ILP and Iterative LP Solutions for Peak and Average Power Optimization in HLS

A. Allam[1] and J. Ramanujam[2]

[1]Electrical Engineering Dept., Assiut University, Egypt
[2]Electrical and Computer Engineering Dept, Louisiana State University, USA
atef @aun.edu.eg, jxr@ece.lsu.edu

Abstract

In this paper, we tackle the problem of peak and average power optimization in high-level synthesis. Because of the quadratic relationship of supply-voltage to the dynamic power consumption, voltage scaling is considered as the most efficient technique for reducing power consumptions in CMOS circuits. We present an MILP formulation for the scheduling problem using multiple supply-voltages in order to optimize peak power as well as average power and energy consumptions. As the design problem becomes large, exact solution takes a tremendous amount of run-time; and to explore the design space in a reasonable amount of time, a high quality heuristic is needed. Thus, we devise a two-phase heuristic to solve the multiple supply-voltages scheduling for peak and average power minimization. In the first phase, a guided LP relaxation is developed. Following the relaxed LP schedule, a power-area-saving procedure is developed. Results for peak and average power of our two-phase heuristic well match those obtained by the optimal solution as has been validated through extensive experiments on several benchmarks.

I. Introduction

High-level synthesis (HLS) is the process of mapping the behavioral specification of the system into register transfer description. The outcome of the high-level synthesis is a structural view of the data path and a logical view of the control unit. High-level synthesis involves three main tasks: scheduling, allocation, and binding. The central task is scheduling, which is the process of determining at which control step(s) each operation in the data-flow graph (DFG) executes. We define *Scheduling for Low Power and Energy (SLoPE)* in high-level synthesis as the process of determining at which control step(s), and at what voltage level each operation in the DFG executes with the goal of minimizing power and energy. Although conventional design metrics such as performance, size and testability are important, the most critical design metric nowadays is power. The demand for long-life batteries within tolerable size and weight and the reliability of integrated circuits are the main factors that dictate power-aware design of embedded systems. Reliability of integrated circuits is tightly related to the peak and average power consumption.

Dynamic power consumption is the power consumption due to charging and discharging in CMOS gate and it is considered to be the significant part in the total power consumption. It is given by the equation (1).

$$P_{dynamic} = \tfrac{1}{2}\, \alpha\, C_L\, V_{dd}^2\, f_{clock}, \qquad (1)$$

where C_L is the load capacitance at the gate output, f_{clock} is the circuit clock frequency, V_{dd} is the supply voltage, and α is the average number of transitions per clock cycle at the gate output, referred to as the *switching activity*.

Power/energy reduction in embedded system can be achieved by carefully designing each of its constituent components targeting low power/energy design.

A. Related Work

In recent years, a lot of research work has been done to solve the multiple supply voltages scheduling (MVS) problem. Some of these research works addressed the MVS problem using heuristics [1, 2, 3, 11, 12], while others addressed it using *integer linear programming* (ILP) [3, 4, 5, 6, 8]. Lin et al. [3] proposed an ILP formulation and a heuristic for solving the scheduling with power-minimization problem using variable supply voltages technique. Their heuristic is a list-based scheduler with $O(n^3 \log n)$ time complexity. Chang and Pedram [2] have presented a dynamic programming technique to solve the problem of multiple supply voltage scheduling. Their technique assigns a voltage level (selected from a given fixed number of voltage levels) to each operation in the DFG to minimize the energy consumption under time constraint. The algorithm is pseudo-polynomial and gives optimal results for trees, but is suboptimal for a general directed acyclic graph. Mohanty and Raganathan [8] introduced an ILP based optimization technique for simultaneous minimization of peak and average power using a multiple supply voltages scheme. They introduced two datapath scheduling schemes, one using multiple supply voltages and dynamic clocking and the other using multiple supply voltages and multicycling. Shiue [9] has presented an ILP model and a modified force-directed scheduling (MFDS) heuristic that minimizes peak power under latency constraint considering multicycling and pipelining but he did not consider multiple supply voltages.

