A Gate Resizing Technique for High Reduction in Power Consumption

P. Girard C. Landrault S. Pravossoudovitch D. Severac

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier,
UMR 5506 UNIVERSITE MONTPELLIER II / CNRS
161 rue Ada, 34392 Montpellier Cedex 05, FRANCE.
E–mail: girard@lirmm.fr Web page: http://www.lirmm.fr/~w3mic

Abstract

With the advent of portable and high density microelectronic devices, the power dissipation of VLSI circuits is becoming a critical concern. In this paper, we propose a post–mapping technique that can reduce the power dissipation by performing gate resizing. This technique consists of replacing some gates of the circuit with devices in a complete cell library having smaller area and, therefore, smaller gate capacitance with lower power consumption. The slack time of each gate in the circuit is first computed to determine the set of gates that can be down–sized. A global optimization procedure based on integer linear programming and the simplex method is then applied to determine the best overall gate resizing solution. Experimental results on benchmark circuits have shown a power reduction in the range from 2.8 to 27.9 % compared to circuits without resizing. The most relevant features of our technique are that it is applicable to large digital circuits and gives an optimal resizing solution in a short computation time (no more than 15.8 seconds).

1. Introduction

Due to the advance of integrated circuit technologies, it is now possible to integrate several millions of transistors into a small chip area with high performance. However, power consumption problem rises owing to the increased circuit density and speed. Higher power consumption may reduce circuit reliability, shorten the life time and thus require extra device to remove heat. Therefore, power consumption has emerged as an important optimization goal in the design of VLSI circuits.

Low power consumption can be targeted at various levels of the design process, i.e. at the system, architectural, logic and physical levels [1]. Optimization at the logic level may occur during logic synthesis, structural netlist optimization, or gate resizing. Combinational logic synthesis is usually partitioned into two phases. First, a boolean network is optimized independently of the chosen target library [10][11]. Second, functions in the network are mapped to the library by using an efficient technology mapping algorithm [15][16]. This phase yields a logic netlist that can be further optimized by structural transformations [14]. Another possible solution to reduce power consumption at the logic level is to applied gate resizing, that represents an efficient optimization technique before placement and routing. This technique consists of replacing some gates on non–critical paths by devices in a gate library having smaller area and, therefore, smaller capacitive load. Given that the power dissipated by a gate is proportional to its load, reducing that load leads to a reduction of the power dissipated by the circuit as well as a reduction of the chip area.

In this paper, we address the problem of reducing the power consumption of a technology mapped circuit under timing constraints by applying gate resizing. The problem has been formulated as a descrete, global, constrained optimization problem, and has been solved by proposing a fast algorithm based on integer linear programming (ILP) and the simplex method. Inputs to this algorithm are a technology mapped circuit, the timing constraints, the switching activity on each node and a complete cell library. The output is a circuit with minimum power consumption that satisfies the given timing constraints. From a practical point of view, our algorithm works in two steps. In the first one, it computes the slack time of each gate in the circuit by using a backward traversal procedure. The slack of a gate is the amount of delay by which a gate delay may be increased without affecting the critical delay of the circuit. In the second step, it starts from the set of all gates in the circuit that can be resized (gates with a slack time greater than zero), and searches the global resizing solution that gives the highest gain in power reduction. The most relevant feature of our technique is that it can be applied on large circuits within a short computation time.

Gate resizing is a well known technique for delay–constrained power optimization, and several approaches

have already been published [1][2][4][12]. Some of them provide solutions that may not be optimal in terms of power reduction [1][12]. The others ([2][4]) consider the problem as a global optimization problem and always yield optimal gate resizing solutions. However, a common feature of these existing approaches is that they use iterative algorithms and, therefore, may not be able to handle very large circuits in a reasonable CPU time. A comparison between our approach and the existing methods will be performed in the fourth part of this paper.

From a general point of view, the main features of the gate resizing technique are that it does not change the topology of the circuit under optimization, and that circuits re–synthesized with this technique are guaranteed to be as fast as the original implementations, but smaller and less power–consuming. Another interesting aspect is that it allows to eliminate undesired spurious transitions in logic circuits. Spurious transitions account for between 10% and 40% of the switching activity in typical combinational logic circuits [8]. The well known solution to reduce spurious switching activity in a design is to complete path balancing [6]. As the aim in gate resizing is to delayed non critical paths such that all paths have finally the same delay, it is obvious that the power saved by this technique also comes from the decrease of spurious switching activity.

