Variable Voltage Scheduling

Salil Raje and *Majid Sarrafzadeh*
Northwestern University
Electrical and Computer Science
Evanston, IL 60208
USA
{salil, majid}@eecs.nwu.edu

Abstract

This paper presents a low power design technique at the behavioral synthesis stage. Scheduling technique for low power is studied and a theoretical foundation is established. The equation for dynamic power, $P_{dyn} = V_{dd}^2 C_{load} f_{switch}$, is used as a basis. The voltage applied to the functional units is varied, slowing down the functional unit throughput, reducing the power but meeting the throughput constraint for the entire system. The input to our problem is an unscheduled data flow graph *with a timing constraint. The goal is to establish a voltage value at which each of the operations of the* data flow graph *would be performed and thereby fixing the latency for the operation such that the total timing constraint for the system is met. We give an exact algorithm to minimize the system's power. The timing constraint for our system could be any value more than or equal to the critical path. The experimental results for some High-Level Synthesis benchmarks show considerable reduction in the power consumption. For tighter timing constraints, the maximum reduction is about 40% by using supply voltages 5V and 3V and a maximum reduction of about 46% using supply voltages 5V, 3V and 2.4V. For larger timing constraints, the maximum reduction is about 64% by using supply voltages 5V and 3V and a maximum reduction of about 74% using supply voltages 5V, 3V and 2.4V.*

1 Introduction

The advent of portable communication and computing services has stirred a great deal of interest in both commercial and research areas.

The minimization of power consumption in these battery operated circuits has become the crucial issue in the design process. We are of the opinion that the power consumption issue should be addressed both at lower levels (logic synthesis and layout) and at the behavioral level. Lower levels provide new insights while decisions made at the higher levels of the design process are likely to have a greater impact on different aspects of the design quality, power dissipation being one of them. A lot of emphasis has been placed in recent literature on power minimization at the logic synthesis level, e.g. see [4], [5], [6] and at the architecture level [8], but there has been significantly less progress at the *behavioral level.*

The only research that we knows of at the behavioral level [8] relies on algorithmic transformations. The HYPER-LP system minimizes power consumption in application specific datapath intensive CMOS circuits using architectural and computational transformations.

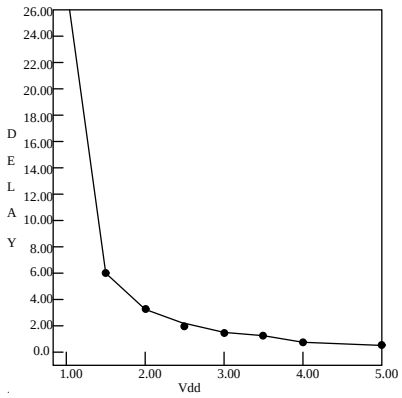

Figure 1: Normalized delay versus Vdd curve, [8]

The effects of reducing the supply voltage on the speed of the functional module are quite well known. The curve in Figure 1 characterises this effect for functional modules. The functional module throughput

decreases as its supply voltage is reduced. The dynamic power consumed by a functional module depends on the square of the supply voltage applied to it, $P_{dyn} = V_{dd}^2 C_{load} f_{switch}$. Thus, reduction of supply voltage reduces the dynamic power consumed by the functional module but also has the undesirable effect of reducing its throughput. Attempts were made at the logic synthesis level and the architectural level to utilize this quadratic dependence on voltage. The idea is to maintain throughput at reduced supply voltages through hardware duplication or pipelining. By using parallel, identical units, the speed requirements on each unit are reduced, allowing for a reduction in voltage [8].

Nielsen et. al. in [1], [2] recently showed the practicality of adaptive scaling of supply voltages to reduce power consumption. Their technique combines self-timed circuitry with a mechanism that adaptively adjusts the supply voltage to the minimum possible, taking into account: process variations, operating conditions, and data dependent computation times. They monitor the activity of the circuit as proportionate to the amount of data in the input buffer and scale the supply voltage accordingly to maintain the throughput of the system a constant.