The rest of this paper is organized as follows. Section 2 introduces the MVS problem targeting peak and average power minimization. Section 3 describes the MILP formulation for the optimal solution. In Section 4, we develop our iterative LP procedure for the fast near-optimal solution followed by the power-resources saving algorithm in Section 5. Section 6 shows the results of some benchmarks to illustrate our proposed solutions. Section 7 concludes with a summary.

II. Problem Definition

The input to the problem include a DFG representation of the design problem, *G(V, E)* in which each vertex $v \in V$ represents a computational operation and each edge *(u,v)* means that operation *u* has to finish its execution before operation *v* starts, a set of voltage levels for the operating resources, and a power/delay table that contains the average power consumption and the delay time needed for each resource operating on each voltage level and the time constraint, λ. The task at hand is to get a schedule (in which each operation is stamped to a control step, cstep ∈ (1, 2, … , λ) and a voltage level from the set of input voltage levels) that minimizes the peak power consumption as well as the average power and energy consumption according to one of the set of constraints such as time constrained scheduling (TCS), and time and resource constrained scheduling (TRCS).

We propose two solutions for the multiple supply-voltages scheduling (MVS) problem targeting peak power consumption as well as other design factors such as average power and energy consumption, and area. The first is an exact solution based on a mixed integer linear programming (MILP) formulation, while the second solution is a two-phase heuristic that first obtains a guided iterative relaxed LP solution of the MILP formulation followed by a *power-resources saving* procedure, which is a revisit of the output schedule from the first phase in which it tries to minimize the power and/or the operating resources more through scheduling operations in a lower voltage level if possible and/or through moving the operations within their new time-frames if possible without violating the peak power obtained from the first phase.

Our proposed iterative LP solution has not been presented elsewhere to the best of our knowledge. In addition, our presented MILP formulation differs from the one presented in [9] that deals with a single voltage level, while ours considers multiple voltages. In addition, the variables used in the MILP formulation by [8] are 4-dimensional variables, while ours are 3-dimensional, which has a big impact on decreasing the solution run-time.

A. Notations

In the sequel, the following notations will be used.

$p(i, v)$	power consumed by operation *i* using voltage level *v*.
$d(i, v)$	delay (in # of control steps) of operation *i* using voltage level *v*.
P_j	power consumed by all functional units at step *j*.
P_{peak}	maximum power consumed by all functional units at any step
FU_k	functional unit of type *k*
M_k	maximum number of functional units of type *k*
$cost_k$	cost of functional units of type *k*
cstep	control step
λ	total number of control steps.

III. Exact Solution

Our proposed optimal algorithm for the multiple supply voltages scheduling (MVS) problem is as shown in Figure 1. It assumes that the clock-selection phase is already done and so the input delays for DFG nodes are expressed in number of csteps. First, the as-soon-as-possible (ASAP) and as-late-as-possible (ALAP) schedule (computed using the highest voltage-level) is calculated as a preprocessing step to tighten the time-frames for graph vertices and so the number of variables in the MILP formulation. Then, the MILP formulation is developed to solve one of the two problems, the *TCS problem* using the objective function (2) and the set of inequalities (3)-(6), or the *TRCS problem* using the objective function (2) and the set of inequalities (3)-(7).

- *Calculate ASAP and ALAP using the highest voltage-level.*
- *Construct the MILP formulation as in Equations (2)-(7).*
- *Use an ILP solver to solve the model.*
- *Construct the optimal schedule.*

Figure 1: Procedure of the multiple supply-voltages scheduling.

A. TCS Problem

Given the time constraint λ and a set of voltage levels for the operating resources, and a power/delay table that contains the average power consumption and the delay time needed for each resource operating on each voltage level, find a schedule that minimizes peak power and/or energy consumptions.

