SYNTHESIS OF LOW POWER CMOS VLSI CIRCUITS USING DUAL SUPPLY VOLTAGES

Vijay Sundararajan *Keshab K. Parhi*

Dept. of ECE, University of Minnesota
Minneapolis, MN 55455
E-mail: vijay@ece.umn.edu, parhi@ece.umn.edu

ABSTRACT

Dynamic power consumed in CMOS gates goes down quadratically with the supply voltage. By maintaining a high supply voltage for gates on the critical path and by using a low supply voltage for gates off the critical path it is possible to dramatically reduce power consumption in CMOS VLSI circuits without performance degradation. Interfacing gates operating under multiple supply voltages, however, requires the use of level converters, which makes the problem modeling difficult. In this paper we develop a formal model and develop an efficient heuristic for addressing the use of two supply voltages for low power CMOS VLSI circuits without performance degradation. Power consumption savings up to 25% over and above the best known existing heuristics are demonstrated for combinational circuits in the ISCAS85 benchmark suite.

1. INTRODUCTION

Supply voltage reduction is one of the most effective techniques in reducing power consumption of CMOS circuits. The majority of power consumed is dynamic power, which is reduced quadratically with the voltage V_{DD}. Reducing V_{DD}, unfortunately, leads to an increase in delay which results in performance degradation of the entire circuit. Recently many papers have been published on techniques to reduce V_{DD} without degrading performance, [1], [2], [3], [4].

Scaling down the threshold voltage, V_{th}, with V_{DD} was employed in [1]. This approach, however, faces the problem that the standby leakage current increases significantly because of low V_{th}. To cut off the leakage current in a sleep mode, several approaches have been proposed such as a multi-threshold-voltage CMOS, [2], or a variable-threshold-voltage scheme, [3]. These require, however, additional process steps, [2], or additional circuits for substrate bias control, [3]. Another approach is to use parallel/pipelined architectures for keeping the required throughput of the circuit even with degraded transistor-performance, [1]. However, parallel architectures lead to a heavy area penalty in the datapath and pipelined architectures incur undesirable latency and a slight area penalty for the pipelining latches. Yet another approach involves using dual supply voltages one high, V_{DD}^H, and the other low, V_{DD}^L, so that the circuit part on the critical path is operated with V_{DD}^H and the circuit part off the critical path is operated with V_{DD}^L, [4]. This results in reducing the power without degrading the entire circuit performance. Advantages of this approach are: 1) no need of changing V_{th} and hence no need for changing the regular fabrication process; 2) no need for creating parallel/pipelined datapaths causing heavy area penalty or undesired increase in latency. The practicality of using dual supply voltages is demonstrated in [4] for the design of a low power media processor fabricated with dual supply voltages. A substantial savings in power consumption is demonstrated for the media processor in [4] at the expense of a slight area penalty.

*This research has been supported by the Army Research Office under grant number DA/DAAG55-98-1-0315.

In the design of high performance circuits reducing the critical path of designs takes up the majority of the design time. This results in excessive slack in various structural paths in the design. Therefore, it is extremely desirable that CAD tools automatically find such slack and exploit it for power reduction. In [4] simple greedy sub-optimal heuristics are proposed for utilizing this available slack using a dual supply voltage scheme for obtaining significant reduction in power consumption.

In this paper we develop a formal model for the use of two or more supply voltages for reducing power consumption in CMOS circuits without degrading performance. An efficient heuristic is then derived from this model. Our technique uses an iterative method based on *linear programming, LP*, and solves the problem in a near optimal manner using reasonable amount of CPU time.

