Tabu Search Based Multiple Voltage Scheduling under both Timing and Resource Constraints

Jianmo Ni[†‡], Nan Wang[†], Takeshi Yoshimura[†]

[†]Graduate School of IPS, Waseda University, Hibikino 2-7, Wakamatsu-ku, Kitakyusyu, 808-0135, Japan

[‡]Dept. Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
E-mail: nijianmo@gmail.com

Abstract— In this work, we address the multiple voltage scheduling problem to minimize power consumption under both timing and resource constraints. We develop a tabu search-based algorithm with a general vector representation of solution and an effective performance estimation. Moreover, our approach tends to solve a series of scheduling problems with respect to multiple voltage designs. Specifically, our method represents each solution by a vector of operation types and solves the scheduling subproblem corresponding to a certain solution vector. To get feasible solutions of the subproblem, a two-stage method is presented by first adopting the PLNSM algorithm [3] to generate delay assignment satisfying the timing constraint and then performing delay adjustment iteratively until the resource constraints are met. A heuristic estimation is introduced to predict the total power and resource usage of the neighborhood. Our proposed method achieves near-optimal solutions with only an average power increase of 0.84% compared with ILP for small-size designs, and a 24.1% reduction over a previous tabu search-based algorithm [7] for a set of benchmarks.

Keywords— Multiple voltage scheduling, tabu search, vector representation

I. Introduction

In high level synthesis (HLS), the scheduling problem is a central task due to its significant impact on design structure and metrics. In recent years, reducing power consumption has become one of the important issues and multiple voltage technique has been proposed as one of the promising methods because of the quadratic relationship between voltage and power. Consequently, the multiple voltage scheduling (MVS) problem has been widely concerned in low power VLSI designs. In this work, we focus on the MVS problem for power minimization under both timing and resource constraints.

According to different constraints, the related works can be classified into three categories: timing constrained [1] [2] [3] [5] [6] [8], resource constrained [4] [5] [6] and both timing and resource constrained [5] [7]. Hao et al. [3] presented a Piecewise Linear Network Simplex Method (PLNSM) to solve a timing constrained MVS problem which minimized the power and resource usage simultaneously. Zhang et al. [8] proposed a Dual Piecewise Linear Network Simplex Method (DPLNSM) to minimize power and resource usage under timing constraint. However, both [3] and [8] cannot be applied under resource constraints, which are important factors in practice. Lin et al. [5] first gave an ILP formulation for the latency and resource constrained MVS problem, and then proposed an iterative improvement method based on a priority function. In this paper, we define a new priority function by adding the resource usage estimation into consideration, which effectively reflects the congestion of the scheduling and thus improves the performance of the priority function.

Wang et al. [7] developed a tabu search-based scheduling scheme (TS_scheme) which combined the scheduling and partitioning problems. The TS_scheme has a disadvantage that its initialized solution is simply generated by assigning the highest voltage to each operation. This leads to a lack of power-awareness and makes the algorithm less effective. Moreover, resource constraints in [7] are given specifically for each supply voltage of each resource type. A more practical way is to consider the total number of a certain resource type as constraint, which provides more flexibility because the supply voltage of each resource can be assigned freely. Consequently, the delay assignment of operations becomes more flexible and more operations can be assigned to lower supply voltages, which leads to a smaller power consumption. In our problem, we consider the total number of each resource type as resource constraints.

In this paper, we propose a tabu search-based method with a general vector representation of solution and an effective performance estimation. Each solution is presented by a vector of operation types of operations. Given the solution vector, we present a two-stage method to solve the MVS subproblem. Firstly, the PLNSM algorithm [3] is run to produce a resource-aware delay assignment satisfying the timing constraint under an input of operation types. Secondly, delay adjustment is performed iteratively based on a priority function until the resource constraints are met. When searching for a new solution in the neighborhood of the current solution, the solution vector is changed and the two-stage method is performed to obtain the feasible solution. A heuristic estimation is introduced to estimate the resource usage under a given delay assignment. With this estimation, the performance of the neighborhood can be predicted by considering both power and resource. Therefore, the neighbor with a better performance can be picked out effectively.