We present here a supply voltage driven technique to minimize power consumption at the *behavioral level*. We exploit the potential parallelism evident among operations in the *data flow graphs* that are used to define systems at this level of abstraction. Given a timing constraint, there are some operations in the *data flow graph* that could be slowed down by applying a smaller supply voltage and the system throughput would still meet the timing constraint. Assignment of supply voltages to each of the operations and thereby defining their latencies to minimize power consumption while still meeting the timing constraint is the problem we address here. A simple illustration of how voltage scaling at this level could minimize power is seen in Figure 2.

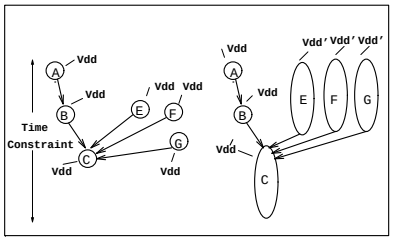

Figure 2: *Data flow graph*, Operation E, F, G, C are slowed down by applying Vdd'.

As can be seen from Figure 2, operation C, E, F and G could be slowed down by applying a smaller voltage Vdd', all the other operations are applied voltage Vdd and hence the total power consumed would reduce from $6Vdd^2 C_{load} f_{switch}$ to $2Vdd^2 C_{load} f_{switch}$ + $4Vdd'^2 C_{load} f_{switch}$. If $Vdd' = Vdd/2$, the power consumed reduces by $(6/(2 + 4/2^2) = 2)$ half. The effect on the finite state machine and other controller effects are in general negligible.

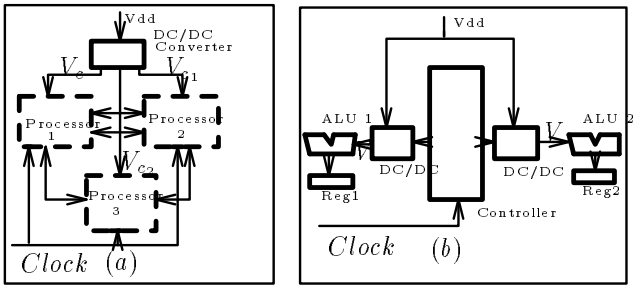

Figure 3: (a) voltage scaling approach at system level, (b) voltage scaling approach used at chip level.

The voltage scaling approach could be used at the system level as well. The nodes in the graph could represent different chips and each of these could be running at different supply voltages to minimize power but at the same time maintaining the system throughput. Figure 3 a shows the voltage scaling approach used at system level and 3 b shows the voltage scaling approach used at the chip level.

Our paper is organized as follows: Section 2 formulates the problems under a particular set of assumptions. Section 3 discusses some of the preliminary notations. Section 4 enumerates the steps in the algorithm without getting into the details of the proof for optimality. Section 5 expounds on the proofs for optimality of the algorithm. Section 6 describes some results and we conclude in Section 7.

2 Problem Formulation

Data flow graph, DFG, represents a partial ordering restrictions of operations. It is represented as a directed acyclic graph $G = (V, E)$. τ is the set of operations and there exists a mapping $Type$ from the nodes to τ, $Type : V \rightarrow \tau$. A directed edge from node v_i to node v_j means execution of v_i must precede that of v_j; this is termed as the precedence constraint. A functional unit, F, is a hardware unit in the functional units library that can perform one or more operations, $F \subset \tau$. The latency of a node, v_i, is the amount of execution time needed for $Type(v_i)$ and is dependent on the supply voltage V_{dd}. The supply voltage versus latency curve is assumed to be the same for all

functional units.

A base voltage V_c is fixed and it is the maximum supply voltage applied to any of the functional units. A base latency of all the operations, t_c, is the execution time for the functional units with a supply voltage V_c. A timing constraint for the system is given in terms of an integral multiple of t_c as in kt_c, where k would be an integer.