2. PRELIMINARIES AND RELATED WORK

If a gate operating at a lower supply voltage were to directly feed a gate operating with a higher supply voltage then large amount of static current is likely to flow in the PMOS of the gate with the higher supply voltage. This is due to the fact that when the output of the low voltage gate is at "1" its voltage level may not be sufficient to turn-off the PMOS of the succeeding high voltage gate. This problem can be avoided by using the level conversion circuit shown in Fig. 1. While the level-converter eliminates the static power dissipation it dissipates substantial power while toggling. In addition, introducing a level converter in the circuit may lead to performance degradation due to the propagation delay of the level converter which may or may not be substantial. Any formal technique that attempts to formulate the use of multiple supply voltages for circuit design must therefore take the delay of and the power dissipated in the level converters into account.

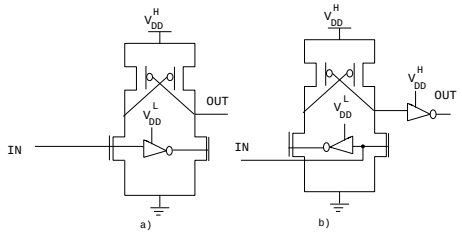

Figure 1. a), b) Level converter circuit for interfacing gates operating at a lower supply voltage of V_{DD}^L to gates at a higher supply voltage of V_{DD}^H. b) can be used to interface V_{DD}^L gates with multiple fanouts with V_{DD}^H gates.

Previous work in addressing this problem has concentrated on solving the problem at function level, [5], [6], [7], [8]. In these, the problem is addressed as one of finding an optimal schedule for a data-flow description of an algorithm. At function level there are two factors that allow for relatively easy problem modeling: 1) The problem sizes are relatively small; so slow techniques like *Integer Linear Programming*, ILP, can be fruitfully employed. 2) The level converter delay is insignificant in comparison to functional unit delays and hence can be ignored. These two factors can be completely unrealistic when we consider gate level circuits. Our technique tackles these difficulties by: 1) Using *linear programming*, LP, techniques which are polynomial time techniques.

2) Explicitly modeling the delay of the level converters and the power consumed by them.

3. NOTATION

A gate-level combinational circuit can be represented by a directed acyclic graph, G = (V, E), referred to as a circuit-graph. Every circuit-graph node $u \in$ V represents a logic gate or a primary input and every directed edge $e_{uv} \in$ E denotes the wire that connects the output of gate u to the input of gate v. The primary input junctions PIs are also modeled as nodes $\in$ V. The gates with a fanout of 0 constitute the primary outputs, PO. The delay of a node/gate u is denoted by $delay(u)$. The maximum propagation time for a signal through the circuit-graph, for any path from a primary input node to a primary output node constitutes the *critical path*, $CP(G)$, of the circuit-graph. We now define three attributes for every node in G. For a node u these are namely, the arrival time $AT(u)$, the required time $RT(u)$ and the slack, $sl(u)$. Additionally, every wire $e_{uv} \in$ E has the attribute *edge-slack*, $esl(e_{uv})$. We will now define all the attributes mentioned previously.

$$\begin{cases} AT(u) = external\ time\ of\ arrival,\ u \in PI, \\ else, \\ AT(u) = max_{v \in fanin(u)}(AT(v) + delay(v)), \end{cases}$$

$$\{\ CP(G) = max_{u \in V}(AT(u) + delay(u)),$$

$$\begin{cases} RT(u) = CP(G) - delay(u),\ u \in PO, \\ else, \\ RT(u) = min_{v \in fanout(u)} RT(v) - delay(u), \end{cases}$$

$$\begin{cases} sl(u) = RT(u) - AT(u), \\ esl(e_{uv}) = RT(v) - AT(u) - delay(u). \end{cases} \quad (1)$$

We call a circuit safe when all nodes $u \in$ V have $sl(u) \geq 0$ and all wires have $esl(e_{uv}) \geq 0$.

