Low Power Technology Mapping for LUT based FPGA – A Genetic Algorithm Approach

Rohit Pandey and Santanu Chattopadhyay
Dept. of Computer Sc. & Engg.
Indian Institute of Technology, Guwahati
E-mail: santanu@.iitg.ernet.in

Abstract

In this paper we consider the problem of LookUp Table(LUT) based FPGA technology mapping for power minimization in combinational circuits. The problem has been previously proven to be NP-complete and here we present an efficient Genetic Algorithm solution for it. Considering that the connection switches posses large resistance and capacitance in LUT based FPGA, the fitness of the chromosome is selected based on its ability to reduce the transition probability on "visible" edges of mapped logic circuits by hiding the paths with high transition activity in the "invisible" edges. Meanwhile, the number of LUT is also kept small.

1 Introduction

Recently there has been a surge of interest in low power devices and design techniques mainly due to the booming of personal wireless communication equipments. The new applications require high speed, yet low power consumption. In addition, power dissipation in chips rises with increased integration density and circuit speed. Low power design is thus essential to lower the packaging and cooling costs and prolong the life of the ICs. Because of its short design cycle and low manufacturing cost, FPGA technology offers the best solution to time-to-market pressure.

The problem of finding a technology mapping solution of minimum estimated power consumption is indeed NP-hard[6], hence Genetic Algorithm can provide an efficient solution. *Covering* is one of the most crucial steps towards technology mapping in FPGA. The performance of the synthesised circuit, in terms of its area, delay and power requirement depends to a great extent on this step. Traditionally, technology mapping tools have been targeted towards area efficient designs[6]. However recently a lot of work has been carried out on power minimised designs.

Previously work has been done to reduce power consumption by optimization at different phases and level of design. Tsui *et al.* [5] proposed a NAND tree decomposition approach in which postorder and preorder traversals of the tree is done to reduce power. Their method gains 21% transition reduction on the penalty of 12.6% increase of the area. Tiwari et al. [4] also proposed their power-saving technology mapping which utilized DAG covering. The transition reduction is about 10% while the penalty of increase of chip area is about 12%. None of these methods are suitable for *Lookup table (LUT)* based FPGAs. Hwang *et al.* [1] re-synthesize LUT based FPGA circuits for low power design after technology mapping, placement and routing are performed. Kumthekar *et al.* [2] , targets power optimization through in-place re-programming of one or more configurable logic blocks(CLBs) used in the circuit implementation.

No matter which programming technology, e.g., SRAM-based cells or anti fuses, is used for the interconnection in FPGAs, the connection switch can be deemed as an RC delay circuit. The range of R is from hundreds of ohms to 4 KΩ , while the C is from 10 to 20 ff according to [3]. These switches certainly consume a great portion of power. In contrast, the power resulted from one transition inside the LUTs is relatively much smaller, since that one transition inside the LUT only causes one RAM output bit to change states. Hence, reducing power dissipation can be achieved by hiding the edges of the original circuit with high transition probability inside the LUTs while exposing the edges with low transition probability, if necessary, outside the LUTs. Wang *et al* [3] proposed an algorithm which minimized power at the decomposition phase of *technology mapping*. The algorithm work similar to *bin-packing algorithm* [6] used previously for reducing the number of CLBs, and is based on above concept of hiding of high transitions. There is power reduction of around 10% compared to result obtained after bin-packing algorithm. As shown in [6] better result in term of area and delay can be obtained by applying better covering algorithm. So in our work

Proceedings of the 16th International Conference on VLSI Design (VLSI'03)
1063-9667/03 $17.00 © 2003 **IEEE**

we have applied the above hiding concept while doing covering.

In this paper, we have presented a genetic algorithm based approach for solving the binate covering problem during covering phase targeting low power. The power consumed by these circuits has been compared with the circuits resulting from mapping by CAD tool SIS[7]. Experimentation with a large number of benchmark circuits shows that the results produced by GA is, on average 25.51% better than that by the SIS. The number of CLBs is also kept minimal. Further, the inherent parallelism of GA makes it an ideal candidate for solving the problem in a multiprocessor environment.