A set of allowable supply voltages S is obtained from the supply voltage versus latency curve. If (V_{c_1}, $2t_c$), (V_{c_2}, $3t_c$), $\cdots$, (V_{c_i}, $(i+1)t_c$) are points on the voltage-latency curve then $S = \{V_c, V_{c_1}, V_{c_2}, \cdots, V_{c_i}\}$. The choice of the number of allowable voltages will depend on the technology and the designers preference. The control could also get complicated with a large number of supply voltages and so this number should be kept relatively small. The curve could also dictate this choice as some of the voltages at higher latency points could get very close. See Figure 4.

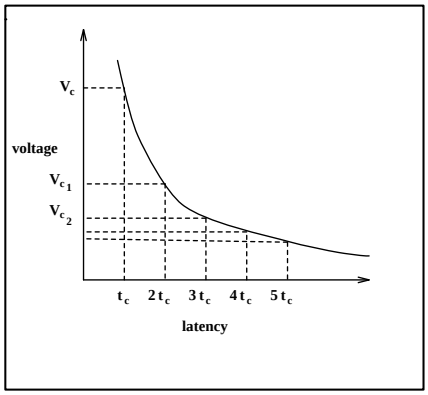

Figure 4: Voltage versus Latency curve and choice of voltages. The range is 1V-5V in Figure 1.

Definition 2.1 *A critical path of a system is defined as the path in the DFG, $\{v_1, v_2, \cdots, v_k\}$, such that the summation of the latencies of the nodes in the path is maximal among all the paths of the DFG. The sum, C_p, is termed as the critical path length.*

The problem now could be defined as: Given a $DFG = G(V, E)$, t_c (or V_c), a timing constraint kt_c, where k is an integer, and the number of allowable supply voltages, obtain S and a mapping $\tau : V \rightarrow S$ to minimize

$$\sum_{v_i \in V} \tau(v_i)^2$$

such that the critical path length of the DFG is less than or equal to kt_c.

3 Preliminaries

For the $DFG = G(V, E)$, we define the set of *fan-in* nodes of a node $v \in V$ as

$$\pi^-(v) = \{v' | (v'v) \in E\}$$

and the set of *fan-out* nodes as

$$\pi^+(v) = \{v' | (v, v') \in E\}.$$

The input node set I is the set of nodes for which $\pi^-(v) = \phi$. The output node set O is the set of nodes for which $\pi^+(v) = \phi$.

Each node, v, is associated with a delay, $d(v)$, equal in value to its latency. The original delay for all nodes is $d(v) = t_c$, this is because the algorithm will proceed with an initial assignment of V_c to all the nodes of the DFG and then decrement the voltage for some nodes obtaining an optimal assignment. At each step of the algorithm, the current voltage assignment to node v is $\tau(v)$.

For the nodes $v \in O$ we can define a required arrival time $t_r(v) = kt_c$ which is the timing constraint for the system. Actual *in-time* for each of the nodes is defined recursively as,

$$t_i(v) = d(v) + max_{z \in \pi^-(v)} t_i(z)$$

Similarly, the actual *out-time* for each of the nodes is defined recursivley as,

$$t_o(v) = d(v) + max_{z \in \pi^+(v)} t_o(z).$$

Therefore, the length of the longest path in DFG passing through the node v is given by,

$$l(v) = t_i(v) + t_o(v) - d(v).$$

An efficient way of implementing this procedure is to perform a *depth first search*. For each $v \in V$ we compute *slack*,

$$s(v) = kt_c - l(v).$$

The DFG is called *safe* if $\forall_{v \in V} s(v) \geq 0$, meaning that the critical path length of the DFG is less than kt_c. An assignment of voltages to nodes such that $\forall_{v \in V} s(v) \geq 0$ is called a *safe assignment*. Also, since for every node v, $\tau(v) \in S$, $l(v)$ is an integral multiple of t_c. $s(v)$ is an integral multiple of t_c as well.