B. TRCS Problem

Given the time constraint λ, the number of resources of each type of computational element, a set of voltage levels for the operating resources, and a power/delay table that contains the average power consumption and the delay time needed for each resource operating on each voltage level, find a

schedule that minimizes peak power and/or energy consumption.

Define x_{ijv} to be a 0-1 unknown variable that takes value 1 if node *i* starts execution at cstep *j* with voltage level *v* and 0 otherwise. Then, the MILP formulation is as follows.

$$Minimize: \quad \alpha P_{peak} + \beta (1/\lambda) \sum_i \sum_j \sum_v x_{ijv} d(i,v).p(i,v) \qquad (2)$$

$$\sum_v \sum_j x_{ijv} = 1 \qquad \forall i \in V \qquad (3)$$

$$\sum_v \sum_j (j + d(i,v) - 1) x_{ijv} + \sum_v \sum_j j x_{ljv} \le -1 \qquad \forall (i,l) \in E \qquad (4)$$

$$\sum_i \sum_v \sum_{j_1 = j - d(i,v) + 1}^{j} x_{ijv} p(i,v) \le P_{peak} \qquad \forall j \in [1, \lambda] \qquad (5)$$

$$\sum_v \sum_j (j + d(i,v) + 1) x_{ijv} \le \lambda \qquad \forall node\ i\ without\ successors \quad (6)$$

$$\sum_{i \in FU_k} \sum_v \sum_{j_1 = j - d(i,v) + 1}^{j} x_{ijv} \le M_k \qquad \forall j \in [1, \lambda],\ FU_k \qquad (7)$$

$$x_{ijv} \in \{0,1\} \qquad (8)$$

Equation (2) is a flexible weighted objective function, where the weight factors can be set according to the design requirements. Equation (3) forces each node to start at only one cstep, and be scheduled using one and only one voltage level. The precedence relations are satisfied by Equation (4). Peak power can be set as a constraint or can be used as a variable to be minimized in the objective function as shown in Equation (5). To meet the latency requirement, each node without successors is forced to finish execution by λ through Equation (6). Finally, Equation (7) is presented to constrain the number of resources used from each type. Our MILP formulation is flexible since the peak power can be treated as an objective to be optimized or can work as a constraint for those applications that have hard limits on peak power consumption. In addition, for design space exploration, power consumption (average and/or peak) can be set as a constraint for area or time minimization.

IV. Iterative LP Relaxation

When the design problem is large, the solution run-time for the MILP becomes a problem because of the exponential nature of the ILP solution algorithms. LP relaxation might be a solution of this problem in some cases, but the quality of the final solution depends on the method of relaxation. In addition, because the LP relaxation of the IP problem in general is not integral, there should be some way to guide the relaxation to get a solution quality close to the optimal solution. The integrality constraint in Equation (8) can be relaxed by Equation (9). The fractional values that result from running the LP solution once by itself do not reflect meaningful information. Just rounding off the fractional value associated with the 0-1 variable in the final solution is not a good idea because first, the correctness is not guaranteed (for example, precedence relations among the nodes might be violated), second, even if the precedence relations are met, the final solution is far from the optimal one.

$$0 \le x_{ijv} \le 1 \qquad (9)$$

We devise a guided way of relaxation called "*iterative LP relaxation"* as shown in Figure 2. The idea is to develop the LP solution iteratively in several stages. In each stage, the variables (especially the 0-1 variables) that are relatively "large" (variables with largest fractional values, which if they are set to 1's, they lead to optimal or near-optimal solution) to contribute to the optimal solution are selected. If their resultant values are integral, they are set to these values and fixed during the successive iterations. If their resultant values are fractional, they are tested against a threshold value, and any 0-1 variable passing the test, it is set to one and fixed during the successive iterations. At the same time the rest of the 0-1 variables associated with the same node are set to zeros. The solution iterates until all the 0-1 variables pass the threshold test. The threshold value is set dynamically as the maximum value of the 0-1 variables from the resultant solution of the current LP iteration. This is to elect the most mature 0-1 variables to contribute to the final solution.

