Scheduling with Integer Time Budgeting for Low-Power Optimization

Wei Jiang, Zhiru Zhang, Miodrag Potkonjak and Jason Cong
Computer Science Department
University of California, Los Angeles, CA 90095, USA
Email: {wjiang, zhiruz, miodrag, cong}@cs.ucla.edu

ABSTRACT

In this paper we present a mathematical programming formulation of the integer time budgeting problem for directed acyclic graphs. In particular, we formally prove that our constraint matrix has a special property that enables a polynomial-time algorithm to solve the problem optimally with a guaranteed integral solution.

Our theory can be directly applied to solving a scheduling problem in behavioral synthesis with the objective of minimizing the system power consumption. Given a set of scheduling constraints and a collection of convex power-delay tradeoff curves for each type of operation, our scheduler can intelligently schedule the operations to appropriate clock cycles and simultaneously select the module implementations that lead to low-power solutions. Experiments demonstrate that our proposed technique can produce near-optimal results (within 6% of the optimum by the ILP formulation), with 40x+ speedup.

I. INTRODUCTION

Power-efficiency is emerging as a first-order design metric in nanometer-scale IC designs. Innovations on low-power optimizations are required throughout all stages of the VLSI CAD design automation flow. One of the most effective methods for reducing circuit power consumption without significantly degrading performance is to exploit the “slacks” (or timing slacks) available in the system. Slack is defined as the amount of extra delay that each component (either a small gate or a large module) of a design can tolerate without violating the given timing constraint. Since many inter-dependent components of a system or a subsystem may compete for the same timing slack, it is very important to consider and budget the slowdowns of these modules simultaneously to achieve the optimal solution. This is generally referred to as the “time budgeting problem” (also known as delay budgeting).

In this work we focus on power optimization at the high level during the behavioral synthesis step, which automatically transforms untimed or partially timed functional specifications into cycle-accurate RTLs. Behavioral synthesis promises a higher optimization potential and is able to explore a wider range of tradeoffs within a shorter turn-around time than what can be achieved by logic synthesis. As pointed out in [16], using synthesis techniques such as scheduling, resource binding, pipelining, and behavior-level optimization can potentially achieve up to a 70% power reduction, whereas RT-level and logic-level optimization can only achieve, at most, a 40% power saving.

Scheduling, which exploits the parallelism in behavior-level design and determines the time at which different computations and communications are performed, is commonly recognized as one of the most important problems in behavioral synthesis. Not surprisingly, a variety of scheduling techniques have been proposed over the years to reduce power consumption. An optimal scheduling solution is given in [18] for the time-constrained scheduling problem under variable supply voltages. However, the optimality only holds when the voltage curve satisfies the linear differential property, i.e., the difference of the squares of consecutive voltage points on the curve must be a constant. Multi-Vdd scheduling for high throughput, functionally pipelined designs is addressed in [2] based on a dynamic programming approach. An ILP formulation is given in [8] for the scheduling problem under multiple Vdds. A dual-Vth scheduling and module selection problem is proposed in [19]. The authors heuristically solve a maximum weight independent set problem to reduce leakage power under the latency constraint. Their approach considers the operation binding constraints for resource sharing.

In this paper we simultaneously consider the integer time budgeting (ITB) problem and scheduling problem to efficiently explore the large solution space at the high level. Our low-power scheduler (LPS) can efficiently handle various scheduling constraints such as dependency constraints, latency constraint, frequency constraint, and resource sharing constraints, etc. Without resource sharing constraints, the LPS can optimally minimize the total node power consumption by minimizing the separable convex objective function. When the resource sharing constraints are present, we are still able to optimize the total functional unit power in a near-optimal way. Overall, the contributions of our approach include:

(i) We formulate the low-power time budgeting problem as a linearly constrained separable convex optimization problem (to be defined in Section II). In particular, we formally prove that the underlying constraint matrix of the formulated problem is totally unimodular, which automatically guarantees that the optimal integral solutions can be obtained in polynomial time.