4 Algorithm Overview

We present in this section an overview of the algorithm, A, without going into the details of the proofs for optimality. The following are the steps of the algorithm:

- Step 1: *Initialization*
  Each of the nodes, $v \in V$, in $G(V,E)$ is originally assigned a voltage V_c (also denoted by V_{c_0}), $\tau(v) = V_c$. $d(v)$ value is therefore initialized.

- Step 2: *Computation of slack*
  Using *depth first search* calculate $l(v)$ values for each of the nodes $v \in V$ and hence obtain $s(v) = kt_c - l(v)$.

- Step 3: *Maximal slack value*
  Identify the maximum slack value s^* in the graph G and all the nodes, v, such that $s(v) = s^*$. If the maximal slack value $s^* = 0$ terminate the algorithm; we have obtained an optimal voltage assignment. The set of nodes with maximal slack value s^*, P, induces a subgraph $G_p(P,E')$ in G.

- Step 4: *Dual graph H_p*
  A dual graph $H_p(P,E_p)$ is obtained for the graph $G_p(P,E')$: Obtain a Depth-First Search (DFS) ordering of the graph G_p. Let $D(v)$ be the order in which the node v is visited during the DFS. The dual graph, H_p, is constructed on the node set P. If $u,v \in P$ and there does not exist a path from v to u and from u to v in G_p, then if $D(u) > D(v)$ then $(u,v) \in E_p$.

- Step 5: *Weight assignment*
  If a node $v \in P$ is assigned a voltage $\tau(v) = V_{c_k}$ then assign a weight $W(v) = (V_{c_k}^2 - V_{c_{k+1}}^2)$ in H_p.

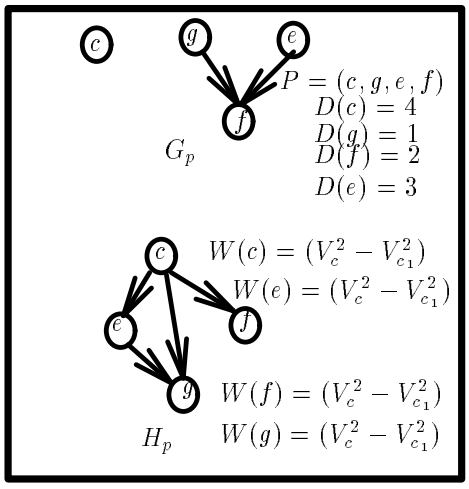

Figure 5: Dual graph H_p of the graph G_p

- Step 6: *Longest weighted path*
  Obtain a longest weighted path in H_p. The longest weighted path in a directed acyclic graph is defined as the path in the graph which has the maximum total node weight and it can be obtained using a single *breadth first search*.

- Step 7: *Reassigning voltages to nodes in the longest path*
  Reassign voltages to nodes in the longest weighted path obtained in the previous step. If a node in the longest path, v, has a prior voltage assignment $\tau(v) = V_{c_k}$ then change this assignment to $\tau(v) = V_{c_{k+1}}$. The new assignment of voltages to nodes in P changes the delay values ($d(v)$ values) for nodes.

The longest path in H_p in Figure 5 is (c,e,g), so $\tau(c) = \tau(e) = \tau(g) = V_{c_1}$.

- Step 8: *loop*
  Go to Step 2.

Theorem 1 *Given a set of allowable voltages S and* data flow graph $G(V,E)$, *the above algorithm produces a mapping $\tau : V \rightarrow S$ that minimizes $\sum_{v_i \in V} \tau(v_i)^2$.*

Theorem 2 *The time complexity of the algorithm is $O(kn^2)$, where kt_c is the timing constraint and n is the number of nodes in G.*

The proofs for the above theorems have been left out of the discussion for brevity.