II. Problem Definition

The input Data Flow Graph (DFG) is denoted as $G = (V, E)$. $V = \{s, t\} \cup V_{op}$ is a set of nodes, where V_{op} represents the set of operations and s and t are the source and sink nodes. $E = E_{op} \cup E_s \cup E_t$ is a set of edges, in which each edge of E_{op} represents the data dependency between two operations, E_s is the set of directed edges from s to the zero in-degree operations, and E_t is the set of directed edges from zero out-degree operations to t.

TABLE I
Operation Types

Operation Type	Possible Delay Interval
mul1	$DL_{mul1} = \{2, 3, 4\}$
mul2	$DL_{mul2} = \{2, 3\}$
mul3	$DL_{mul3} = \{2\}$
add1	$DL_{add1} = \{1, 2, 3\}$
add2	$DL_{add2} = \{1, 2\}$
add3	$DL_{add3} = \{1\}$

We consider the set of operations $V_{op} = \{v_i | 1 \leq i \leq n\}$ with n_{type} operation types. For each operation v_i, its operation type is represented as $type(v_i) = op_k \in OP = \{op_k | 1 \leq k \leq n_{type}\}$. For each operation type $op_k \in OP$, *possible integer delay interval* represents the set of the delay values that can be assigned to operations of type op_k. It is represented as $DL_{op_k} = \{d | d^k_{min} \leq d \leq d^k_{max}\}$. In this paper, We define 6 operation types as Table I shows. Operations with operation type $mul1, mul2, mul3$ or $add1, add2, add3$ can be executed by resource $Multiplier$ or $Adder$, respectively. $Multiplier$ and $Adder$ are two specific resource types and their delay intervals with respect to different supply voltages are given as $DL_{mul} = \{2, 3, 4\}$, $DL_{add} = \{1, 2, 3\}$.

The MVS problem under both timing and resource constraints can be defined as: Given a data flow graph, a module library which contains multiple-supply-voltage designs of the same resource, a timing constraint T_{con} which is given as number of the maximum control steps, and resource constraints R^{tp} for each resource type tp which represents the maximum allowable number of resources at each control step. Our goal is to find a delay assignment for each operation, then schedule them into appropriate control steps while all constraints are satisfied, and minimize the total dynamic power consumption. It can be formulated as follows:

$$min: \quad \sum_{v_i \in V} P_{type(v_i)}(D(v_i)) \tag{1}$$

$$s.t: \quad \sum_{d \in DL_{tp}} R^{tp}_d \leq R^{tp}, \forall tp \in TP \tag{2}$$

$$0 < t - s \leq T_{con} \tag{3}$$

where $P_{type(v_i)}(D(v_i))$ is the dynamic power consumption of the operation whose operation type is $type(v_i)$ and delay is $D(v_i)$; R^{tp}_d is the number of assigned resources of type tp with delay d which corresponds to a specific supply voltage. TP is the set of resource types. As Eq.(2) shows, in this paper we consider the total number of resources of each voltage supply as resource constraint.

III. Overview of the Applied Tabu search

The solution representation in [7] consists of the final resource allocation of each operation. It has a limitation that the supply voltage of each resource has to be determined before scheduling. When the total number of resources is taken as constraint, the supply voltage of each resource remains to be decided during the scheduling process. Therefore, the previous solution representation is not applicable here. In our problem, a more general solution representation is introduced. Each solution S is defined as a vector of the operation type of each operation:

$$S = (opt_1, opt_2, ..., opt_i, ..., opt_n) \tag{4}$$

With this representation, each solution vector consists of less elements and the limitation on resource allocation is removed. Basically, the MVS problem can be divided into two subproblem: delay assignment and scheduling. Given a different solution vector in the current representation, a different result of delay assignment is generated and a unique solution is finally obtained.

A. Generate initial solution

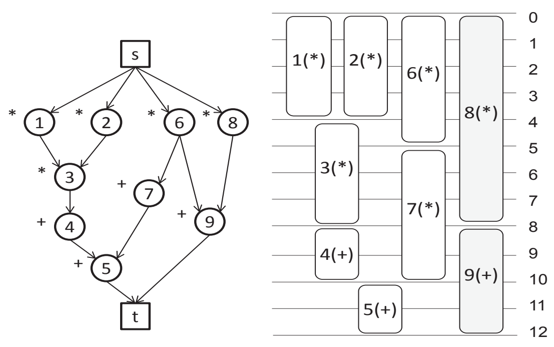
To start the algorithm, we have to initialize a solution S_0. The operation type of each operation is set to be the one with the largest possible delay interval. With a given solution vector, we presented a two-stage method to solve the MVS subproblem with respect to the solution vector.