1. *Calculate ASAP and ALAP times using the highest voltage-level.*
2. *Construct the MILP formulation as in Equations (2)-(7).*
3. *Relax the integrality of the 0-1 variables by substituting (8) by (9).*
4. *Iterative procedures:*
   - *4.1 Use an LP solver to solve the model.*
   - *4.2 In case of resource constraint, if the solution is infeasible, increase the number of resources by one and solve again.*
   - *4.3 Set the threshold value to be the maximum of the 0-1 variables in the LP solution from step 4.1.*
   - *4.4 If any 0-1 variable passes the threshold do:*
     - *4.4.1 set its value to 1 and fix it during the successive iterations.*
     - *4.4.2 Set all the 0-1 variables associated with the same node to 0.*
   - *4.5 Update ASAP and ALAP values.*
   - *4.6 If NOT all 0-1 variables are set (to either 0 or 1) GoTo 4.1.*
5. *Construct the schedule.*

Figure 2: Iterative LP procedures.

In the case of time and resource constraint scheduling (TRCS), the LP output solutions may become infeasible during an iteration because the accumulated candidate 0-1 variables that have been set to one in this iteration might violate the resource constraints. Thus, at any iteration during the iterative relaxation, when the resultant LP solution is infeasible, the number of resources is increased by one and the

model is resolved again. That increase in the number of resources will be restored again using the power-resources -saving phase of the heuristic.

As a detailed example, consider the DFG in Figure 3-(a). Assume that the delays of a DFG operation are 2 and 1 time steps if it is scheduled with a high and low voltage-level, respectively and let each operation consume 20 μwatt and 8 μwatt average power if it is scheduled with a high and low voltage-level, respectively. The associated 0-1 variables for each DFG node for time constraint 4 are in Table 1. The iterative LP procedure takes 3 iterations to complete the solution. The outcome of each iteration is as shown in Table 2. After each iteration, the threshold value is set to the maximum value among the 0-1 variables written in bold face in Table 2. After the first iteration, x8 variable is set to 1 and the rest of associated variables of node c, x7 and x9, are set to zeros forcing node c to be scheduled at cstep 3 with the higher voltage level. Node d is scheduled at cstep 1 with the lower voltage level after its associated variable x13 passes the threshold value in the second iteration and is set to 1, while nodes a and b are scheduled in the third iteration. The final LP solution results in the same schedule as that obtained from the MILP solution as shown in Figure 3-(b).

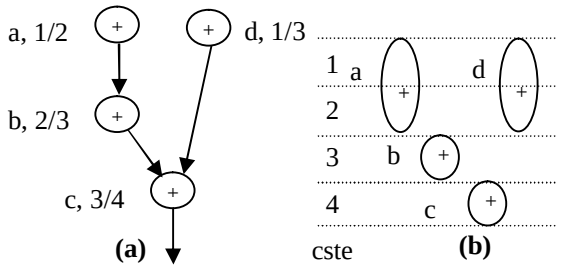

Figure 3: (a) Sample DFG annotated with ASAP/ALAP values for $\lambda = 4$, **(b)** simultaneous peak and average power minimization.

Table 1: DFG nodes and their associated 0-1 variables for Figure 3-(a)

node	indexed variables	ILP solver variables
a	x_{a11} x_{a21} x_{a12}	x1 x2 x3
b	x_{b21} x_{b31} x_{b22}	x4 x5 x6
c	x_{c31} x_{c41} x_{c32}	x7 x8 x9
d	x_{d11} x_{d21} x_{d31} x_{d12} x_{d22}	x10 x11 x12 x13 x14

Although, the worst-case number of iterations is the number of nodes in the DFG, the actual number of iterations is very small as will be clear in the experimental results for most benchmarks. This is because at each iteration many variables pass the threshold and the time-frames become tighter forcing many variables to settle.