(ii) We apply our time budgeting theory to low-power scheduling in behavioral synthesis. Our scheduler generically captures the power-delay tradeoffs of different resources as a set of convex pareto-optimal curves, and intelligently assigns the operations to appropriate clock cycles and selects resource implementations simultaneously through the slack distribution.

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

(iii) We prove that the low-power scheduling problem is NP-Complete with the presence of resource sharing constraints. Using our time budgeting theory, we propose an efficient heurisitc which globally rounds the optimal fractional solution to integral solution in polynomial time.

The remainder of this paper is organized as follows: Section II give the preliminaries and problem formulation of our ITB problem, and presents our new theories that provide foundation of our solution to the ITB problem; Section III applies our time budgeting approach to solve a low-power scheduling problem for behavioral synthesis; Section IV reports experimental results and is followed by conclusions and ongoing work in Section V.

II. Integer Time Budgeting for DAGs

In this section, we propose a mathematical programming based approach on the ITB (integer time budgeting) problem, which can handle convex objective functions. This formulation can be extended to solve the power optimization problem by adding various scheduling constraints.

The problem of distributing the slacks to different modules of a design is generally referred to as the time budgeting problem (also known as delay budgeting). In addition to power minimizations, time budgeting has been extensively studied in many other optimization domains, such as design timing closure (e.g., [9]), timing-driven placement/floorplanning (e.g., [14]), etc. However, most of the early techniques cannot guarantee the optimal integral values of the results. In fact, the requirement for an integral solution of time budgeting is intrinsic to behavioral synthesis where the clock cycle time is fixed for synchronous designs, and each operation has to take an integral number of control steps.

An optimal integral budget assignment algorithm was proposed in [1]. The maximum weighted sum of the delay budgets can be obtained using linear programming followed by a re-budgeting algorithm to optimally convert the solution with fractional values to integers. This work was later enhanced by [5] using a network-flow-based formulation for better efficiency. A recent work [11] explored the delay relaxation problem for design closure. Their approach can handle concave objective functions; the general integer budgeting formulation is transformed into a convex cost integer dual network flow problem which can be solved in polynomial time.

In this section, we propose an alternative approach to the ITB problem with separable convex objective function. We formulate the ITB problem in a mathematical programming framework which allows utilization of the existing state-of-the-art solvers in this field. The application to low-power scheduling also shows that the mathematical programming formulation is flexible and easily extendible to handle various application-specific design constraints.

In the following, we first show that the constraint matrix of our formulated time budgeting problem possesses a special property called total unimodularity. Using this important property, we prove that the problem is optimally solvable in polynomial time when the objective function is separable convex.

A. Problem Formulation

Given a directed acyclic graph $G(V, E)$ and a time constraint T, we describe our generalized time budgeting problem using a mathematical programming formulation:

$$\min f(b_1, b_2, ..., b_{|V|}) \tag{1}$$

$$a_j \geq a_i + b_i \quad \forall e_{ij} \in E \tag{2}$$

$$b_i \geq d_i \quad \forall v_i \in V \tag{3}$$

$$a_i \geq 0 \quad \forall v_i \in PIs \tag{4}$$

$$a_i \leq T \quad \forall v_i \in POs \tag{5}$$

In the DAG, each node $v_i \in V$ is associated with a non-negative constant d_i, which represents the minimum possible time required to propagate all the input signals of node v_i to its outputs. We also assign a delay budget variable b_i to each node v_i. The value of b_i represents the total budgeted delay on node v_i, and it should be larger than or equal to d_i. Primary inputs (PIs) refer to the nodes without predecessors and primary outputs (POs) refer to the nodes without successors. We also define the concept of arrival time a_i to be the maximum total budgeted delay along any path from PIs to node v_i.

In the constraint system, equation (2) captures the dependences between nodes; Equation (3) ensures that the delay budget of each node is valid, i.e., b_i is no less than the minimum delay d_i; Equations (4) and (5) define the timing constraints at PIs and POs.

The objective is to minimize a convex function f on the delay budget variables for all the nodes (or a subset of nodes) in G. Together with the linear constraints, we form a special class of linearly constrained convex programming problems (LCCP). Specialized convex programming solvers, such as [10], are available to efficiently generate the optimal solutions for continuous problems.