3.1. Delay Balancing

A given circuit-graph G can be transformed to a functionally equivalent circuit-graph G' by introducing appropriate number of unit delay buffers into the circuit in such a manner that for every $e_{uv} \in$ E $esl(e_{uv}) = 0$ and $CP(G') = CP(G)$. This process is known as delay balancing. For our purposes we use delay balancing as a tool to capture all the slack in the circuit. The delay buffers we use for delay balancing are *fictitious* entities whose only purpose is to model the slack present in the circuit. We refer to these fictitious buffers as UDFs (Unit Delay Fictitious-Buffers). Fig. 2 shows a gate level circuit and Fig. 3 shows its delay balanced counterpart; the "boxed" numbers in the wires of the circuit in Fig. 3 represent the number of UDFs on that wire.

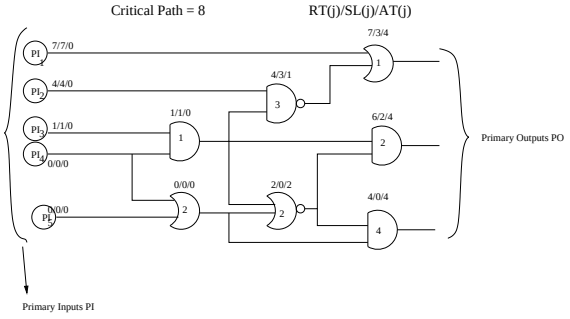

Figure 2. An example of a circuit-graph the integer numbers within each node/gate represent its delay and each gate u has the triplet (RT/SL/AT) above it.

Starting with a given circuit-graph there are several possible ways to produce a delay balanced graph. Any such delay balanced graph will from now on be referred to as a delay balanced configuration.

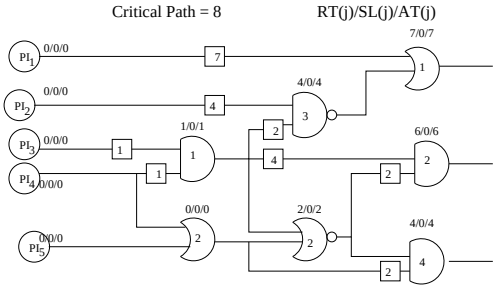

Figure 3. The circuit in Fig. 2 after delay balancing. The boxed integers on wires represent the number of UDFs added to the wires for delay balancing.

3.2. UDF-Displacement

We define *UDF-Displacement*, a circuit-graph transformation technique, as a mapping r:V$\longrightarrow$Z, {Z: the set of integers}; such that the number of UDFs in the wire e_{uv}, $UDF^r(e_{uv})$, after UDF-Displacement is related to the number of UDFs before UDF-Displacement, $UDF(e_{uv})$, by,

$$UDF^r(e_{uv}) = UDF(e_{uv}) + r(v) - r(u). \quad (2)$$

A *UDF-Displacement* is legal if and only if $UDF^r(e_{uv}) \geq 0$ for all wires $e_{uv} \in$ E.

We state the following without proof.

Theorem 1 *All legal delay balanced configurations for a given circuit-graph G are* UDF-Displaced *versions of each other.*

Theorem 2 *The net change in the number of UDFs in any structural path from a node/gate u to another node/gate v after* UDF-Displacement *is always $r(v) - r(u)$.*

The above theorem gives rise to the following corollary.

Corollary 1 *If we connect all the gates $\in$ PO (primary outputs) of a given combinational circuit to a common dummy node O through dummy wires then, if we restrict r(O) to be exactly 0 and also r(I) for every input node I $\in$ PI to be exactly 0, then the critical path of the transformed circuit-graph after UDF-displacement remains unaltered.*

4. PROBLEM FORMULATION

4.1. Dual Supply Voltages for Gate-Level Circuits

We first formulate the problem at gate-level. Furthermore, for simplicity we assume that gate u has delay d_u^H at V_{DD}^H and a delay d_u^L at V_{DD}^L with $d_u^L = d_u^H + k_u$ where k_u is a positive integer. Also, we assume that the delay of a level converter introduced between a gate at voltage V_{DD}^L and a gate at voltage V_{DD}^H is a fixed integer $= d_{lev}$. We call the above model the *integral delay-difference model.*

We later extend the model to cases where k_u, d^{lev}, are non-integral.