Table 2: Iterative LP solution for the DFG in Figure 1-(a)

node	var	itr#1	itr#2	itr#3	Final sol
a	x1	0.45339	0.59979	0.47056	0
	x2	0	0	0	0
	x3	0.54661	0.40021	**0.52944**	1
b	x4	0.32353	0	0	0
	x5	0.54661	0.40021	**0.52944**	1
	x6	0.12986	0.59979	0.47056	0
c	x7	0	0	0	0
	x8	**0.67647**	1	1	1
	x9	0.32353	0	0	0
d	x10	0	0	0	0
	x11	0	0	0	0
	x12	0	0	0	0
	x13	0.63207	**0.67122**	1	1
	x14	0.36793	0.32878	0	0
	Ppeak	17.5	23	23	20

V. Power- Resources Saving

The goal of the second phase of the algorithm, power-resources saving procedure, is to gain additional power and/or resources saving through exploiting any available flexibility for an operation. It tries to schedule the DFG operation with a lower voltage level (more power saving) and/or to move it up and down within the available room without violating the peak power obtained from the first phase, to get more peak power saving and/or resources saving. The algorithm for power-resources saving is shown in Figure 4.

Power-resources saving*(scheduleStep, vLevel, peak_power)*
marked = 0 for all operations, count = 0
while(count < numOperations) do
for(op = 1: numOperations)
1. *if (marked(op)=1 or indegree(op) > 0) skip the rest of loop body.*
2. *update the time frames*
3. *compute the room of op using scheduleStep of its predecessors and its successors.*
4. *for(v = numVlevels:1 step -1)*
   - *4.1 if (there is a room to schedule op with v without violating the peak power and the input resource constraints if any)*
     - *4.1.1 schedule op at cstep within its time frame to get smaller power and/or resources and set:*
     - *4.1.2 marked(op) = 1.*
     - *4.1.3 vLevel(op) = v.*
     - *4.1.4 scheduleStep(op) = cstep.*
     - *4.1.5 count = count + 1.*

Figure 4: Power-resources saving algorithm.

The inputs to the algorithm are the resultant schedule attributes from the first phase (the iterative LP) where *scheduleStep* and *vLevel* are the cstep and the voltage-level stamped to each operation, respectively, and peak_power is the resultant peak power consumption from the first phase.

The power-resources saving algorithm has a worst-case time complexity in the order of $O(n^2)$, where n is the number of operations in the DFG. The detailed complexity derivation is found in [13].

VI. Experimental Results

The two proposed solutions, the exact and the two-phase heuristic, are tested on standard benchmarks like HAL, ARF, and EWF using the module library shown in Table 3. The Experiments take place on the *SUN ENTERPRISE 4500* workstation. This workstation has eight 333MHz SPARC CPU's and 2GB RAM, and it works with SOLARIS 8 operating system. In the tabulated results, "avg" means average power, "peak" means peak power, "[*, +]" means the number of [multipliers and adders] resources used, and "#itr" is the number of LP iterations until the iterative LP solution is finalized. Experiments are run for each benchmark with time constraint varying from the critical path length to twice the critical path length. The exact ILP solution and the results of the two-phase heuristic after each phase of the algorithm are tabulated in the set of columns under the heading "ILP Sol", "Iterative LP Sol", and "After Power Saving", respectively. Results of the tested benchmarks under different time constraints for the TCS problem are shown in Tables 4 through 6; while the results for the TRCS problem are shown in Tables 7 through 10 under different sets of resource constraints. Results of the iterative LP approach followed by the power-resources saving procedure are compared to the optimal solution (ILP solution) as tabulated. These results show that, in most cases the results of our heuristic well match those obtained from the optimal solution; and for those results that do not exactly match the optimal solution, the error is very small. The results also show the efficiency in run time of the iterative LP solution compared to the ILP. For example, the run time for the ARF benchmark under time constraint 19 in Table 5 takes a fraction of second for the iterative LP solution compared to 132 seconds for the exact solution; while the two solutions are the same. In case of the TRCS problem, resource constraints might be violated in the LP solution under certain time constraints as shown in Table 10. However, the resource constraints are satisfied in the final solution using the power-resources saving procedure as shown in the results.