The rest of the paper is organized as follows. In the next section, we detail the calculation of the slack time on each gate, which is required to determine the set of gates that can be resized. Section 3 describes the global optimization procedure for low power, which is based on integer linear programming. In Section 4, we compare our approach with existing gate resizing techniques. Experimental results are presented in Section 4, and conclusion is given in Section 5.

2. Slack time calculation

This method is an input vector independent approach that quickly estimates the slack on each gate of a combinational circuit. The first step to compute the slack time on each gate is to calculate the *source delay SD* of each lead in the circuit by performing a simple topological delay analysis. The source delay of a lead is the maximum delay of a subpath from a primary input to the given lead. Next, the *required time* on primary outputs is initialized. The required time on a lead *j*, denoted as *RT(j)*, is the time at which the signal on lead *j* is required to be stable whatever the input vector applied to the circuit may be. The required time on primary ouputs is set to the value of the true delay of the circuit, and is next computed on each lead by using a backward traversal procedure and equation (1):

$$\forall\, j \in \{ \text{ inputs to } G \} \qquad RT(j) \;=\; RT(z) \;-\; \delta(G) \qquad (1)$$

where *j* is one of the inputs and *z* the output of gate G respectively ($\delta(G)$ is the propagation delay of gate G). In the case where line *j* is a fanout stem, the required time is evaluated from the required time on the fanout branches, as expressed in equation (2):

$$RT(j) \;=\; \min_k \{\, RT(j_k) \,\} \qquad (2)$$

where *k* ranges over all fanout branches j_k of stem *j*. After the required time has been computed on each line, the slack time is calculated. The slack time of a gate G, denoted as *slack(G)*, represents the difference between the required time and the source delay *SD(j)* on the output of the gate:

$$slack(G) \;=\; RT(j) \;-\; SD(j) \qquad (3)$$

According to the above definition, a circuit is safe, meaning it satisfies the given timing constraints, if for each gate, the slack time is greater than or equal to zero.

Example: Consider the example circuit shown in Figure 1 with the gate delays reported inside the gates. Having determined the source delays in a forward pass of the circuit, the required time on the primary outputs is initialized to the value of the longest structural path delay (7 time units) and the slack time on gates G_7, G_8 and G_9 is calculated. Next, the process continues until the required times on the primary inputs are calculated. In our example, only four gates have a slack time greater than zero. These non–zero slack times are reported on the circuit diagram.

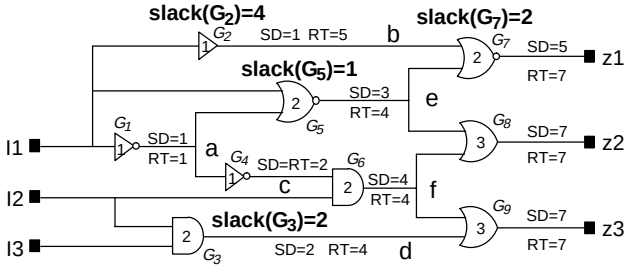

Figure 1: Calculation of the slack times

A drawback of this method for slack time calculation is that it may underestimate the slack time of a gate, because the path sensitizability is not taken into account. Although it is possible to solve this problem, this would be too CPU time consuming compared with the gain in accuracy of the slack values. Indeed, the calculation of the actual slack time on each gate would require to consider the longest path passing through the gate, and verify the existence of an input vector for this path. Analysing the path sensitizability also requires the use of a timing analysis that leads to an increase of the computation time [3]. As the technique we propose always provides right results (it does not overestimate the slack time of gates, and hence, the possible reduction in power consumption), it can be accepted as solution to our problem. Moreover, it is not the most significant contribution in this paper (see Section 3 for that).