1.1 Power Estimation Model

The amount of energy dissipated by a CMOS gate is dominated by charging and discharging in capacitive loads whenever a transition occurs at the gate output. In this paper, we assume a *zero-delay* power estimation model, i.e., we don't consider the contribution of glitches to the power consumption, as at the logic level, glitching, which is caused by unbalanced path lengths are difficult to model accurately. Under this zero-delay assumption, the power consumed by a CMOS circuit is given by

$P_{circuit} = \frac{1}{2} V_{dd}^2 \cdot f. \sum_{\forall i} C(i) \cdot E(i)$.

where V_{dd} is the supply voltage, and f is the clock frequency, $C(i)$ denotes the total capacitive load driven by signal i, and $E(i)$ is the *transition probability* of signal i, that is, the probability that signal i makes a $0 \rightarrow 1$ or a $1 \rightarrow 0$ transition from one clock to the next.

At the logic level, the supply voltage V_{dd} and f is fixed. Therefore power reduction at the logic level is achieved by reducing $\sum_i C(i) \cdot E(i)$. As in [8], we compute the transition probability of a signal assuming temporal independence of the primary inputs. Thus, the transition probability of a signal s is $E(s) = 2 \cdot p(s) \cdot (1 - p(s))$, where $p(s)$ is the signal probability of s.

2 FPGA synthesis

Many tools for LUT-based FPGA technology mapping have been proposed[6]. Among them, SIS is comprehensive synthesis tool [7]. FPGA technology mapping in SIS consists of two main steps: *decomposition* and *reduction*. Decomposition step makes an infeasible network feasible by applying various algorithms like cube-packing, roth and karp decomposition, kernel extraction, etc. In the reduction step, we try to optimize the feasible network obtained from decomposition step by solving binate covering problem which is NP-complete. SIS solves this problem by applying various heuristics. Genetic algorithm[9], which simulates the natural process of evolution, has been proposed as an efficient paradigm for solving hard optimization problem. It has been successfully employed for solving many problems in areas such as logic synthesis, VLSI placement, state assignment, etc. Zhuang *et al.* [10] has succesfully applied GA for binate covering problem targeting area minimization. In our work we have formulated GA for binate covering problem targeting low power.

2.1 Binate covering formulation

Definition 1: **Covering** operation can be stated as follows. Given an m-feasible Boolean network η, iteratively collapse nodes into their fanout such that the resulting network η' is m-feasible and the power consumption by η' is minimum.

Definition 2: Given a graph $G = (V, E)$, and $U \in V$, the induced subgraph of G on U is (U, E_1), where $E_1 = \{(u_1, u_2) \text{ — } u_1, u_2 \in U \text{ and } (u_1, u_2) \in E\}$. In other words, (U, E_1) is G restricted to U.

Definition 3: A **supernode** corresponding to a node n of the network η is an induced directed subgraph ς of η, with a single root n such that ς does not contain any primary input node of η.

Definition 4: A supernode ς is m-feasible if its support cardinality is at most m. After decomposition, an m-feasible network is obtained, which can be implemented straightway on the FPGAs by mapping each node to a CLB. This strategy, however, yields sub-optimal results.

Example 1: consider the following optimised network η, with one primary output f, eleven primary inputs, and four internal nodes. Let m be 5.

$$\begin{aligned} f &= x + y + z \\ x &= degh \\ y &= abc \\ z &= ijkl \end{aligned}$$

Now η can be mapped onto target LUT architecture using 4 CLB and the power consumption is proportional to E(f)+E(x)+E(y)+E(z). However if we collapse y into f, η can be realize using 3 CLB and power consumed is proportional to E(f) + E(x) + E(z). Hence, covering routine is very important to reduce power consumption.