5 Experimental Results

Experiments were conducted using some of the High-Level Synthesis benchmarks and results have been tabulated in Table 1. Using the normalized delay versus supply voltage curve in Figure 1 obtained by [8] we select three supply voltages mainly 5V, 3V and 2.4V for purposes of our experiments. The increment in the delay in going from 5V to 3V and from 3V to 2.4V are equal; refer to Figure 1 and Figure 4. The functional module delay at a supply voltage of 5V is termed t_c. The timing constraint is kt_c and results were obtained for different values of k for each of the benchmarks. Table 1 shows results obtained for smaller values of k, typically starting from the length of the longest path in te DFG with increments of 1. Table 2 shows results obtained for k equals twice the length of the longest path and thrice the length of the longest path of the DFG. Three sets of results are shown in the tables: power consumed allowing for only 5V to be the supply voltage, power consumed allowing for 5V and 3V to be the supply voltage, and power consumed allowing for 5V, 3V and 2.4V as supply voltages. The power consumed, $P_{dyn} = V_{dd}^2 C_{load} f_{switch}$. We $C_{load} f_{switch}$ is maintained constant and only $\sum V_{dd}^2$ values are shown. The

column x1 shows the percentage reduction in power consumption using 5V and 3V for the benchmark over using a single supply voltage of 5V. The column x2 shows the percentage reduction in power consumption using 5V, 3V and 2.4V for the benchmark over using a single supply voltage of 5V.

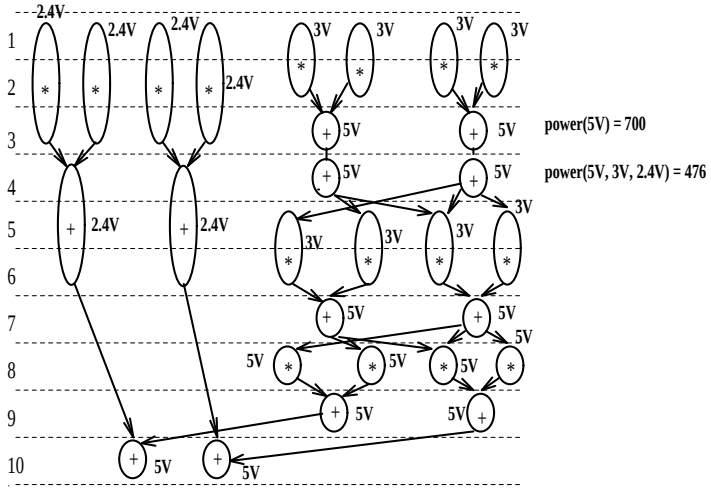

Figure 6: AR-Lattice Filter schedule with $k = 10$, using 5V, 3V, 2.4V

As can be seen from Table 1 there is an average reduction of 8.4 % in power for column x1 and an average reduction of 11.63 % for column x2. Table 2 shows improved reduction in power consumption: The average reduction in column x1 is 16 % and in column x2 it is 23.56 %.

Figure 6 shows the schedule for the AR-Lattice Filter for $k = 10$ and a possible set of voltages: 5V, 3V and 2.4V.

6 Conclusion

We have presented here a novel technique for reducing power at the behavioral level. Research at this level is particularly important since decisions made at this level have greater impact on the design quality. Ease of change in the design structure can also be cited as one of the advantages of going to a higher level of abstraction.

Our algorithm uses variable voltages for scheduling the DFG to minimize power consumption. The speed issue is also considered: Timing constraints are satisfied. Given a particular set of voltages shown in Figure 4, we were able to obtain an optimal schedule for the DFG. The algorithm is time efficient and runs in $O(kn^2)$ time where kt_c is the timing constraint and n is the number of nodes in the DFG. The experimental results for some High-Level Synthesis benchmarks show considerable reduction in the power consumption. For tighter timing constraints the maximum reduction is about 40% by using supply voltages 5V and 3V and a maximum reduction of about 46% using supply voltages 5V, 3V and 2.4V. For larger timing constraints the maximum reduction is about 64% by using supply voltages 5V and 3V and a maximum reduction of about 74% using supply voltages 5V, 3V and 2.4V. Clearly for progressively larger timing constraints and for smaller supply voltages the power consumption reduces giving us a speed versus power consumption tradeoff. Our work lays a strong theoretical foundation for future research in reducing power consumption at the behavioral level.