In addition to the structure of the DFG, the quality of the iterative LP solution is affected by the distance between the time constraint and the length of the critical path, which is reflected in the length of time-frame of each DFG node and so the number of 0-1 associated variables. When the time constraint is very close to the critical path length, the number of variables associated with each node is very small limiting the number of candidate variables that contribute to the objective function minimization. On the other hand, when the time constraint is far from the critical path length, the decision taken at a certain point to fix the schedule of some nodes does not have much impact on those nodes whose time-frames get limited. This is because there is a big chance for each node even after its time-frame is restricted to have a variable to be mature enough to contribute to the objective function minimization. As the results show, the iterative LP solution is very close and matches in most cases the exact ILP solution at both ends of the results tables, when the time constraint is close to or far from the critical path length. The MILP solution run time is not only affected by the structure of the DFG and the number of nodes in it, but it is also affected by how far the time constraint is from the critical path length as shown in most benchmarks especially the EWF benchmark in Tables 6, 9, and 10.

Table 3: Modules library for MVS

Module	5.0 V		3.3 V	
	d	power	d	power
MULT16	2	84	4	13
ADD16	1	26	2	6
SUB16	1	26	2	6

Table 4: HAL under TCS

	ILP Sol				**Iterative LP Sol**					**After Power Saving**		
L	avg	peak	[*,+]	time	avg	peak	[*,+]	#itr	it_t	avg	peak	[*,+]
6	162.33	265	4,3	0.01	162.33	265	4,3	3	0.03	162.33	265	4,3
7	122.57	181	3,3	0.02	122.57	181	3,3	3	0.03	122.57	181	3,2
8	78.25	110	4,3	0.03	78.27	123	4,3	4	0.04	78.27	110	4,3
9	55.4	97	4,2	0.12	56.67	97	4,3	5	0.05	55.4	97	4,3
10	39.4	45	3,3	0.30	39.4	45	3,3	5	0.05	39.4	45	3,3
11	34.82	39	3,3	0.06	34.82	39	3,3	7	0.07	34.82	39	3,3
12	31	39	3,3	0.04	31	39	3,2	8	0.08	31	39	3,3

Table 5: ARF under TCS

	ILP Sol				Iterative LP Sol					After Power Saving		
L	avg	peak	[*,+]	time	avg	peak	[*,+]	#itr	it_t	avg	peak	[*,+]
11	225.27	362	6,4	0.03	225.27	388	8,4	2	0.02	225.27	362	6,3
12	215.25	349	5,4	0.22	204.67	388	8,4	6	0.12	204.67	362	6,4
13	154.92	336	6,4	0.65	188.92	388	8,4	7	0.28	187.23	362	6,4
14	142.28	336	8,3	1.86	142.28	348	8,4	7	0.42	142.28	336	8,4
15	103.33	194	8,4	1.74	103.33	336	8,4	6	0.42	103.33	336	6,2
16	95.5	194	8,3	7.49	95.5	336	8,4	5	0.4	95.5	336	6,3
19	54.84	64	4,2	132	54.84	64	4,4	6	0.66	54.84	64	4,2
22	45.36	52	4,3	22.3	44.36	64	4,4	5	0.75	44.36	64	4,3