B. Find local best in neighborhood

The neighborhood $N(S)$ of the current solution S is the set of solutions obtained by a move. A move from the solution S to a neighbor S_i is defined as one of the elements opt_i in S is changed to the opt^*_i in S_i. In each iteration, the neighborhood of the current solution is generated. A performance estimation is introduced to predict the quality of the neighbors. The neighbor with the best estimated value is chosen as the next state of the tabu search and the current solution vector is stored in a tabu list. When tabu search terminates, the scheduling result with a near-optimal power consumption is returned.

IV. Scheduling under Timing and Resource Constraints

In this section, we discuss the two-stage method to solve the MVS subproblem. At first, the PLNSM [3] algorithm is adopted to provide a power-optimum delay assignment with only timing constraint. Next, the delay adjustment is conducted repeatedly until resource constraints are met. In each iteration of the adjustment, one of the operation is picked out and its operation type is changed so as to change the delay assignment.



(a)DFG (b)mobility allocation

Fig. 1. An example of DFG and mobility allocation

A. Mobility Allocation and Delay Determination

Figure 1(a) shows an example of a DFG $G = (V, E)$. Given a DFG and the operation type of each operation, the PLNSM algorithm generates the mobility allocation and therefore the mobility of each operation can be calculated. In general, the *mobility* of the operation v_i is defined as an interval of consecutive control steps $M(v_i) = \{t|ms_{v_i} \leq t \leq me_{v_i}\}$, where $ms_{v_i} \geq T_{asap}(v_i)$ and $me_{v_i} \leq T_{alap}(v_i)$. Here $T_{asap}(v_i)$ and $T_{alap}(v_i)$ are control steps where v_i is scheduled by ASAP and ALAP scheduling.

Given the mobility and the operation type of the operation v_i, we choose the maximum delay that can be assigned within the mobility because the operation with a larger delay consumes smaller power. It can be denoted as: $D(v_i) = min\{me_{v_i} - ms_{v_i}, d_{max}^{type(v_i)}\}$. Figure 1(b) gives an example of the mobility allocation and the delay assignment for the first iteration. For example, the mobility of operation 8 is $M(v_8) = [0, 8]$, thus its delay $D(v_8)$ is assigned to $min\{8, d_{max}^{mul1}\} = min\{8, 4\} = 4$.

B. Delay Adjustment

After the delay assignment is determined based on the mobility allocation, list scheduling is performed to minimize the resource usage.

When list scheduling finds no feasible solution, we have to change the delay assignment of the operations. Essentially, how to choose the operation to be changed is the key problem. We define a priority function by adding the consideration of resource usage to the priority function in [5] as:

$$Prio(v_i) = (\alpha PP^*(v_i) + \beta(1 - M^*(v_i)) + \gamma RU^*(v_i)) * W \tag{5}$$

here $PP^*(v_i)$ is the normalized power penalty of operation v_i, which represents the power increase with respect to the delay reduction; $M^*(v_i)$ is the normalized mobility of operation v_i; $RU^*(v_i)$ is the normalized resource utilization of the resource $R(v_i)$ that executes the operation v_i. Resource utilization represents how many of the total control steps are occupied for a certain resource and is estimated as: $RU(v_i) = T_{occupied}^{R(v_i)}/T_{con}$, where $T_{occupied}^{R(v_i)}$ is the number of occupied control steps of the resource $R(v_i)$. α, β and γ are coefficients and are set to 5, 1 and 2.5. γ is set to 0 in some cases that the delay assignment violates the resource constraints and there is no list scheduling result. W is the factor related to the *resource usage* estimation, here *resource usage* means the number of resources used in the scheduling.

Intuitively, if the resource constraints are violated , there must be congestion of operations in some control steps of the current mobility allocation. To satisfy the resource constraints, one of the methods is to assign the operations on these control steps with shorter delays, which reduces the congestion in theses control steps and increases the flexibility of the scheduling. Since the exact resource usage can not be determined until the final scheduling is obtained, a heuristic resource usage estimation is introduced to predict the congestion of each control step.