Definition 1 *A function $f(x_1, x_2, ...x_n) : R^n \rightarrow R$ is* ***separable convex*** *if and only if $f(x_1, x_2, ...x_n) = \sum_{i=1}^{n} f_i(x_i)$ where each f_i is a univariate convex function.*

In this paper, however, we are more interested in solving the integer version of the problem formulated by (1)–(5). Specifically, assuming that the given minimum delay values d_i's and the timing constraint T are integral, we show in subsequent sections that the optimal integral solution can also be obtained when the objective function f is separable convex.

B. Totally Unimodular Constraint Matrix

Definition 2 (Papadimitriou and Steiglitz [15]) *A matrix A is a* ***totally unimodular matrix (TUM)*** *if every square submatrix of A has a determinant of -1, 0, or 1.*

For an integer linear program whose constraint matrix is totally unimodular, the problem can be solved efficiently since its LP relaxation gives rise to integer basic solutions.

In this subsection we prove that the underlying matrix formed by constraint inequalities (2)–(5) is a TUM. For the sake of convenience, we transform all the inequality constraints into a standard form $Ax \leq s$. A is an $m \times n$ matrix where m is the total number of linear constraints and n is the total number of variables. Since each node $v_i \in V$ is associated with an arrival time variable a_i and a delay budget variable b_i, we have $n = 2|V|$. Specifically, we move all variables to the left-hand side of the inequality constraints. We then unify the variables a_i and b_i into x_j in the following manner:

$$x_j = \begin{cases} a_j, & 1 \le j \le n/2 \\ b_{j-n/2}, & n/2+1 \le j \le n \end{cases} \tag{6}$$

After the transformation of the inequalities, the resulting standard-form matrix A has the following form:

$$x_i + x_{i+n/2} - x_j \le 0 \quad \forall e_{ij} \in E \tag{7}$$

$$-x_{i+n/2} \le -d_i \quad \forall v_i \in V \tag{8}$$

$$-x_i \le 0 \quad \forall v_i \in PIs \tag{9}$$

$$x_i \le T \quad \forall v_i \in POs \tag{10}$$

A necessary and sufficient condition for a matrix $A = (a_{ij})_{m \times n}$ to be totally unimodular is proposed by Ghouila-Houri [6] as follows:

Lemma 1 (Ghouila-Houri [6]) *A* $0, \pm 1$ *matrix* A *is totally unimodular if and only if each subset J of the columns can be partitioned into two classes J_1 and J_2 such that for each row i, we have* $|\sum_{j\in J_1} a_{ij} - \sum_{j\in J_2} a_{ij}| \le 1$.

Using the above lemma, we can formally prove that the constraint matrix of our time budgeting problem is a TUM.

Theorem 1 *The standard-form constraint matrix* A *of the time budgeting problem is totally unimodular.*

Proof: Observe that the corresponding rows of the inequalities (8), (9), and (10) are simple, and each single row in this category has at most one non-zero element. Thus they always satisfy the Ghouila-Houri condition regardless of the selected column subset. Therefore, we only need to consider the rows introduced by the inequality (7). Given any column subset J, we partition J into two classes, J_1 and J_2, which are defined as follows:

$$\begin{aligned} &J_1 = \{ j \in J \mid (j > n/2) \wedge (\exists_i a_{ij} = 1) \wedge (j - n/2 \in J) \} \\ &J_2 = J - J_1 \end{aligned} \tag{11}$$

Suppose that row i is generated by one particular dependencies edge e_{kl}. In this row we have $a_{i,k} = a_{i,k+n/2} = 1$ and $a_{i,l} = -1$, which are the only non-zero elements. Intuitively, $J1$ and $J2$ serve to separate two $+1$ entries $a_{i,k+n/2}$ and $a_{i,k}$ if they coexist in J. In the meantime, one of them will be together with the -1 entry $a_{i,l}$.