Table 6: EWF under TCS

	ILP Sol				Iterative LP Sol					After Power Saving		
L	avg	peak	[*,+]	time	avg	peak	[*,+]	#itr	it_t	avg	peak	[*,+]
17	111.64	252	3,5	0.02	111.64	258	3,5	5	0.05	111.64	258	3,5
18	105.4	168	2,5	0.06	103	252	3,5	7	0.14	103	252	3,5
20	71.15	110	3,5	1.77	71.7	168	4,5	10	0.7	70.6	168	4,4
21	56.19	97	4,5	3.69	60	107	4,5	10	1.1	60	107	4,5
28	30.7	37	2,5	677	29.14	39	3,5	22	9.68	29.14	39	3,5
34	22.7	26	2,4	3340	22.7	39	2,4	24	16.6	22.7	32	2,4

Table 7: HAL [3*, 3+] under TRCS

	ILP Sol			Iterative LP Sol					After Power Saving		
L	avg	peak	time	avg	peak	[*,+]	#itr, #f	it_t	avg	peak	[*,+]
6	183.5	252	0.02	183.5	275	3,3	4,0	0.04	181.67	258	3,3
7	122.57	181	0.02	122.57	181	3,3	3,0	0.03	122.57	181	3,2
8	92.75	181	0.33	92.75	181	3,3	8,0	0.08	92.75	181	3,3
9	56.67	110	0.18	56.67	110	3,3	5,0	0.05	56.67	110	3,2
10	39.4	45	0.30	39.4	45	3,3	4,0	0.08	39.4	45	3,3
11	34.82	39	0.06	34.82	39	3,3	8,0	0.08	34.82	39	3,3
12	31	39	0.05	31	39	3,2	8,0	0.24	31	39	3,3

Table 8: HAL [2*, 2+] under TRCS

	ILP Sol			Iterative LP Sol					After Power Saving		
L	avg	peak	time	avg	peak	[*,+]	#itr, #f	it_t	avg	peak	[*,+]
7	155.71	174	0.02	155.71	174	2,2	3,0	0.03	155.71	174	2,2
8	139	168	0.19	137.62	214	2,2	8,0	0.08	133.5	174	2,2
9	95.33	103	0.10	56.67	116	3,2	5,0	0.1	69.55	103	3,2
10	83.6	103	1.34	64.8	113	3,2	4,0	0.08	61.5	103	3,2
11	56.9	97	0.73	56.91	97	2,2	8,0	0.24	56.91	97	2,2
12	50.33	97	1.7	40.67	97	3,2	8,0	0.16	50.33	97	2,2

Table 9: EWF [3*, 5+] under TRCS

	ILP Sol			Iterative LP Sol					After Power Saving		
L	avg	peak	time	avg	peak	[*,+]	#itr, #f	it_t	avg	peak	[*,+]
17	111.64	252	0.02	111.64	252	3,5	4,0	0.04	111.64	252	3,5
18	105.44	168	0.09	103	258	3,5	6,0	0.18	103	252	3,5
20	71.15	110	2.69	86.35	187	3,5	12,0	0.84	84.7	187	3,5
21	61.19	107	9.79	60.67	126	3,5	11,0	1.32	60.67	126	3,5
28	30.71	37	1704	29.14	39	3,5	21,0	8.61	29.14	39	3,5
34	22.7	26	12350	22.7	31	2,5	24,0	14.4	22.7	29	2,4

Table 10: EWF [2*, 3+] under TRCS

	ILP Sol			Iterative LP Sol					After Power Saving		
L	avg	peak	time	avg	peak	[*,+]	#itr, #f	it_t	avg	peak	[*,+]
18	104.83	174	0.05	104.83	258	3 3	8,0	0.16	103.6	186	2,3
20	80.55	120	5.57	80.55	126	2,3	17,0	1.7	80.55	120	2,3
21	71.19	107	18.67	77.24	136	2,3	19,0	2.47	75.14	119	2,3
28	29.92	38	662	29.92	38	2,3	19,0	10.64	29.92	38	2,3
34	22.7	26	3860	22.7	32	2,3	23,0	18.17	22.7	32	2,3