3. Global optimization for low power

3.1 Power consumption model

There are three major sources of power dissipation in digital CMOS circuits: the charging and discharging of load

capacitances during output switchings (dynamic power), the short circuit current that flows during output transitions, and the leakage current. Since the last two sources are in the nanowatt range [17], the dominant source of power dissipation in a well–designed circuit is the dynamic power dissipation [1]. The average dynamic power consumption for a gate G in a synchronous CMOS circuit is therefore:

$$P_{avg}(G) = \frac{1}{2} \cdot f_{clk} \cdot C_{load}(G) \cdot V_{DD}^2 \cdot D(G) \quad (4)$$

where f_{clk} is the clock frequency, $C_{load}(G)$ is the load capacitance, V_{dd} is the supply voltage, and $D(G)$ is the transition density (the average number of transitions per clock cycle) at the output of gate G. The load capacitance of gate G can be expressed as follows [16]:

$$C_{load}(G) = \sum_{j \in fanout(G)} C_{gate}(j) + C_{wire} \quad (5)$$

where $C_{gate}(j)$ is the gate capacitance of j, and C_{wire} is the wiring capacitance of the output net. Since the gate capacitance is proportional to the gate area, the traditional approach for minimizing the power consumption has been to minimize the total gate area. However, since the power consumption also depends on the switching activities of the gates, a more effective solution is to minimize the total weighted switching activity in the circuit $\sum C_{load}(j).D(j)$.

3.2 Evaluation of the gain function

After the slack time on each gate has been calculated, the next step in our method is to select a limited number of gates for resizing since all gates in the circuit cannot be slowed down. First, only gates with a slack time greater than zero may be candidates for resizing. Secondly, two or more gates belonging to the same path cannot be down–sized simultaneously in order to satisfy the timing constraints. Of course, it is possible to resize several gates on the same path by sharing the amount of available slack on each gate, but slowing down one gate among all provides the same result when a complete cell library is available. For this assertion to be valid, we need to consider the transition density together with the slack time of gates. This point will be developed subsequently.

In order to select the gates to be slowed down, we have first to determine the sets of gates that can be resized simultaneously, and then to select the set that provides the highest power reduction (the best resizing solution). In accordance with our assumption that only one gate per path can be selected for resizing, gates that can be resized simultaneously are those that belong to disjoint paths in the circuit. Those sets of gates can be determined very easily. To determine the set of gates representing the best resizing solution, we need to use an evaluation or gain function that allows to evaluate a solution in terms of power reduction. This power reduction in our technique is due to the decrease in total load capacitance of the circuit. Since the reduction in load capacitance is directly linked to the increase in propagation delay of the resized gates, we define a gain function *Gain* in which the slack time of gates is taken as evaluator. Moreover, the power consumption also depends on the transition density of gate outputs. Therefore, we need to consider this value in the expression of the evaluation function. However, it is well known that the replacement of a gate by a slower template decreases the load capacitance of the fanin gates. For this reason, the gain of a set of gates has been defined according to the following expression:

$$Gain(set_i) = \sum_{G \in set_i} \left(slack(G) \cdot \sum_{g \in fanin(G)} D(g) \right) \quad (6)$$

where set_i is a set of gates that can be resized simultaneously and $Gain(set_i)$ is the gain of set_i. $D(g)$ is the transition density at the output of gate *g*, with g being a fanin gate of G.

As the aim in gate resizing is to minimize the total weighted switching activity in the circuit, a more accurate evaluation function would be a function based on the difference in capacitance of the gate to be resized. However, using this information still requires that the slack time on each gate has been determined. Although the use of such information requires additional computation time, we have made experiments with the following evaluation function:

$$Gain2(set_i) = \sum_{G \in set_i} \left(\sum_{g \in fanin(G)} C_{load}(g) \cdot D(g) \right) \quad (7)$$

The result of these experiments is that we have obtained nearly the same solution (the same set of gates to be resized) than that provided by the first evaluation function, but in a CPU time significantly increased. We have therefore retained the first function in the final algorithm.

Before calculating the gain values for each gate that can be down–sized, an important task is to determine the transition density (also called the switching activity) on each line of the circuit. This task is performed in a preprocessing step by using a simple and straightforward simulation–based estimation technique. Input patterns applied to the circuit are randomly generated, and a statistical mean estimation technique is used to decide when to stop. The main advantage of this technique is that it provides a way to quickly estimate the average number of transitions on each line of the circuit. Of course, this technique to determine the transition density could be replaced by an existing probabilistic technique [13].

