire multiplier. By tabulating all possible time periods that each cell can transition as described in Figure 5, we were able to reduce the 36 AND cell and 30 adder cell sleep resistors into 21 AND cell and 15 adder cell sleep resistors. The total equivalent resistance for the multiplier could then be written as $(R_{add}/15)$ // $(R_{and}/21)$, corresponding to 30Ω and 60Ω for 10% and 20% maximum degradation. The merged resistance is a factor of two greater than the case where no merging takes place, which would correspond to a factor of two decrease in sleep transistor sizing. The branches of this wallace tree structure were not completely balanced because adder cells at inner levels of the tree could actually receive inputs from two levels before. As a result, this implementation has fewer mutual exclusive gates because a fair amount of glitching can occur. Another implementation that balances the paths more carefully would result in larger compression results from the merging algorithm.

For a 6x6 bit multiplier, there are 2^{24} possible input vector pairs, so again it was not possible to exhaustively verify the circuit. However 6 representative vectors were simulated for output nodes P6 and P1 as shown in Table 3.

Input Vector	CMOS [ns]	R =30Ω [ns], %degr	R = 60Ω [ns], %degr
1	8.79ns	9.01ns, 2.6%	9.26ns, 5.4%
2	8.46ns	8.87ns, 4.9%	9.16ns, 8.3%
3	8.72ns	8.92ns, 2.2%	9.111ns, 4.4%
4	11.17ns	11.28ns, 1.0%	11.40ns, 2.1%
5	12.31ns	12.43ns, 1.0%	12.76ns, 3.7%
6	6.55ns	6.79ns, 3.6%	7.07ns, 8.0%

(a) P_6 Delay

Input Vector	CMOS [ns]	R =30Ω [ns], %degr	R = 60Ω [ns], %degr
1	3.19ns	3.24ns, 1.9%	3.40ns, 6.8%
2	3.19ns	3.25ns, 2.1%	3.41ns, 6.9%
3	2.98ns	3.09ns, 3.7%	3.17ns, 6.8%
4	2.98ns	3.05ns, 2.3%	3.10ns, 3.8%
5	2.98ns	3.12ns, 4.8%	3.21ns, 7.8%
6	3.16ns	3.31ns, 5.1%	3.46ns, 10.1%

(b) P_1 Delay

Table 3. Degradation of delays (P_6 and P_1) in multiplier circuit for various sleep resistances and vectors.

By using the hierarchical sizing algorithm, the degradation in any path within the multiplier should degrade by no more than the nominal amount (10% and 20%). As shown in Table 3, two very different paths, ending at P6 (which can include 6 cells), and P2 (which always includes 2 cells), are ensured to meet these performance requirements. Again, since only an NMOS sleep transistor was used, typically only 1/2 the transitions will actually be affected and total degradations will be limited to near 5% and 10%. Also, as described earlier, the restriction that all paths meet the same performance constraint will yield an overly conservative estimate of sleep transistor sizing. For example, an inherently slow path like P2 should be allowed to degrade more since it will unlikely be the worst case delay. By relaxing the degradation requirements for non critical gates, then the sleep transistor can be reduced in size. Nonetheless, the hierarchical sizing strategy provides at least an upper bound on the size of the sleep transistor needed to ensure performance.

7. CONCLUSION AND FUTURE WORK

A sleep transistor sizing methodology for MTCMOS circuits based on mutual exclusive gate discharge patterns was presented in this paper. This methodology provides an upper bound on the sleep transistor size to guarantee a performance level in an MTCMOS circuit by placing delay constraints on individual blocks. Although this sizing algorithm will give an overestimate compared to the optimal sleep transistor size, it is straightforward and can be applied at many hierarchical levels within a circuit. This algorithm is most useful when applied to a large module in conjunction with a detailed simulator that can provide more accurate sleep transistor sizing information for the module's sub-blocks.

The sleep transistor merging algorithm currently relies on a unit delay model and purely structural analysis to determine mutual exclusivity. By using a more complicated delay model and utilizing logic dependencies, the merging algorithm can be improved and more accurate sleep transistor size requirements can be computed. Work is also being done to reduce the constraint that all individual gates meet a fixed degradation requirement. By allowing more flexibility where the speed degradation of individual gates within the circuit can be greater or less than the nominal, one can more efficiently size the sleep transistor to maintain performance in MTCMOS circuits.

8. ACKNOWLEDGEMENTS

This work was funded by DARPA contract #DABT63-95-C-0088.

9. REFERENCES

[1] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, J. Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," IEEE JSSC, vol. 30, no. 8, pp. 847-854, August 1995.

[2] S. Mutoh, S. Shigematsu, Y. Matsuya, H. Fukada, J. Yamada, "1V Multi-Threshold CMOS DSP with an Efficient Power Management Technique for Mobile Phone Application", IEEE ISSCC, pp. 168-169, 1996.

[3] W. Lee, et al., "A 1V DSP for Wireless Communications," ISSCC, pp. 92-93, Feb., 1997.

[4] A. Chandrakasan, I. Yang, C. Vieri, D. Antoniadis, "Design Considerations and Tools for Low-voltage Digital System Design," 33rd Design Automation Conference, pp. 113-118, June 1996.

[5] J. Kao, A. Chandrakasan, D. Antoniadis, "Transistor Sizing Issues and Tool For Multi-Threshold CMOS Technology," 34th Design Automation Conference, pp. 409-414, June 1997.

[6] T. Sakuta, W. Lee, P. Balsara, "Delay Balanced Multipliers for Low Power/ Low Voltage DSP Core," IEEE Symposium on Low Power Electronics, pp. 36-37, 1995.

[7] S. Devadas, K. Keutzer, J. White, "Estimation of Power Dissipation in CMOS Combinational Circuits Using Boolean Function Manipulation," IEEE JSSC, vol. 11, no. 3, pp. 373-383, March 1992.

[8] N. Weste, K. Eshraghian, "Principles of CMOS VLSI Design," Addison-Wesley, Reading MA., pp. 554-557, 1993.