To formally verify that the Ghouila-Houri condition holds for the rows of this type, we discuss two separate cases:

1. $k \in J$: According to equation (11), we have column $(k+n/2) \in J_1$ if $(k+n/2) \in J$, and column $l \in J_2$ if $l \in J$. This implies that $\sum_{j\in J_2} a_{ij} = a_{ik} + a_{il} = 0$ if $l \in J_2$, and $\sum_{j\in J_2} a_{ij} = a_{ik} = 1$ if otherwise. In the meantime, $\sum_{j\in J_1} a_{ij}$ is also either 0 or 1, since only column $(k+n/2)$ is eligible for J_1. Thus we safely conclude $|\sum_{j\in J_1} a_{ij} - \sum_{j\in J_2} a_{ij}| \le 1$.
2. $k \notin J$. Based on equation (11), column $(k+n/2)$ must not belong to partition J_1. This implies that $\sum_{j\in J_1} a_{ij} = 0$ since none of the non-zero columns qualify J_1. For partition J_2, $\sum_{j\in J_2} a_{ij}$ can be -1, 0 or 1 depending on whether columns $(k+n/2)$ and/or l are selected in set J or not. Thus $|\sum_{j\in J_2} a_{ij} - \sum_{j\in J_2} a_{ij}| \le 1$ holds.

Therefore, we prove that for any subset J of the columns, we can always derive an appropriate partition on J so that Ghouila-Houri condition holds for all rows in A. According to Lemma 1, A is a totally unimodular matrix. □

To our knowledge, this is the first formal proof on the total unimodularity of the constraint matrix for the time budgeting problem. In [1] the authors mention total unimodularity of the matrix in special cases when the input graph is a directed path. Authors in [5] show that the dual of the edge budgeting problem can be mapped to a min-cost network flow problem, which indirectly suggests that the constraint matrix of the dual problem is totally unimodular.

C. Separable Convex Objective Function

In this subsection we further show that our ITB problem can be solved optimally when the objective function f is separable convex, namely, $f(b_1, b_2, ..., b_{|V|}) = \sum_{v_i \in V} f_i(b_i)$ where each f_i is a univariate convex function.

As pointed out by Hochbaum and Shanthikumar [7], convex separable optimization is not much more difficult than linear optimization. Combining their work and Theorem 1, we immediately arrive at the main theorem of this paper.

Theorem 2 *Integer time budgeting (ITB) problem formulated by (1)–(5) is optimally solvable in polynomial time, if the objective function is a convex separable function.*

In this work we employ a more efficient and scalable approach by Miller and Wolsey [13] to solve the integer separable convex programming problem. Specifically, we replace each convex function f_i in the objective with its piecewise-linear approximation function f_i', and reformulate the original problem as a linear programming problem.

III. Low-Power Scheduling Problem

In this section we apply our proposed ITB algorithm to solve low-power problems in the behavioral synthesis domain. We first introduce the problem formulation of our low-power scheduling problem. We then present our time-budgeting-based scheduling algorithm to reduce the overall power consumption of the resulting system.

As opposed to the previous low-power scheduling methodologies, we are not limiting our optimization scope purely to voltage scaling. For each type of operation, we generically capture the power-delay tradeoffs as a convex curve, on which each point corresponds to one particular pareto-optimal hardware implementation for the corresponding operation. In theory, we can vary the supply voltage, threshold voltage, gate sizing, and even perform the microarchitecture switching (e.g., changing a ripple-carry adder to a carry-lookahead adder) to derive the tradeoff curve.

A. Problem Statement

In this work we focus on the low-power scheduling problem for data-flow-intensive designs such as those in digital signal processing and image processing domains. Power-efficiency is a forefront concern for these types of designs since they are widely used in mobile battery-powered consumer applications. During the process of behavioral synthesis, we typically use a data flow graph to capture the behavior of an input design.