3.3 A solution based on ILP

The solution we propose to determine the best set of simultaneously resizable gates is based on the integer linear programming (ILP). This technique is often selected in the general problem of allocating *limited resources* among *competing activities* in the best possible way [9]. Linear programming uses a mathematical model to describe the problem of concern. To formulate the mathematical model, let x_i represent a decision variable having the binary form:

$$x_i = \begin{cases} 1 & \text{if } \textit{decision i is yes} \\ 0 & \text{if } \textit{decision i is no} \end{cases} \quad (8)$$

The linear programming model is then to select the values for x_1, x_2, ..., x_n so as to maximize:

$$Z = c_1.\, x_1 + c_2.\, x_2 + \ldots + c_n.\, x_n \qquad (9)$$

subject to the restrictions:

$$\begin{aligned} a_{11}\, x_1 + a_{12}\, x_2 + \ldots + a_{1n}\, x_n &\le b_1 \\ a_{21}\, x_1 + a_{22}\, x_2 + \ldots + a_{2n}\, x_n &\le b_2 \\ \ldots & \\ a_{m1}\, x_1 + a_{m2}\, x_2 + \ldots + a_{mn}\, x_n &\le b_m \end{aligned} \qquad (10)$$

c_i represents the coefficient of x_i in the objective function, a_{ji} takes the value 1 if the variable x_i appears in constraint j and 0 otherwise, n is the number of decision variables, m the number of constraints.

In our problem, a decision variable x_i is associated to each resizable gate G_i of the circuit, and has the value 1 or 0 depending on whether the gate must be chosen to be resized or not. As no more than one gate per path can be selected for gate resizing, some decisions represent *mutually exclusive alternatives* such that only one decision among several must be selected. These mutually exclusive alternatives are called constraints in our problem, and have the form:

$$\begin{aligned} a_{11}\, x_{G_1} + a_{12}\, x_{G_2} + \ldots + a_{1n}\, x_{G_n} &\le 1 \\ a_{21}\, x_{G_1} + a_{22}\, x_{G_2} + \ldots + a_{2n}\, x_{G_n} &\le 1 \\ \ldots & \\ a_{m1}\, x_{G_1} + a_{m2}\, x_{G_2} + \ldots + a_{mn}\, x_{G_n} &\le 1 \end{aligned} \qquad (11)$$

where x_{Gi} is the decision variable corresponding to gate G_i, n the number of resizable gates and m the number of constraints. In the example circuit of figure 1, there is a constraint between gates G_2 and G_7, which belong to the same path. Similarly, another constraint exists between G_5 and G_7. The overall set of constraints is therefore:

$$x_{G_2} + x_{G_7} \le 1 \qquad x_{G_5} + x_{G_7} \le 1$$

As the best set of simultaneously resizable gates is the one having the highest gain value (thus leading to the highest power reduction), the objective function Z we have to maximize can be formulated as described below:

$$Z = Gain(G_1).\, x_{G_1} + Gain(G_2).\, x_{G_2} + \ldots + Gain(G_n).\, x_{G_n}$$

$$with \quad Gain(G_i) = slack(G_i) \cdot \sum_{g \in fanin(G_i)} D(g) \qquad (12)$$

Again consider the circuit of Figure 1, the problem is to select the values for x_{G2}, x_{G3}, x_{G5} and x_{G7} so as to maximize:

$$Z = Gain(G_2).\, x_{G_2} + Gain(G_3).\, x_{G_3} + Gain(G_5).\, x_{G_5} + Gain(G_7).\, x_{G_7} = 4.x_{G_2} + 2.x_{G_3} + 1.x_{G_5} + 2.x_{G_7}$$

$$\text{with the constraints} \quad \begin{aligned} x_{G_2} + x_{G_7} &\le 1 \\ x_{G_5} + x_{G_7} &\le 1 \end{aligned}$$

Remark: For the sake of simplicity in our example, we took $\Sigma D(g)$ equal to 1 for each $Gain(G_i)$ in the objective function. Results different from those given here for this example may therefore be obtained when one considers $\Sigma D(g) \neq 1$.

Since there is no constraint on the variable x_{G3}, a solution to maximize the objective function Z is to select x_{G3}=1. Then, we can see that the solution is to select x_{G2}=1, x_{G5}=1 and x_{G7}=0, producing the optimal value Z_{max}=7. Several techniques exist to solve ILP problems, such as the branch–and–bound technique, the simplex method, etc.. The simplex method has been used in this work.