In our synthesis scheme, we first obtain an m-feasible network η by SIS for an input circuit. Then

corresponding to each node in η we generate all supernodes by repeatedly applying the Maxflow algorithm as given in [6]. Having generated the set of all m-feasible supernode, the *binate covering matrix* M is generated as follows[6,10].

For each supernode ς_i, there is a column. The rows of binate covering matrix are basically for the nodes of the network (there are additional rows to take care of the binate covering constraints). $M(i,j) = 1$ if the node corresponding to row j is contained in the supernode ς_i. Since every supernode can be implemented by one CLB, the task of the covering is to find a subset τ of supernodes that satisfies the following constraints and minimizes the cost function.

- **Output constraints:** for each primary output PO, at least one supernode with PO as the root must be chosen.
- **Implication constraints:** if a supernode ς is in τ, each input of τ should be either a primary input or an output of some supernode in τ.

An exact algorithm can only be applied for small circuits (normally having less than 30 nodes) [10]. On the otherhand, heuristic algorithms normally proceed step by step. In a particular step, it selects one of the possible supernodes showing good promise in terms of cost function. It then deletes the column corresponding to this supernode and all the rows corresponding to the nodes covered by the supernode. The process continues till there exists at least one node uncovered. However, for large circuits there may be hundreds of columns with the same cost function. Hence, such arbitrary selection may not provide a good solution. To circumvent this, we have formulated a genetic algorithmic solution to the problem of minimizing power consumption for LUT based FPGA mapped circuits.

2.2 Weight of supernode

Weight of supernode has to be calculated for selection of supernode for each iteration of covering. For area minimizing algorithm[10] weight of each supernode was the number of node it contain. However, for the power minimization case, we have used the sum of switching probabilities E that it can absorb as the cost function. It may be noted here that a supernode will be realized by a CLB. Hence, it absorbs switchings of all non-output nodes within it.

$$\text{weight}(\varsigma) = \sum_{\forall i} E(i),$$

where, $i \in \{\ x | x \in \varsigma \text{ and } x \notin \mathit{output}(\varsigma)\ \}$, $\mathit{output}(\varsigma)$ gives the output nodes of ς.

3 Genetic algorithm problem formulation

Genetic algorithm[9] are stochastic optimisation search algorithms based on the mechanics of natural selections and natural genetics. The algorithm starts with an initial population usually consisting of a set of randomly generated solutions. Depending upon a reproductive plan, the chromosomes undergo evolution for a number of generations. The reproductive plan conists of applying genetic operators, the most common being crossover and mutation. Depending upon the selection policy and fitness values, the set of chromosomes for the next generation are selected. This evolution process is continued for a number of generations. Depending upon some terminating criteria(which may be the maximum number of generation the GA is run, or the maximum number of successive generations without cost improvement), the algorithm terminates. The best solution at that generation is accepted as the solution produced by the GA.

3.1 Problem representation

To apply the Genetic Algorithm to the binate covering problem, we first generate N initial chromosomes, S_1, S_2, ... , S_N, each containing a set of P random numbers, which ranges from 1 to P. The value of P is set to limit the search space. It actually limits the maximum number of supernodes that we consider at any point of time. It is the sum of following two quantities.

- 20% of the number of supernodes having maximum absorbing power, that is weight, with the maximum being 10 (if the number exceeds 10) and
- 10% of the number of supernodes having maximum number of nodes, with the maximum being 5 (if the number exceeds 5).

Hence the maximum value of P that we have used for experimentation is 15. Each chromosome S_i is applied to the binate covering column selection procedure using the following algorithm.

1. For the jth selection step, find all k columns containing the maximum weight and having maximum node count, that is, those supernodes absorbing maximum switchings and also those supernodes covering maximum number of nodes.
2. Relabel these columns as $C_0, C_1, \ldots, C_{k-1}$, and select the column C_v to provide covering, where, $v = S[P \bmod j] \bmod k$.