4.1.1. The Integral Delay-Difference Model

Beginning with a circuit where all gates are operated with a supply voltage of V_{DD}^H if gate u has a slack of at least k_u time units then it is a potential candidate for being switched to V_{DD}^L. However, only a subset of all the gates with the requisite slack can be switched to V_{DD}^L. Identifying that particular subset of gates which lends us a maximum power benefit is our problem at hand.

The key fact that *UDF-Displacement* can generate all delay balanced configurations of a circuit-graph, see Theorem 1, leads to the following.

Objective: To employ UDF-Displacement to identify that particular delay balanced configuration which identifies all the logic gates which can be switched to a lower supply voltage while providing a maximal reduction in power consumption, while also maintaining the critical path. The process of obtaining a Power Reducing Optimization with UDF-Displacement will here onwards be referred to as PROUD.

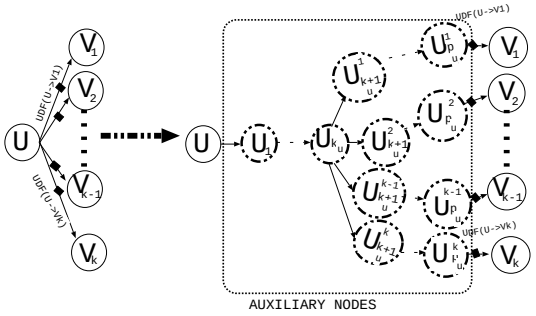

Figure 4. The circuit-graph is transformed under "PROUD" wherein the neighborhood of every node u in the circuit-graph is transformed as shown above.

PROUD will be modeled as a linear programming (LP) problem and can be summarized as,
Input: A combinational gate-level netlist, with all gates using V_{DD}^H.
Output: A power optimized version of the above circuit utilizing the same gates as before but with some gates using V_{DD}^L.
First, transform the circuit-graph by adding a common fanout node O for all nodes $\in PO$ as in corollary 1. For each node/gate, $u \in$ V, define $p_u = k_u + d_{lev} - 1$ and then:
1. Produce a random delay balanced configuration of the given circuit-graph. We use a depth first UDF insertion heuristic for this.
2. Transform the circuit-graph using the transformation illustrated in Fig. 4 as follows: i)introduce p_u auxiliary nodes/gates with a delay of 0 units in between gate u and each of its fanout nodes as shown in Fig. 4, ii)Exactly k_u of these auxiliary nodes are common to all the fanout nodes and are denoted by u_i $i \in \{1 \cdots k_u\}$. The remaining $p_u - k_u$ nodes are unique to each fanout wire and for the fanout wire $u \rightarrow v_j$ these are denoted by u_i^j $i \in \{k_u + 1 \cdots p_u\}$, iii)Every auxiliary arc introduced between dummy nodes has exactly 0 UDFs. Note that the transformed graph is also delay balanced.
3. Associate with each "real" node/gate u the variables, $r(u), low(u), level\_conv(u)$ and with each auxiliary gate u_i, u_i^j the variable $r(u_i)$, $r(u_i^j)$ and set up the following constraints for every typical "real" node/gate, like the gate in Fig. 4:

$$\begin{cases} low(u) & \leq & r(u_1) - r(u), \\ low(u) & \leq & r(u_i) - r(u_{i-1}), i \in \{1 \cdots k_u\} \end{cases} \quad (3)$$

$$\begin{cases} 0 & \leq & r(u_i^j) - r(u_{i-1}^j), \\ & & i \in \{k_u+1 \cdots p_u\}, j \in \{1 \cdots k\} \\ 0 & \leq & r(v_j) - r(u_{p_u}^j) + UDF(e_{uv_j}), \\ & & j \in \{1 \cdots k\}, \\ low(u) - & low(v_j) \leq & r(u_i^j) - r(u_{i-1}^j), \\ & & i \in \{k_u+1 \cdots p_u\}, j \in \{1 \cdots k\} \\ low(u) - & low(v_j) \leq & r(v_j) - r(u_{p_u}^j) + UDF(e_{uv_j}), \\ & & j \in \{1 \cdots k\}, \end{cases} \quad (4)$$