4. Comparison with existing techniques

Several approaches have already been proposed for gate resizing. In [1], a heuristic approach is presented to deal with this problem. The gates of a circuit are processed from the primary outputs in a depth–first search manner. When a gate with a positive slack is found, the algorithm tries to replace it by a slower template extracted from the cell library. Once the gate is replaced, the new gate delay is updated, and the slacks for its fanin gates are re–computed. In [12], the power reduction algorithm proceeds in two phases. Phase one performs single gate resizing iteratively, until down–sizing a gate causes violation of the timing constraints. Phase two performs multiple gate resizing also iteratively, until no more improvement can be made.

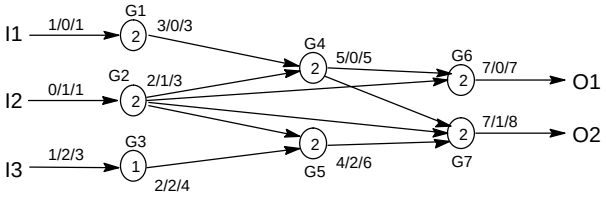

Figure 2: The corresponding graph of an example circuit

A common drawback of these two approaches is that they yield solutions that may not be optimal in terms of power reduction. This is due to the fact that down–sizing a gate at the early stage of the optimization process may prevent further power reduction [2][5]. For example, let us consider in Figure 2 the corresponding graph of an example circuit. The triplet shown next to each node *j* denotes *SD(j)/Slack(G)/RT(j)* where line *j* is the output of G. Now, consider Figure 3.a given below, which shows the result of applying the algorithm presented in [1] to the circuit graph of Figure 2. In this case, the delay of gate G5 and gate G7 increases by one unit each, thus leading to a reduction in power dissipation. However, we can see that it is not the optimal solution for gate resizing. Figure 3.b shows the result of using the approach in [2], where the total increase in delay units is 3 (the delay of gates G2, G3 and G7 is increased by one unit each) instead of 2 in the previous case. Figure 3.c shows the result of using our approach, that provides the same increase in delay units (the delay of gates G2 and G3 is increased by one and two units respectively). In the latter two cases, a better solution has been obtained without violating the timing constraints, proving that the approaches in [1] and [12] may not be optimal. Another drawback of the approach in [1] is that ADDs (Algebraic Decision Diagrams) are used to determine the slack time on each gate, thus preventing large circuits to be handled due to high memory requirements of the ADD data structure.

The most recent approaches dedicated to gate resizing ([2][4]) are efficient in the sense that they always yield globally optimal solutions to the gate resizing problem. Moreover, the approach in [4] may handle fairly large circuits in a reasonable CPU time in spite of the fact that it is based on an iterative algorithm (concerning the approach in [2], no result is given about the CPU time taken by the iterative process, and only small circuits have been experimented). However, results given in the last section of this paper demonstrate that our approach can provide globally optimal solution for large circuits in a CPU time that is five times lower than those given in [4]. The main reason for this result is that our approach is based on a non–iterative algorithm (in the case of a complete cell library) that does not need to compute the slack time and solve the ILP problem several times before obtaining a solution.

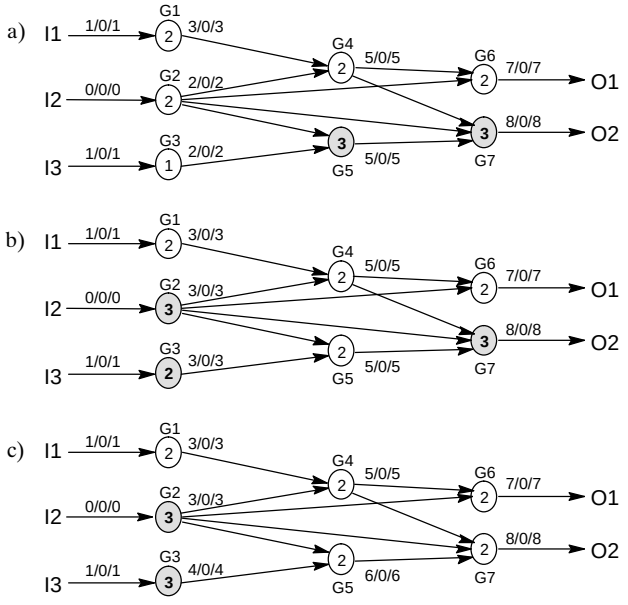

Figure 3: Comparison between gate resizing approaches

The algorithm described in the previous section, that always find a globally optimal solution, is based on the assumption that a complete cell library is available. However, this assumption is not always true in real designs, where the number of logically equivalent cells in a library may be limited. In such a case, a simple solution consists in applying iteratively the algorithm described in section 3 until there is no available slack. Although our algorithm still yields an optimal solution in this case, it is obvious that our approach is less attractive in such situation.