Definition 3 *A* ***data flow graph (DFG)*** *is a directed acyclic graph $G(V_o,\ E_d)$, where the vertices in V_o represent the operation nodes, and the edges in E_d represent the data dependencies between operations. A directed edge $e(v_i,\ v_j)$ denotes that operation v_i produces one of the input operands for operation v_j.*

The problem we are seeking to solve in this section is a latency-constrained low-power scheduling problem (*LPS*), which is formally stated as follows:

Given: (1) A DFG $G(V_o,\ E_d)$; (2) A set of scheduling constraints which may include latency constraint T (in terms of the number of clock cycles on G), cycle time constraint, resource sharing constraints, etc.; (3) A set of power-delay tradeoff curves for each type of operation such as addition, multiplication, etc.

Goal: The scheduler assigns each operation to one or more consecutive time steps within T so that the given scheduling constraints are satisfied and the final total power is minimized.

B. Low-Power Scheduling Algorithm

In this subsection we extend the SDC-based scheduling algorithm [4] with the proposed ITB techniques to solve the latency-constrained low-power scheduling problem. The SDC-based scheduling algorithm forms a system of difference constraints (SDC) to represent a number of common scheduling constraints including dependency constraints, resource constraints, frequency constraints, latency constraints, and relative timing constraints. It is also capable of modeling and optimizing the schedule latency with a linear objective function.

To explicitly optimize the slack distribution and power consumption, we introduce the concept of budgeting variables into the scheduling constraints. Although the new constraint system is no longer a pure SDC, the underlying matrix remains totally unimodular. We also use a set of convex functions to describe the power-delay tradeoff curves for different resources. Without resource sharing constraints, our low-power scheduler (LPS) can optimally minimize the total node power consumption by minimizing the separable convex objective function. When the resource sharing constraints are present, we are still able to optimize the total functional unit power in a near-optimal way.

B.1 Scheduling and Budgeting Variables

To formally capture the schedule of an operation node in the DFG, we introduce the concept of *scheduling variables* and *budgeting variables*, which are defined as follows.

Definition 4 *Given a DFG $G(V_o,\ E_d)$, each node $v_i \in V_o$ is associated with a* ***scheduling variable*** *$sv(v_i)$ and a* ***budgeting variable*** *$bv(v_i)$. The value of a scheduling variable $sv(v_i)$ captures the starting time step of the operation v_i in the final schedule, and the value of a budgeting variable $bv(v_i)$ represents how many clock cycles the operation v_i lasts in the final schedule.*

For the sake of simplicity, we assume that the latency of each operation is at least one clock cycle (i.e., $bv(v_i) \geq 1$). [1]

B.2 Modeling Scheduling Constraints

As discussed in [4], the SDC-based scheduler is able to mathematically model a number of common scheduling constraints as a set of linear constraints such as resource constraints, dependence constraints, overall latency constraint, and relative timing constraints. With the new budgeting variables, we need to adjust the data dependency constraint and latency constraint. We also introduce our method for handling the resource sharing constraints.

- **Data dependency constraint**: Data dependencies are intrinsic scheduling constraints that must be satisfied to preserve the functionality of the input description. To be more concrete, if there is a data edge from node v_i to node v_j, then v_j cannot be scheduled unless v_i has completed its execution.

$$\forall e(v_i,\ v_j) \in E_d:\ sv(v_i) + bv(v_i) - sv(v_j) \leq 0 \qquad (12)$$

- **Latency constraint**: A latency constraint specifies the maximum acceptable latency over the given DFG. Suppose that a latency constraint T is specified, we then generate the following constraint on each operation node.

$$\forall v_i \in V_o : sv(v_i) + bv(v_i) \leq T \qquad (13)$$

- **Resource sharing constraints**: We can also efficiently model the resource sharing constraints described in [19] to consider the resource sharing. Essentially, these binding constraints partitions V_o into a set of operation groups $F = \{f_j | f_j = (v_1^j, v_2^j, ..., v_k^j), j = 1, ...m\}$. Each group f_j represents one functional unit which executes the operations $v_1^j, v_2^j, ..., v_k^j$ in order. To enforce the ordering between the operations bound to the same function unit, we generate the following precedence constraint.

$$\forall v_i^j \in f_j:\ sv(v_i^j) + bv(v_i^j) - sv(v_{i+1}^j) \leq 0 \qquad (14)$$

Note that the final latency of a functional unit is determined by the minimum time budget among all relevant operations bound to this module instance. To capture this relationship, we can create a budgeting variable $bv(f_j)$ for each functional unit f_j and generate the following constraints.

$$\forall v_i^j \in f_j : bv(v_i^j) \geq bv(f_j) \qquad (15)$$

[1] This implies that operation chaining will not be allowed in the final schedule. However, we can easily extend our algorithm to support operation chaining.