$$\{ \; level\_conv(u) \geq low(u) - low(v_j), j \in \{1 \cdots k\} \quad (5)$$

$$\begin{cases} 0 & \leq & low(u) \leq 1. \\ 0 & \leq & level\_conv(u) \leq 1. \end{cases} \quad (6)$$

4. Set up the linear power benefit function, COST,

$$Minimize\ COST = \sum_{u \in V} \alpha(u) low(u) ({V_{DD}^L}^2 - {V_{DD}^H}^2) C_{gate} f_{clk}$$
$$+ \alpha(u) level\_conv(u) ({V_{DD}^H}^2 C_{inv2} + {V_{DD}^L}^2 C_{inv1}) f_{clk}. \quad (7)$$

5. Solve PROUD using the heuristic PRHEUDENT, Power Reducing Heuristic with UDF Displacement and output the power gain.

The inequalities, (4),(5),(7), ensure non-negativity of number of UDFs on edges of the transformed circuit-graph.

Under PROUD a gate u is switched to V_{DD}^L if and only if $low(u) = 1$, otherwise $low(u) = 0$ and gate u remains with a supply voltage of V_{DD}^H. A level converter is necessary at the output of gate u if and only if $level\_conv(u) = 1$, otherwise $level\_conv(u) = 0$.

The inequalities (4) ensure that for $low(u)$ to be 1 at least k_u UDFs must be available in the path from node u to node u_{k_u}, i.e., at least "1" UDF must be available in each of the wires, $u \rightarrow u_1$, $u_i \rightarrow u_{i+1}$ $i \in \{1 \cdots k_u - 1\}$.

Similarly, the inequalities (5) ensure that if any fanout node v_j of node u remains at V_{DD}^H, i.e., $low(v_j) = 0$, then $low(u)$ can be 1 only if the path from u_{k_u} to v_j has at least d_{lev} additional UDFs.

The inequalities (6) along-with the fact we are minimizing the objective function ensure that $level\_conv(u) = 1$ exactly when $low(u) = 1$ and at least one fanout node v_j of node u has $low(v_j) = 0$.

The objective function in (7) models the global power benefit as a linear function of $low(u)$ and $level\_conv(u)$ for all nodes $u \in V$. The power dissipated in the level converter, see Fig. 1a, is modeled as the power dissipated by two inverters one at V_{DD}^L, $inv1$, and the other at V_{DD}^H, $inv2$.

If we solve the formulated problem as an ILP problem by restricting $low(u)$, $level\_conv(u)$ to integers then we get an optimal solution but the ILP solver may take long CPU time. We therefore use the heuristic PRHEUDENT which will be discussed subsequently.

We now detail the steps of PRHEUDENT.

1. Set up the constraint and cost function using steps 1.-4. in PROUD.
2. Solve for the optimal solution using a standard LP solver.
3. For those nodes having $low(u) = 1$ in the optimal solution in 2. fix the supply voltage at V_{DD}^L, i.e., force $low(u) = 1$. Now,

i)If substantial number of nodes, u, have a non-integral value for $low(u)$ then force $low(u)$ to 0 for these nodes. Now go back to step 1 after updating the value of delay(u) for all nodes, u, which have been switched to V_{DD}^L, and iterate till only a few nodes with non-integral $low(u)$ remain.

ii)If a relatively small number of nodes have non-integral $low(u)$ then use the heuristic in [4] to determine the supply voltage for these nodes. Now terminate PRHEUDENT and output the power gain and the supply voltages used for all the gates in the netlist.