3. Delete column C_v and all the rows covered by the 1's in column C_v from the matrix M.

4. Repeat steps 1 to 3 until all nodes are covered.

Example 2: Let the ith chromosome be S_i = (5,3,3,1,2). Here P is assumed to be 5. First of all, we find the supernodes having the maximum weight. Let there be three such nodes – s_0, s_1, and s_2. Then we find the supernode covering maximum number of nodes. Let there be only one such node s_3. So, we have $P = 5$, $j = 1$ (this is the first selection step), $k = 4$ (since there are four supernodes to consider, s_0, s_1, s_2, and s_3). So, $S_i[P \bmod j] \bmod k = S_i[5 \bmod 1] \bmod 4 = S_i[0] \bmod 4 = 5 \bmod 4 = 1$. So, the supernode s_1 is selected. The column corresponding to it and the rows covered by it are removed from the matrix M. With the updated matrix, we proceed to the next iteration. Now, $j = 2$. We again find the candidate supernodes to be considered at this step. Let that be 5. So, $k = 5$. Applying the formula given in *Step 2* of the algorithm, we again select a supernode in the cover set. The process continues till all the nodes are covered.

3.2 Crossover operator

Our GA formulation has been biased towards selecting the chromosomes with better fitness to participate in crossover. For this purpose, the whole population is sorted according to their fitness values. A certain percentage of the population with better fitness values is defined to be the 'best class'. To select a chromosome participating in crossover, first a uniform random number between 0 and 1 is generated. If the number is greater than 0.5, a chromosome from the 'best class' is selected randomly. Otherwise a chromosome from the entire population is chosen. This approach of selecting more fit chromosomes to participate in crossover leads to the generation of better offspring as compared to the truly random one.

3.3 Mutation operator

Mutation is very important operator as far as bringing variety into the population is concerned. As the population size is finite, the crossover operator alone cannot bring enough variation to the population. Thus the solution quality and rate of convergence suffers. The mutation operator brings more effective variations into the chromosomes, introducing newer search options.

In our case, the mutation operator first selects a chromosome S_i from the population randomly. It then modifies the value at position p_1 which it selects randomly with a random number in range 1 to P.

3.4 Evaluation of fitness

Let T be the set of supernodes obtained after applying chromosome C for covering. The fitness of C is inversely proportional to power consumed by the set T which is given by $\frac{1}{\sum_{\forall i} Fanout(i) \cdot E(i)}$

where, i corresponds to the set of variables associated with supernodes in T, *Fanout(i)* is number of fanout for variable i and *E(i)* is the transition probability for variable i.

4 Experimental Result

In this section we present the results of the GA based power optimization algorithm. The algorithm has been applied on a good number of *LGSynth91* combinational benchmark combinational circuits. *Table 1* details the results for each of the benchmarks. For the sake of comparison we have also applied the covering algorithm *xl_cover* of SIS [7] over the circuits. The total number of CLBs required and the cumulative switching activity (that is, transition probability multiplied by fanout for each of the CLBs) have been noted down. For the GA algorithm proposed in this paper, similar values are noted under the column for GA. The parameters used for GA are as follows:

Population size: 90
Stopping criteria: No improvement in last 8 generations
Mutation rate: 0.2

The column *generation* notes the number of generations required to arrive at the reported results. The program has been implemented on a *Pentium III* machine with 550 MHz clock, 64 MB RAM. The column *CPU time* notes the actual CPU time required in seconds to reach these solutions. As shown in Table 1, on an average, the scheme results in 25.51% reduction in cumulative switching activity over SIS. Moreover, for circuits like *Maskmx* reduction is as high as 46%. without any significant increase in the area. It may be noted that the scheme proposed in [3] results in only about 10% improvement over SIS.

5 Conclusion

In this paper we have presented a GA based formulation for solving the binate covering problem for technology mapping in LUT based FPGA targeting low power. On average, the scheme results in 25.51% reduction, and on Maskmx circuit it shows 46% reduction in power requirement over SIS without any significant increase in the area.