B.3 Optimizing Low-Power Objectives

Given the power-delay tradeoff curve for each type of operation, we can model the relation between power and time budget on each node (or each functional unit) with a *power function*.

Definition 5 *A **power function** $pw_{op}(bv(v_i))$ of a node v_i (or $pw_{op}(bv(f_j))$ of a functional unit f_j) is the piecewise-linear approximation of the tradeoff curve for the operation type $op = type(v_i)$ (or $op = type(f_j)$).*

Optimizing total node power: Without the resource sharing constraints, we can optimize the total node power by solving a linearly constrained convex separable optimization problem with the following objective function:

$$\sum_{v_i \in V_o} pw_{type(v_i)}(bv(v_i)) \qquad (16)$$

Similar to the proof for Theorem 1 in Section II, we can show that the constraint matrix formed by (12)–(13) remains totally unimodular and LPS can optimally minimize the object function (16) in polynomial time by linear programming relaxation. Because each SDC constraint can have at most two nonzero elements, SDC constraint must satisfy the Ghouila-Houri condition.

Optimizing total functional unit power: With the resource sharing constraints specified by F, it is natural to optimize the total power consumption on the functional units with the following objective.

$$\sum_{f_j \in F} |f_j| * pw_{type(f_j)}(bv(f_j)) \qquad (17)$$

Unfortunately, we can show that the constraint matrix is no longer a TUM with constraints in (15). We can further prove that the scheduling problem formulated by constraints (12)–(15) and the objective (17) is NP-Complete by a polynomial reduction from the 3-SAT problem. Due to page limitation, we omit the proof here.

Therefore, we propose a two-step heuristic to solve this problem. At the first step, we solve the continuous version of the original problem to get the "optimal" budget $fb(v_i)$ for each node v_i. Although the values of $fb(v_i)$ might be fractional, they provide very good hints for achieving optimal integral solution. At the second step, we first remove the constraints in (15) to recover the total unimodularity. We then perform a global rounding on the node budgets by minimizing a least-squares objective.

$$\sum_{v_i \in V_o} (bv(v_i) - fb(v_i))^2 \qquad (18)$$

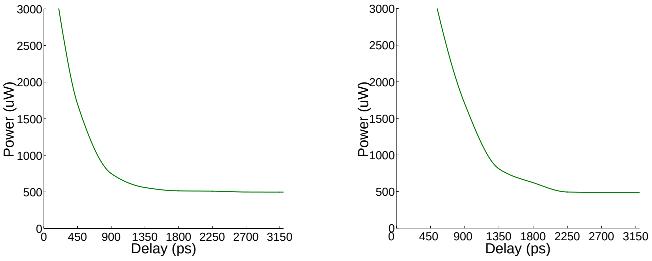
Since the above objective function is separable convex, we can optimally round the node budgets to the desired integral values with minimum least-squares error.

IV. Experimental Results

A. Experiment Setup

We test our low-power optimization algorithm on a set of real-life DSP programs: PR, LEE , ARAI, DIF, DIT, and MCM from [17]. These benchmarks mainly consist of additions, subtractions, and multiplications. In this experiment we assume a uniform 16-bit precision for the operands and operators.