For each operation v_i, $prob_i^t$ is defined as the probability that v_i is scheduled in control step $t (1 \leq t \leq T_{con})$. Assuming that the probability has a uniform distribution, it can be estimated as:

$$prob_i^t = \begin{cases} \frac{D(v_i)}{me_{v_i} - ms_{v_i}}, & ms_{v_i} \leq t \leq me_{v_i} \\ 0, & other \end{cases} \tag{6}$$

The resource usage of resource type *tp* for control step t can be estimated by summing up the probability $prob_i^t$ of each operation which is executed by the resource of type *tp*. It can be denoted as:

$$ER_t^{tp}(S_i) = \left\lfloor \sum_{\forall Rtype(v_i) = tp} \{Prob_i^t\} \right\rfloor \tag{7}$$

here S_i represents an input solution vector; $Rtype(v_i)$ is the type of the resource executing operation v_i. Then the resource usage of each resource type can be estimated as the maximum estimated value among all the control steps:

$$ER^{tp}(S_i) = \max_{1 \leq t \leq T_{con}} \{ER_t^{tp}(S_i)\} \tag{8}$$

The factor W can be calculated as the ratio with the estimation resource usage and the constraint:

$$W^{tp} = ER^{tp}(S_i)/R^{tp} \tag{9}$$

If $W^{tp} \leq 1$, then W is set to 1; otherwise, $W = W^{tp}$ when calculating the factor W of a certain operation v_i which is executed by a resource of type *tp*.

After we calculate the weight of each operation by Eq.(5), we choose v_i with highest weight and change its operation type to the type with shorter delay interval. By changing the operation type, operation v_i will not be assigned to a larger delay in the new iteration, which makes it more likely to satisfy the resource constraints.

V. Heuristic Performance Estimation

As mentioned before, the exact result of the neighbor can not be determined until the two-stage method finds out the feasible solution, which needs to run list scheduling iteratively and is quite time consuming. Naturally, a heuristic evaluation is introduced to accelerate the search process.

Since the PLNSM method guarantees that the solution satisfies the timing constraint, our goal is to find out solutions which are more likely to satisfy the resource constraints with minimum power increase. Therefore, the estimation takes into consideration two parts: the total power and resource usage of each neighbor solution vector S_i. The heuristic function can be expressed as

$$Cost(S_i) = \frac{Power(S_i)}{Power^t(S)} + \frac{ER(S_i)}{\sum\limits_{\forall tp \in TP} R^{tp}} \quad (10)$$

For each solution vector S_i, $Power(S_i)$ is the corresponding power consumption of the delay assignment determined by the PLNSM algorithm. $Power^t(S)$ is the power consumption of the feasible scheduling obtained of the current solution S. By Eq.(8), we can estimate the resource usage for each resource type. Naturally, the total resource usage $ER(S_i)$ can then be estimated as

$$ER(S_i) = \sum_{\forall tp \in TP} ER^{tp}(S_i) \quad (11)$$

Considering if the resource constraints are violated, it is obvious that the delay assignment with a smaller estimated resource usage is more likely to be the final feasible solution. Therefore, a better neighbor is a solution that has a largely decreased estimated resource usage with only a small power penalty. According to Eq.(10), the solution S_i with the smallest $Cost(S_i)$ is preferred and chosen as the next solution vector.

As other iterative improvement methods, tabu search can not ensure the global optimality. Specifically in our problem, even though we choose the neighbor with the best estimated value in each iteration, the power of its scheduling result might be worse than the previous solution. This situation is more likely to happen when the initial solution has a relatively good performance (for example, when the timing constraint is loose, the initial solution will be near-optimal which means there is no much room for improvement). To escape from such situations, a restart technique is introduced:

- If the power of the scheduling obtained in the current iteration is worse than the previous iteration consecutively for a certain times, the search will stop and restart a new iteration with a randomly initialized solution state.

VI. Experimental Results

In this work, our proposed tabu search-based algorithm and two other algorithms (ILP and the TS_scheme [7]) have been implemented in C and run on a Linux Workstation with ARM Opteron 2.6GHz CPU and 4GB memory. As Table II shows, a set of benchmarks from [10, 11] has been tested: diff, ar, ellip, fft, rand0, rand1. The power and delay data are from [9].