Future work in this area should involve combining resource allocation and other lower level design decisions along with scheduling and optimize the final design.

References

[1] L. S. Nielsen and J. Sparso, "Low-Power Operation Using Self-Timed Circuits and Adaptive Scaling of the Supply Voltage", *Proc. of the International Workshop on Low Power Design*, April 1994, pp. 99–104.

[2] L. S. Nielsen, C. Niessen, J. Sparso, and K. V. Berkel, "Low-Power Operation Using Self-Timed Circuits and Adaptive Scaling of the Supply Voltage", *IEEE transactions on VLSI Systems*, Vol. 2, No. 4, December 1994, pp. 391–397.

[3] A. Chandrakasan, S. Sheng, and R. Brodersen," Low-power CMOS Digital Design", *IEEE Journal of Solid-state circuit*, April 1992, pp. 473–484.

[4] J. Monteiro, A. Devadas, and A. Ghosh, "Retiming Sequential Circuits for Low Power," *Proc. of the IEEE ICCAD-93*, November 1993, pp. 398–402.

[5] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power", *IEEE transactions on VLSI Systems*, Vol. 2, No. 4, December 1994, pp. 426–436.

[6] K. Roy and S. Prasad, "SYCLOP: Synthesis of CMOS Logic for Low Power Applications", *Proc. of the International Conference on Computer Design: VLSI in Computers and Processors*, October 1992, pp. 464–467.

[7] R. A. Walker and R. Camposano, "A Survey of High-Level Synthesis Systems". Kluwer, MA, 1991.

[8] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing Power Using Transformations", *Transcript of the journal paper (not yet published).*

Bench-mark	Timing Constraint k	Power using 5V	Power using 5V, 3V	x1 % reduc.	Avg. % reduc.	Power using 5V, 3V, 2.4V	x2 % reduc.	Avg. % reduc.
Diffeq	4 †	275	195	29.1	40.73	195	29.1	44.56
	5	275	179	34.91		172.52	37.27	
	6	275	147	46.55		130.8	52.44	
	7	275	131	52.36		111.56	59.43	
FIR	9 †	525	349	33.53	40.38	326.32	37.84	46.4
	10	525	317	39.62		287.84	45.17	
	11	525	301	42.67		265.36	49.46	
	12	525	285	45.71		246.12	53.12	
AR-Lattice Filter	8 †	700	604	13.71	27.43	584.56	16.49	30.20
	9	700	540	22.86		520.56	25.63	
	10	700	476	32.0		456.56	34.77	
	12	700	412	41.14		392.56	43.92	
EWFilter	15 †	850	690	18.82	25.88	677.04	20.35	27.88
	16	850	642	24.47		629.04	25.99	
	17	850	610	28.24		590.56	30.52	
	18	850	578	32.00		555.32	34.67	

Table 1: Power Consumption Results for smaller Timing Constraints
†: Corresponds to the longest path length for the DFG.

Bench-mark	Timing Constraint k	Power using 5V	Power using 5V, 3V	x1 % reduc.	Avg. % reduc.	Power using 5V, 3V, 2.4V	x2 % reduc.	Avg. % reduc.
Diffeq	8	275	99	64	64	82.8	69.89	73.43
	12	275	99	64		63.36	76.96	
FIR	18	525	189	64	64	150.12	71.41	74.19
	27	525	189	64		120.96	76.96	
AR-Lattice Filter	16	700	252	64	64	232.56	66.77	71.87
	24	700	252	64		161.28	76.96	
EWFilter	30	850	306	64	64	280.08	67.05	72.00
	45	850	306	64		195.84	76.96	

Table 2: Power Consumption Results for larger Timing Constraints