Table 1: Benchmark results on switching activity reduction

Circuit	SIS		GA			
	# CLB	Cum. switching	# CLB	Cum. switching	# Gen.	CPU time(sec)
Misex2	48	3.59	44	2.85	4	2.0
Pmoren	9	1.64	9	1.28	12	2.3
Sao2	97	8.36	93	7.24	2	3.2
Mspset	19	3.04	17	1.7	26	4.8
Con1	6	1.53	6	1.19	2	0.8
5xp1	11	2.24	10	1.71	7	2.9
Maskmx	41	16.76	36	9.01	9	4.3
Needs	25	5.85	20	5.14	15	3.6
Bw	17	3.84	15	2.69	4	2.3
Rd53	16	4.32	15	3.05	35	5.4
Z4ml	22	6.98	18	5.56	21	4.9
Alu4	212	14.32	209	11.31	7	9.2
C5315	366	16.86	361	13.80	10	15.8
Des	984	37.2	977	30.12	6	38.0
C6288	485	20.3	478	16.15	9	19.2
C7552	436	21.6	429	16.9	13	28.3

References

[1] Jan-Min Hwang, Feng-Yi Chiang, and TingTing Hwang. A Re-engineering Approach to Low Power FPGA Design Using SPDF. DAC-1998.

[2] Balakrishna Kumthekar, Luca Benini, Enrico Macii, and Fabio Somenzi. In-Place Power Optimization for LUT-Based FPGAs. DAC-98.

[3] Chau-Chin Wang, and Cheng-Pin Kwan. An Analysis of Low Power Technology Mapping by Hiding High-Transition Paths in Invisible Edges for LUT-Based FPGAs. 1997 IEEE International Symposium on Circuits and System. pp. 1536-1539. 1997.

[4] V. Tiwari, P. Ashar, and S. Malik. Technology mapping for low power. 30th ACM/IEEE Design Automation Conference. pp. 74-79 . 1993.

[5] C-Y. Tsui, M. Pedram, and A. D. Despain. Technology decomposition and mapping targeting low power dissipation. 30th ACM/IEEE Design Automation Conference. pp. 68-73. 1993.

[6] R. Murgai, R. Brayton, and A Sangiovanni-Vincentelli. Logic Synthesis For Field Programmable Gate Arrays. Kluwer academic publication. 1995.

[7] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, and R. K. Brayton. SIS: A system for sequential circuit synthesis. Technical report UCB/ERL M92/41, University of California, Berkeley, May 1992.

[8] F. N. Najm. A Survey of Power Estimation Techniques in VLSI circuits. IEEE Trans. VLSI System. Vol. 2, no. 4, pp. 446-455, January 1995.

[9] D. E. Goldberg. Genetic algorithm in search, optimization, and machine learning. Addison-Wesley Publishing Company. 1989.

[10] Nan Zhuang and Peter Y. K. Cheung. Using Genetic Algorithm in Binate Covering for LUT-based FPGA Technology Mapping. URL: citeseer.nj.nec.com/50251.html.

[11] R. Murgai, N.Shenoy, R.K. Brayton, and Sangiovanni-Vincentelli. Improved Logic Synthesis Algorithm for Table Look Up Architectures. Proc. IEEE International Conference on Computer Aided Design, 1991. pp 564-567.

[12] R.J. Francis, J. Rose, K. Chung. Chortle: A Technology Mapping Program for Lookup Table based Field Programmable Gate Arrays. DAC-1990. pp. 613-619.

[13] R.J. Francis, J. Rose, and Z. Vranesic. Chortle-crf: Fast Technology Mapping for Lookup Table based FPGAs. DAC-1991. pp. 227-233.

[14] T.T. Hwang, R.M. Owens, M.J. Irwin, and K.H. Wang. Logic Synthesis for Field Programmable Gate Arrays. IEEE Trans. on CAD, 1994, 13(10). pp. 1280-1286.

[15] J. Cong and Y. Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup Table based FPGA Design. IEEE Trans. on CAD, 1994 13(1), pp. 1-12.