TABLE II
Details of Benchmarks

DFG	description	operations	dependency
diff	differential equation	11	8
ar	ar lattice filter	28	30
ellip	ellipse filter	34	33
fft	fast fourier transform	129	240
rand0	random graph[10]	91	108
rand1	random graph[10]	157	189

TABLE III
Experimental Results of Optimality Study

DFG	TF	[+,*]	ILP (mW)	TS_improved (mW)	Cmp
diff	1.0	[2,3]	15877	16827	1.0598
	1.5	[3,4]	13523	13523	1.0000
	2.0	[4,6]	13523	13523	1.0000
ar	1.0	[2,5]	45504	45892	1.0085
	1.5	[3,7]	41440	41440	1.0000
	2.0	[4,10]	41056	41056	1.0000
ellip	1.0	[4,4]	51152	51536	1.0075
	1.5	[6,6]	49872	49872	1.0000
	2.0	[8,8]	49872	49872	1.0000
AVG					1.0084

A. Optimality Study

To evaluate the optimality of our improved tabu-search based algorithm (TS_improved), we tested it against the ILP formulation [5] for three small-size benchmarks: ar, ellip, diff. For each benchmark, the timing constraint is set as *critical path length* T_{cri}, which represents the minimum control steps required when all the operations are assigned to the smallest delay. Several resource constraints are considered, denoted as $R^{tp} = TF * R^{tp}_{min}$. Here, TF is the trade-off factor, and R^{tp}_{min} is the minimum number of resources of type tp required when all the operations are assigned to the smallest delay and scheduled by force-directed scheduling. Table III tabulates the comparison result. Column 1-3 show the benchmark, TF factor and resource constraints, respectively. Column 4 and 5 list out the power consumption of ILP and the TS_improved algorithm. Compared with the ILP method, the TS_improved algorithm produced results with only a 0.84% power increase in average.

B. Comparison to previous work and performance analysis

To evaluate the efficiency of the proposed algorithm, we compared it with the TS_scheme [7]. In our experiment, the iteration times of the TS_improved algorithm was set as 20, and the tabu list length as 7. For the TS_scheme, iteration times and tabu list length were set to 100 and 15, respectively. Table IV summaries the power and CPU run time results of the two algorithms. Column DFG and R show the benchmark and resource constraints. The resource constraint R represents the maximum available number of resources for either $Multiplier$ or $Adder$. The timing constraint is set as $2T_{cri}$ for all the benchmarks.

As Table IV shows, our proposed algorithm achieves a

TABLE IV
COMPARISON WITH PREVIOUS TS_scheme [7]

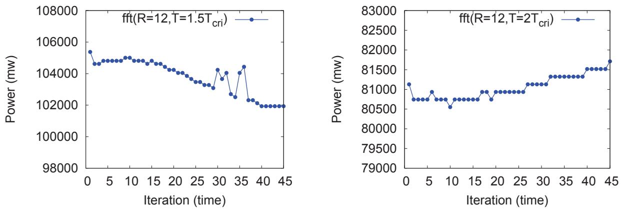
DFG	R	TS_scheme[7]		TS_improved	
		Power(mw)	Time(s)	Power(mw)	Time(s)
diff	3	9269	0.612	6909	1.04
	6	7105	0.287	6909	1.06
ar	6	22900	1.25	18572	1.23
	9	19156	0.983	18572	1.17
ellip	3	30120	1.201	23130	3.85
	6	22348	0.723	21190	2.14
fft	12	119537	3.691	80359	66.19
	15	120543	3.051	79595	64.38
rand0	6	-	13.877	61084	24.07
	9	81298	4.264	54938	20.27
rand1	12	179757	8.717	108075	134.93
	15	180459	8.255	94827	126.97

further 24.1% power reduction over the TS_scheme. Moreover, the reduction ratio increases when the DFG input is larger. It is because that initial solution of TS_improved can effectively shrink the solution space while the TS_scheme needs to start searching from the worst solution. There are even some cases that the TS_scheme cannot find feasible solutions when resource constraints or timing constraints are tight. For example, when benchmark rand0 is under constraint $R = 6$. In contrast, the proposed method is able to solve due to the higher flexibility of the supply voltage assignment of the resources. The run time results show that the TS_improved algorithm requires a longer run time than the TS_scheme. The reason is that the TS_improved algorithm needs to run the PLNSM and delay adjustment to get feasible solutions in each iteration. For larger input DFG, delay adjustment may be executed for hundreds of times before finding a feasible solution, which is quite time consuming. Meanwhile, the TS_scheme only needs to do the list scheduling once to obtain a feasible solution in each iteration. The run time of our algorithm is large but still acceptable.