We measure the power-delay tradeoffs for each type of operation using Magma's Blast Create [12] on a TSMC 90-*nm* standard cell library *tcbn90ghp*. Our goal is to minimize the average power of all function units under a latency constraint T (which actually minimizes the total energy consumption as well). We vary the multi-cycle constraints t on the subject operator to guide the Magma tools to perform appropriate gate sizing, and calculate real average power in time steps T by calculating $average\_power(t) * t/T$ (note that the value of T is not important to the optimization process, we can choose T=1 for any design). The curves for a 16-bit adder and a 16-bit multiplier are shown in Figure 1. Theoretically, we can also perform threshold voltage scaling and supply voltage scaling to obtain the curves of a wider range. Nevertheless, this would not compromise the efficacy of our approach provided we choose the power-delay points so that the consecutiveness and convexity of the tradeoff curve are maintained.



(a) (b)

Fig. 1. Power-delay tradeoff curves for (a) a 16-bit adder and (b) a 16-bit multiplier.

We implement the time-budgeting-based low-power scheduling algorithm in our C-to-RTL behavior synthesis system.We take behavioral C as input and output RT-level VHDL along with multi-cycle constraints to specify the actual needed cycle count between the input registers and output registers of the functional units that implement multi-cycle operations. Typical runtimes of our scheduler on the benchmark designs are within one second on a 2.4*GHz* Pentium 4 Linux PC. We use the Magma Blast Create toolset [12] to synthesize the RTL code and analyze the power consumption of the resulting designs.

B. Scheduling Results

As mentioned in Section III, our LPS algorithm can optimally minimize the total node power in polynomial time. With resource sharing constraints presented, our LPS algorithm can also efficiently optimize the total functional unit power with a least-squares objective in (18). In this experiment, we generated the resource constraints based on the technique proposed in [3], which considers the resource availabilities and tries to minimize the switching activity when serializing the operations.

To evaluate the efficacy of LPS, we make comparisons with two alternative solutions: (i) An heuristic approach that honors operation binding constraints but optimize the total node power

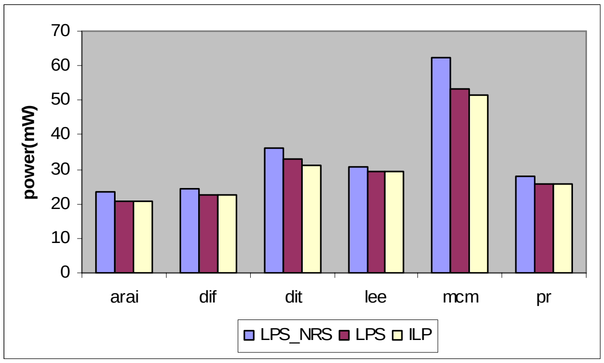


Fig. 2. Power comparisons (estimated by our algorithms) among LPS, LPS-NRS and ILP on all benchmarks.

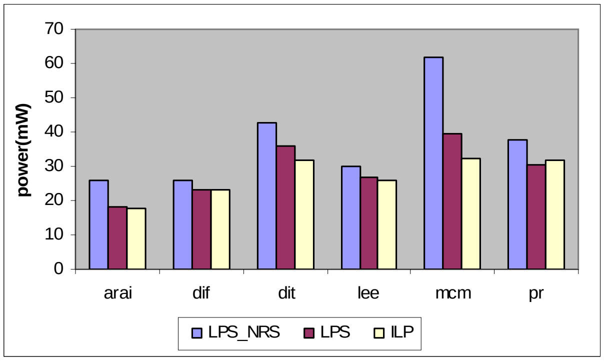


Fig. 3. Power comparisons (reported by Magma tools) among LPS, LPS-NRS and ILP on all benchmarks.

(LPS-NRS); (ii) An exact approach based on integer linear programming (ILP), in which we enforce all budgeting variables to be integral and directly optimize the total functional unit power objective in (17). This approach is exact but has an exponential time complexity.

Figure 2 shows the comparison of the estimated power among these three algorithms with a latency constraint of 1.2x the critical path length. The estimated power consumption by LPS is within 2% of the optimum. The final power data obtained by Magma tools demonstrate high correlation with the power values estimated by the algorithms as shown in Figure 3. On average, the solutions generated by LPS algorithm consume only about 6% more power than the ILP approach and use about 30% less power than the LPS-NRS algorithm. This shows that the LPS algorithm can effectively consider the binding constraints during the optimization process. Not surprisingly, the ILP-NRS algorithm is much faster than the ILP approach in runtime, especially for larger designs (with an approximate 40X speedup). In addition, our proposed algorithm introduces very marginal area (about 3%) and frequency (about 1%) overhead.