In our simulations we iterated the steps 1., 2., 3. i) only once and then used 3. ii) to determine the supply voltage for all the remaining nodes.

4.1.2. The Non-Integral Delay Difference Model

The obvious way to deal with the case where a gate has a non-integral k_u is to employ the $ceil$ operator to k_u and then solve for PROUD. For example if we have the voltage pair (2.5V, 1.8V) then for a gate ,u, with delay of 1 unit at 2.5V the delay is 1.93 at 1.8V, which means $k_u = 0.93$. By using $k_u = 1$ instead we can still use the above model while maintaining the critical path and obtaining near optimal results. On the other hand for the voltage pair (3.3V, 2.5V) a gate, u, with a delay of 1 unit at 3.3V has a delay of 1.582 units at 2.5V with $k_u = 0.582$ substituting $k_u = 1$ may in all likelihood give us suboptimal results. If, however, we multiply all delay values in the circuit by a factor of 10 then the minimum value of k_u for the above example will be $0.582 \times 10 = 5.82$. Also, the minimum gate delay which was 1 initially will now be 10, we can now use the $ceil$ operator as before to get $k_u = 6$. The excess UDFs used to switch gate u to 2.5V is therefore reduced from $1 - 0.582 = 0.418$ to $(6 - 5.82)/10 = 0.018$. This, however, increases the run-time requirements considerably.

Table 1. Simulation results with, Case I: $V_{DD}^{H} = 5V$, $V_{DD}^{L} = 3V$, unit delay gates with delay(5V) = 1 unit, delay(3V) = 2 units, i.e., k_u = 1 unit for all gates, $u \in$ V, delay(level converter) = 0 unit, i.e., $d_{lev} = 0$, Case II: $V_{DD}^{H} = 5V$, $V_{DD}^{L} = 3V$, unit delay gates with delay(5V) = 1 unit, delay(3V) = 2 units, i.e., k_u = 1 unit for all gates, $u \in$ V, delay(level converter) = 1 unit, i.e., $d_{lev} = 1$, Case III: $V_{DD}^{H} = 5V$, $V_{DD}^{L} = 2.4$, unit delay gates with delay(5V) = 1 unit, delay(2.4V) = 3 units, i.e., k_u = 2 units for all gates, $u \in$ V, delay(level converter) = 1 unit, i.e., $d_{lev} = 1$.

Circuit	# gates	% Power Savings Greedy Approach Case I	% Power Savings Proposed Approach Case I	CPU Time (s)	% Power Savings Greedy Approach Case II	% Power Savings Proposed Approach Case II	CPU Time (s)	% Power Savings Greedy Approach Case III	% Power Savings Proposed Approach Case III	CPU Time (s)
c432	160	9.1%	12.4%	5.4	8.5%	12.1%	7.58	9%	16.2%	7.83
c499	202	11.2%	13.6%	6.58	8.7%	11.7%	9.04	8.7%	19%	10.1
c880	383	48.1%	54.3%	24.28	45.6%	54.1%	38.2	31.7%	58.7%	54.4
c1355	546	12%	16.1%	21.3	10.5%	14.6%	42	5%	18.4%	47.2
c1908	880	40.2%	47.1%	8123.7	36.2%	45.1%	254	31.5%	49.5%	739
c2670	1211	43.4%	56.1%	257.7	39.9%	54.3%	713	34.6%	57.6%	1229
c3540	1705	45.2%	57.3%	579	40.7%	55.4%	1743	32.1%	57.7%	1743
c5315	2351	55.1%	58.5%	681.1	52.7%	57.3%	1719	53.9%	62.4%	4243
c6288	2416	56.3%	62.2%	2139.05	55.6%	61.6%	4627	39.8%	62.7%	7736
c7552	3624	52.1%	56.2%	2459.3	46.2%	54.37%	5235	33.8%	59.6%	9475