Figure 2 illustrates the consistency of the TS_improved method. As figure 2(a) plots, for benchmark fft ($T_{con} = 1.5T_{cri}$), the solutions converge within 45 iteration times, which shows a fast convergence. In fact, for all the benchmarks tested when timing constraint ranges from T_{cri} to $1.5T_{cri}$, fast convergence is observed. However, when constraints are loose, tabu search is likely to produce worse solutions because the performance of the initial solution gets better under less strict constraints. For instance, when T_{con} is loosened to $2T_{cri}$ as figure 2(b) shows. To avoid such situations under loose constraints, a restart technique is proposed. Figure 3 shows the comparison of the TS_improved with or without restart for benchmark fft. Two loose timing constraints near $2T_{cri}$ are considered here. With the restart introduced, the search process becomes more stable and escapes from the situation of getting worse gradually.

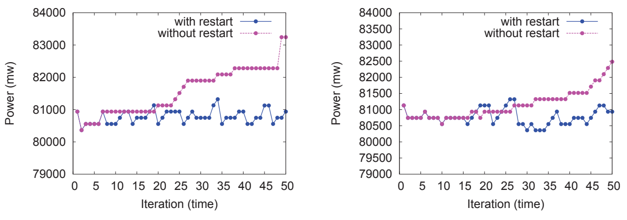
VII. **Conclusion**

In this paper, we have addressed a tabu search-based multiple voltage scheduling by introducing a general solution vector representation and a performance estima-



(a)fft($R = 12, T_{con} = 1.5T_{cri}$) (b)fft($R = 12, T_{con} = 2T_{cri}$)

Fig. 2. Consistency analysis of the TS_improved



(a)fft($R = 12, T_{con} = 39s$) (b)fft($R = 12, T_{con} = 40s$)

Fig. 3. Comparison of the tabu search with and without restart

tion technique to minimize power consumption under timing and resource constrains. The proposed TS_improved scheme shows quick convergence under normal or strict constraints and a restart technique is introduced to maintain the convergence when constraints are loose. Experimental results showed an near-optimality for small benchmarks and a significant power reduction over the previous TS_scheme.

REFERENCES

[1] A.K. Allam and J. Ramanujam, *Modified Force-Directed Scheduling for Peak and Average Power Optimization using Multiple Supply-Voltages*, IEEE International Conference on ICICDT, PP. 1-5, 2006

[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] C. Hao, S. Chen, T. Yoshimura, *Network Simplex Method Based on Multiple Voltage Scheduling in Power-Efficient High-Level Synthesis*, ASP-DAC, 2013

[4] L. Wang, Y. Jiang, and H. Selvaraj,, *Scheduling and Optimal Voltage Selection with Miultiple Supply Voltages under Resource Constraints*, the VLSI Journal, vol. 40, pp.174-182, 2007

[5] Y.-R. Lin, C.-T. Hwang, and A. C.-H. Wu, *Scheduling Techniques for Variable Voltage Low Power Designs*, AC, Trans. On Design Automation and Electronic Systems, vol. 2, no. 2, pp, 81-97, April 1997

[6] W.-T. Shiue and C. Chakrabarti, *Low-Power Scheduling with Resources Operating at Multiple Voltages*, IEEE Transactions on Circuits and Systems, vol. 47, 2000

[7] L. Wang, Y. Jiang, and H. Selvaraj, *Synthesis Scheme for Low Power Designs with Multiple Supply Voltages by Tabu Search*, Proc. IEEEE ISCAS, pp. 261-264, 2004

[8] H. Zhang, C. Hao, N. Wang, S. Chen and T. Yoshimura, "Power and resource aware scheduling with multiple voltages," ASIC (ASICON), 2013 IEEE 10th International Conference on , vol., no., pp.1,4, 28-31 Oct. 2013

[9] W. Jiang, Z. Zhang, M. Potkonjak and J. Cong, *Scheduling with integer time budgeting for low-power optimization*, Design Automation Conference, pp. 22-27, 2008

[10] David Rhodes, Robert Dick, *http://ziyang.eecs.umich.edu/dickrp /tgff/*

[11] C.Lee, M. Potkonjak, and W. Mandione-Smith, *Mediabench: a tool for evaluating and synthesizing multimedia and communications systems in Microarchitecture*, Thirtieth Annual IEEE/ACM International Symposium, pp. 330-335, 1997