5. Experimental results

The complete algorithm has been implemented in C++ language on a SUNSPARC_5 workstation. Benchmarking process was performed on combinational circuits from the ISCAS'85 and MCNC'91 benchmark sets. Experiments performed on each circuit have been done with gate delays and load capacitances obtained after technology mapping from a standard cell library (library of ES2: "European Silicon Structures" company) [7]. Circuits were mapped with CADENCE tools in the 0.7µm standard digital CMOS technology. In the first step, the slack time of each gate has been computed using the method described in Section 2, with the delay constraint set to the length of the longest structural path in the circuit. The number of gates to be considered for down–sizing at the end of this step (gates with a non–zero slack time) is reported for each circuit in column 3 of Table 1. The number of gates chosen to be actually resized after the optimization procedure described in Section 3 is given in the fourth column of Table 1.

Circuit	# gates	# resizable gates	# gates actually resized	ΔPower (in %)	CPU time (in sec)
5xp1	159	156	78	23.5	0.249
c432	171	165	36	19.0	0.349
c499	218	56	21	21.8	0.316
rd73	277	273	165	27.9	0.516
c880	383	368	101	14.2	0.649
clip	401	92	55	6.8	0.433
c1355	564	523	64	3.5	1.783
b12	754	749	396	20.8	1.349
c1908	973	970	110	19.5	2.633
t481	1059	177	101	4.2	1.249
c2670	1211	1184	237	17.0	5.71
c3540	1705	1658	221	7.1	4.966
c5315	2351	2298	473	13.1	4.883
alu4	2381	2371	1309	26.5	6.099
c6288	2416	2339	117	2.8	15.866
apex2	3149	3141	1275	24.4	7.566
c7552	3624	3620	612	11.8	8.566

Table 1: Results of power reduction by gate resizing

The column ΔPower reports the percentage of power reduction obtained for each circuit. This percentage represents the ratio between the total weighted switching activity after and before resizing (see Equation (14), in which n is the total number of gates in the circuit, and i ranges over all the gates of the circuit). The last column in Table 1 reports the CPU time taken by the overall gate resizing process. For each circuit, this time includes: calculation of the slack time on each gate, determination of the best set of gates to be resized (from the simplex method), computation of the new parameters (C_{load}) on the down–sized gates, and calculation of the percentage of power reduction ΔPower(%). The computation of the transition density on each line is performed in a preprocessing step.

From an experimental point of view, the way to compute ΔPower has been the following. Before resizing, the values

of the wiring capacitance on each line and the gate capacitances are known from the standard cell library of ES2, as well as the actual delays on each gate. After resizing of one gate g, the load capacitances of the fanin gates of g are re–computed from the new value of the gate capacitance of g. This step is repeated after each gate resizing, such that we can finally obtained the weighted switching activity on the output of each down–sized gate. By computing the ratio between those values and the same quantity before resizing, we can obtain the power saved ΔP in each circuit.

$$\Delta P = \frac{P_{after\ resizing}}{P_{before\ resizing}} = \frac{\frac{1}{2} \cdot f_{clk} \cdot V_{dd}^2 \cdot \sum_{i=1}^{n} C_{load}'(i) \cdot D(i)}{\frac{1}{2} \cdot f_{clk} \cdot V_{dd}^2 \cdot \sum_{i=1}^{n} C_{load}(i) \cdot D(i)} \quad (13)$$

$$\Delta Power\ (\%) = 100 - \frac{\sum_{i=1}^{n} C_{load}'(i) \cdot D(i)}{\sum_{i=1}^{n} C_{load}(i) \cdot D(i)} \cdot 100 \quad (14)$$

From the results given in Table 1, we can see that about 15.52 % reduction in power consumption has been obtained on the average, with a maximum of 27.9 % for circuit rd73. These results demonstrate the effectiveness of the proposed technique, although they can not be compared with those presented in [1], [2], [4] and [12] (the cell library and the technology mapper are not the same). Another important point is that the CPU time required to find the gate resizing solution is very short for all circuits, and can be neglected in comparison to the total design time, thus increasing the interest of using our method as a postprocessing optimizer. The transition densities were computed with a statistical mean estimation technique as described in Section 3.2. For each circuit, one thousand of input patterns were randomly generated to compute the transition density on each line.