5. EXPERIMENTAL RESULTS

We used the PROUD formulation to explore the use of two supply voltages for all the combinational benchmark circuits in the ISCAS85 benchmark suite. The gates in the circuit were assumed to have a capacitance equaling the maximum number of transistors in a worst case charging/discharging path, e.g, a 5-input NAND/NOR gate is assumed to have a capacitance of 5 units. A toggling level converter like the one in Fig. 1a) can be thought of as two toggling inverters, one at V_{DD}^{L} and the other at V_{DD}^{H}. The level converter was modeled as having a capacitance of 2 units with 1 unit of capacitance switching between V_{DD}^{L} and 0 and the other unit of capacitance switching between V_{DD}^{H} and 0, see the objective function $COST$ in (8). The switching activities for all the gates were calculated using a monte-carlo method and a 0 delay model. The clock frequency, f_{clk}, was fixed at the critical path of the benchmark circuit under simulation.

As can be seen in Table. 1 three different cases were simulated which differed in the values of V_{DD}^{L} and/or the delay of the level converter. As can be observed there is substantial increase in the power savings over the greedy technique in [4] when PRHEUDENT is employed. For case i) with the voltage pair (5V, 3V), see Table. 1, there is 2.5%-12.7% increase in power savings with PRHEUDENT. For case ii) with voltage pair (5V, 3V), see Table. 1, the increase in power savings with PRHEUDENT go up marginally for each benchmark circuit and now lie in the range 3%-14.7%. The real benefit of PRHEUDENT is, however, evident with the solo case with the voltage pair (5V, 2.4V), see Table. 1, here the increase in power savings lie in the range 7.1%-25.8%.

The CPU time for solving PRHEUDENT (Ultra-Sparc10) however increases considerably when we handle more complex situations. This is evident by the considerable increase in CPU time from the simplistic model Case I to the more complicated model Case III in Table. 1.

We believe that better CPU times may be possible if we use a better *LP* package. The *LP* solver currently used was lp_solve, available at *ftp://ftp.es.ele.tue.nl/pub/lp_solve*.

In summary we conclude that we have developed a useful formal mathematical model and an effective heuristic for handling dual voltage supplies to reduce power consumption at gate-level. Future research in this area would be directed towards extending the scheme to more than two supply voltages. Also, important is to figure out ways to reduce the execution time of PRHEUDENT.

REFERENCES

[1] A. P. Chandrakasan and R. W. Brodersen, "Low Power CMOS Digital Design," *IEEE Journal of Solid State Circuits*, vol. 27, pp. 473–484, April. 1992.

[2] S. Mutoh *et al.*, "1-V Power Supply High-Speed Digital Circuits Technology with Multithreshold-voltage CMOS," *IEEE Journal of Solid State Circuits*, vol. 30, pp. 847–854, Aug. 1995.

[3] T. Kuroda *et al.*, "A High-Speed Low-Power 0.3 mm CMOS Gate Array with Variable Threshold Voltage (VT) Scheme," in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 53–56, May 1996.

[4] I. Mutsunori *et al.*, "Low Power Design Method Using Multiple Supply Voltages," *Proceedings of ISLPED'97*, pp. 36–41, 1997.

[5] S. Raje and M. Sarrafzadeh, "Variable Voltage Scheduling," *Proceedings of ISLPD'95*, pp. 9–14, 1995.

[6] M. C. Johnson and K. Roy, "Optimal Selection of Supply Voltages and Level Conversions During Data Path Scheduling Under Resource Constraints," *Proceedings of ICCD'96*, pp. 72–77, 1996.

[7] J. Chang and M. Pedram, "Energy Minimization Using Multiple Supply Voltages," *IEEE Transactions on VLSI Systems*, vol. 5, pp. 1–8, December 1997.

[8] W. Shiue and C. Chakrabarthi, "Low Power Scheduling with Resources at Multiple Voltages," in *Proceedings of ISCAS-98*, (Monterey, CA, USA), June 1998.