V. Conclusions and Future Work

In this paper we present a mathematical programming approach to the integer time budgeting problem and apply the theory to a low-power scheduling problem. Experiments on real-life designs demonstrate near-optimal results (within 6% of the optimum). Our future work is attempting to handle control flows during scheduling.

VI. Acknowledgment

This work is partially supported by the SRC GRC contract 2006-TJ-1400, and the NSF grant CCF-0530261.

References

[1] E. Bozorgzadeh, S. Ghiasi, and M. Sarrafzadeh. Optimal Integer Delay Budget Assignment on Directed Acyclic Graphs. *IEEE TCAD*, 23(8):1184–1199, August 2004.

[2] J.-M. Chang and M. Pedram. Energy Minimization using Multiple Supply Voltages. *IEEE TVLSI*, 5(4):1–8, 1997.

[3] D. Chen, J. Cong, and J. Xu. Optimal Simultaneous Module and Multi-Voltage Assignment for Low-Power. *ACM TODAES*, 11(2):362–386, 2006.

[4] J. Cong and Z. Zhang. An Efficient and Versatile Scheduling Algorithm Based On SDC Formulation. In *Proc. Design Automation Conf.*, pages 433–438, July 2006.

[5] S. Ghiasi, E. Bozorgzadeh, S. Choudhury, and M. Sarrafzadeh. A Unified Theory of Timing Budget Management. In *Proc. Int. Conf. Computer Aided Design*, pages 653–659, 2004.

[6] A. Ghouila-Houri. Caracterisation des Matrices Totalement Unimodulaires. *C. R. Acad. Sci. Paris*, 254:1192–1194, 1962.

[7] D. Hochbaum and J.Shanthikumar. Convex Separable Optimization Is Not Much Harder than Linear Optimization. *Journal of the ACM*, 37(4):843–862, October 1990.

[8] M. Johnson and K. Roy. Datapath Scheduling with Multiple Supply Voltages and Level Converters. *ACM TODAES*, 2(3):227–248, 1997.

[9] C. Kuo and A. C.-H. Wu. Delay Budgeting for a Timing-Closure-Driven Design Method. In *Proc. Int. Conf. Computer Aided Design*, 2000.

[10] LCCP Solver. http://www.ici.ro/camo/nonlin/copl_lc.htm.

[11] C. Lin, A. Xie, and H. Zhou. Design Closure Driven Delay Relaxation Based on Convex Cost Network Flow. In *Proc. Design, Automation and Test in Europe*, 2007.

[12] Magma Website. http://www.magam-da.com.

[13] A. Miller and L.Wolsey. Tight Formulations for Some Simple Mixed Integer Programs and Convex Objective Integer Programs. *Mathmatical Programming*, 98:73–88, 2003.

[14] R. Nair, C. Berman, P. Hauge, and E. Yoffa. Generation of Performance Constraints for Layout. *IEEE TCAD*, 8(8):860–874, August 1989.

[15] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, 1982.

[16] M. Pedram. Low Power Design Methodologies and Techniques: An Overview. http://atrak.usc.edu/m̃assoud/Papers/LPD-talk.ps, 1999.

[17] M. Potkonjak and A. Chandrakasan. Synthesis and Selection of DCT Algorithms using Behavioral Synthesis-Based Algorithm Space Exploration. In *Proc. Intl. Conf. on Image Processing*, 1995.

[18] S. Raje and M. Sarrafzadeh. Scheduling with Multiple Voltages. *INTEGRATION, the VLSI Journal*, 23:37–59, 1997.

[19] X. Tang, H. Zhou, and P. Banerjee. Leakage Power Optimization with Dual-Vth Library in High-Level Synthesis. In *Proc. Design Automation Conf.*, pages 202–207, June 2005.