6. Conclusion

In this paper, a power reduction algorithm by gate resizing is presented. A slack time calculation algorithm is first proposed. A gain function that considers the fanin transition density of gates is then defined to guide the gate selection for resizing. A global optimization procedure for low power, which is based on ILP and the simplex method, is finally presented. Experimental results on benchmark circuits showed a power reduction in the range from 2.8 to 27.9 % compared to circuits without resizing. Compared with existing gate resizing techniques, our method is able to deal with real size circuits, and provides globally optimal solutions in a short computation time (less than 15.8s).

Acknowledgements

The authors would like to thank B. Rouzeyre for interesting discussions on combinatorial optimization, and G. Cathebras for his help on technology mapping of experimented circuits.

References

[1] R.I. Bahar, G.D. Hachtel, E. Macii and F. Somenzi, ***"A Symbolic Method to Reduce Power Consumption of Circuits Containing False Paths"***, IEEE International Conference on Computer–Aided Design, pp. 368–371, June 1994.

[1] A.P. Chandrakasan, S.Sheng and R.W. Brodersen, ***"Low–Power CMOS Digital Design"***, IEEE Journal of Solid–State Circuits, vol. 27, no. 4, pp. 473–484, April 1992.

[2] D.S. Chen and M. Sarrafzadeh, ***"An Exact Algorithm for Low Power Library–Specific gate Re–Sizing"***, ACM / IEEE Design Automation Conference, pp. 783–788, June 1996.

[3] H.C. Chen and D. Hung–Chang Du, ***"Path Sensitization in Critical Path Problem"***, IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 12, no. 2, February 1993.

[4] O. Coudert, ***"Gate Sizing: A General Purpose Optimization Approach"***, IEEE European Design & Test Conference, pp. 214–218, March 1996.

[5] O. Coudert, R. Haddad and S. Manne, ***"New Algorithms for Gate Sizing: A Comparative Study"***, ACM / IEEE Design Automation Conference, pp. 734–739, June 1996.

[6] S. Devadas and S. Malik, ***"A Survey of Optimization Techniques Targeting Low Power VLSI Circuits"***, ACM / IEEE Design Automation Conference, pp. 242–247, June 1995.

[7] European Silicon Structures ES2, process ECPD07, library databook

[8] A. Ghosh, S. Devadas, K. Keutzer and J. White, ***"Estimation of Average Switching Activity in Combinational and Sequential Circuits"***, ACM / IEEE Design Automation Conference, pp. 253–259, June 1992.

[9] F.S. Hillier and G.J. Lieberman, ***"Introduction to Operations Research"***, Holden–Day, Inc., Oakland, 1986.

[10] S. Iman and M. Pedram, ***"Multi–level Network Optimization for Low Power"***, ACM / IEEE International Conference on Computer–Aided Design, pp. 372–377, November 1994.

[11] S. Iman and M. Pedram, ***"Logic Extraction and Factorisation for Low Power"***, ACM / IEEE Design Automation Conference, pp. 248–253, June 1995.

[12] H.R. Lin and T. Hwang, ***"Power Reduction by Gate Sizing with Path–Oriented Slack Calculation"***, IEEE ASP–DAC'95 / CHDL'95 / VLSI'95, pp. 7–12, June 1995.

[13] F.N. Najm, ***"A Survey of Power Estimation Techniques in VLSI Circuits"***, IEEE Transactions on VLSI systems, vol. 2, no. 4, pp. 446–455, December 1994.

[14] B. Rohfleisch, A. Kolbl and B. Wurth, ***"Reducing Power Dissipation after Technology Mapping by Structural Transformations"***, ACM / IEEE Design Automation Conference, p. 789–794, June 1996.

[15] V. Tiwari, P. Ashar and S. Malik, ***"Technology Mapping for Low–Power"***, ACM / IEEE Design Automation Conference, pp. 74–79, June 1993.

[16] C.Y. Tsui, M. Pedram and A.M. Despain, ***"Power Efficient Technology Decomposition and Mapping Under an Extended Power Consumption Model"***, IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 13, no. 9, September 1994.

[17] N. Weste and K. Eshraghian, ***"Principles of CMOS VLSI Design, A Systems Perspective"***, Reading MA: Addison–Wesley Publishing Company, 1988.