VII. Conclusion

In this paper, we have presented an MILP formulation for the scheduling problem using multiple supply-voltages in order to optimize peak power as well as average power and energy consumptions. Our exact solution for optimal peak and/ or average power scheduling is considered under two sets of constraints, time constraint alone (TCS), and time and resource constraints (TRCS). Then, we have devised a two-phase heuristic to solve the multiple supply-voltages scheduling for peak and average power minimization using the same sets of constraints. First, we developed a guided LP relaxation in which the MILP formulation is iteratively relaxed to obtain a minimal peak and/or average power minimization. Then in the second phase, the power-resources saving procedure is developed to restore the violation in resource constraints (if any) in case of TRCS or to achieve minimal resource usage in case of TCS; and to restructure the output LP schedule in order to obtain more power saving. The results for peak and average power of our two-phase heuristic well match those obtained by the optimal solution as have been validated by extensive experiments on several benchmarks

References

[1] L. Wang, Y. Jiang, and H. Selvaraj, "Synthesis Scheme for Low Power Designs with Multiple Supply Voltages by Heuristic Algorithms," in *Proc. ITCC*, pp. 826-829, 2004.

[2] J. Chang and M. Pedram, "Energy Minimization Using Multiple Supply Voltages," *IEEE Trans. on VLSI Systems*, vol. 5, no. 4, pp. 436–443, Dec. 1997.

[3] Y.-R. Lin, C.-T. Hwang, and A. C.-H. Wu, "Scheduling Techniques for Variable Voltage Low Power Designs," *ACM Trans. on Design Automation and Electronic Systems*, vol. 2, no. 2, pp. 81–97, Apr. 1997.

[4] W.-T Shiue and C. Chakrabarti, "ILP-Based Scheme for Low Power Scheduling and Resource Binding," in *Proc. IEEE Intl. Symp. on Circuits and Systems*, pp. 279--282, May 2000.

[5] A. Manzak and C. Chakrabarti, "A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages." *IEEE Trans. on VLSI Systems*, vol. 10, No 1, pp. 6-14, Feb. 2002.

[6] N. Chabini, I. Chabini, E.-M. Aboulhamid, and Y. Savaria, "Methods for Minimizing Dynamic Power Consumption in Synchronous Designs with Multiple Supply Voltages," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 3, pp. 346-351, Mar. 2003.

[7] S. Gupta and S. Katkoori, "Force-Directed Scheduling for Dynamic Power Optimization," in *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 68-73, 2002.

[8] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi, "Simultaneous Peak and Average Power Minimization During Datapath Scheduling for DSP Processors," in Proc. *ACM Great Lakes Symposium on VLSI*, pp. 215-220, 2003.

[9] W.-T. Shiue, "High Level Synthesis for Peak Power Minimization Using ILP," in *Proc. IEEE Int'l Conf. on Application Specific Systems, Architectures and Processors*, pp. 103–112, 2000.

[10] P. G. Paulin and J. P. Knight, "Force Directed Scheduling for the Behavior Synthesis of ASICs," *IEEE Transactions on CAD*, vol. 8, pp. 661-679, June 1989.

[11] W.-T. Shiue and C. Chakrabarti, "Low Power Scheduling with Resources Operating at Multiple Voltages", *IEEE Transactions on Circuit and Systems Part II: Analog and Digital Signal Processing*, vol. 47, no. 6, pp. 536-543, June 2000.

[12] B. Radhakrishnan and M. Venkatesan, "Multiple Voltage and Frequency Scheduling for Power Minimization," in *Proc. Euromicro Symp. on Digital System Design*, pp. 279-285, Sep. 2003.

[13] A. K. Allam and J. Ramanujam, "Modified Force Directed Scheduling for Peak and Average Power Optimization Using Multiple Supply Voltages," *International Conference on IC Design and Technology (ICICDT)*, Padova, Italy, May 2006