Switch-Level Technology Mapping and Modeling

Jordi Riera, Josep Velasco, Lluís Ribas, Jordi Carrabina

Unitat de Microelectrònica - Dpt. Informàtica, Edifici C

Universitat Autònoma de Barcelona

E-08193 Bellaterra (Spain)

Phone +34.3.581.10.78, Fax +34.3.581.30.33

mail. riera@cnm.es, Web. http://www.cnm.es/~riera

Abstract

As circuit specifications are growing in complexity, manual optimization methods for performance improvement become clearly insufficient, and should be replaced by automatic optimization CAD systems. In the logic synthesis field, performance restrictions have been traditionally taken into account by using a library of functions with many versions, each one of them having different performance trade-off. To attain good trade-off between the design parameters, big cell libraries are required. So, its development and maintenance costs are ever increasing. Our approach introduces performance-driven switch-level logic synthesis, closely tied to physical synthesis. This library-free approach, based on *on-the-fly* gate generation, allows using simplified performance models during logic synthesis. The final circuit performance is fine tuned by the conforming module generator.

1. Introduction

The traditional logic synthesis process [1,2] makes a clear separation between technology independent minimization/optimization and technology dependent mapping. The minimization/optimization process aims to reduce the circuit complexity by both sharing common logic and eliminating redundant logic. The resulting circuit is then given to the technology mapping tool, which actually produces the final circuit netlist.

The technology mapping step normally relies on a given library of pre-characterized cells. The mapping process then consists of choosing a set of gates which cover the circuit while minimizing some cost function. To allow real area-power-delay trade-off in the final circuit implementation, the cell library must contain different functions and many versions for each function (with distinct Area-Power-Delay trade-off) [3]. This heavily increases the development and maintenance costs of such cell libraries. So, to reduce these, only a limited number of logic functions and versions would be present on a cell library.

On the other hand, the use of performance-driven module generators [4,5,6,7,8] avoids the need for cell libraries, giving more flexibility to logic synthesis tools [9,10]. Particularly, technology mapping is not anymore restricted to a set of functions and versions (library cells). For any function, with any possible trade-off between design objectives, a switch-level netlist would be generated "on-the-fly" if required to meet the circuit specifications.

This kind of logic synthesis, which relies on module generators, is normally referred as switch-level logic synthesis, and will be discussed in the section 2. Next section, 3, briefly reviews the on-the-fly gate generation characteristic of switch-level synthesis. Performance models are covered in section 4. Section 5 compares library-based with switch-level logic synthesis, when module generators are targeted. Some interesting results are provided. Finally, conclusions to this work are remarked in section 6.

2. Switch Level Logic Synthesis

It is clear that performance-driven module generators (PDMG’s) are able to provide a wide range of design objective trade-off, because of their performance-driven placement, sizing and routing steps. However, PDMGs are constrained by their input circuit netlist, which should be provided at the switch level by a logic synthesis tool. As traditional logic synthesis tools are library-based, they not perform very well at switch-level [10]. A big or huge cell library should be first defined and pre-characterized, then managed by the logic synthesis tools. Furthermore, traditional technology mapping does not take into account the sizing ability of the PDMG. The cost of managing elaborate timing information may be wasted, because soon afterwards the PDMG sizing process may invalidate this timing information. Effectively, transistor sizing occurs when placement is already done. Detailed wiring estimation is possible, allowing accurate timing evaluation which can induce transistor re-sizing, so invalidating the timing information previously used during mapping. Even worse, in a re-synthesis loop, this kind of non-consistent timing information between logic synthesis and module generation can result on non-convergent cycles.

As deduced from the above reasoning, there is a need for specific switch-level synthesis tools, which should not rely anymore on cell libraries. In fact, the switch-level logic synthesis flow is very similar to the traditional one: a minimization/optimization step followed by a technology mapping. Even the mapping objectives are similar (see figure 1): to cover the circuit while minimizing some cost function. The main differences between switch-level logic synthesis tools and the classical library-based ones are two: the capability of “on-the-fly” gate generation for any possible function and the simplified models for performance evaluation.

3. On-the-fly gate generation

Instead of trying to match each function on the circuit against all possible cells in a library, switch-level technology mapping relies on efficient generation of one-stage static CMOS gates to implement these functions. Although standard factorization [11] can be used to do that, we would rather prefer to directly generate the gates from their internal representation managed by the synthesis tools. This will allow us to exploit structural information to predict performance even in the early stages of logic synthesis. Other authors propose different kind of BDD-like structures [12], but they lack from interesting BDD properties and manipulation algorithms.

In our approach, a correct, static, single-stage, CMOS complex gate implementation of a monotone decreasing function can be obtained by direct transistor association to the edges of *any* ROBDD which represent the function [9,10]. The proposed method can be easily extended to cope with non-binate functions. Binate Boolean functions themselves, however, need to be implemented using input/output inversion and traditional factorization algorithms.

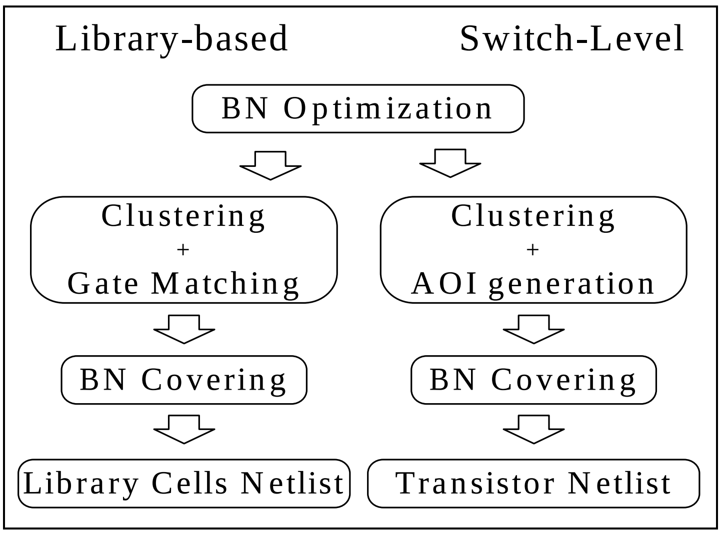

Figure 1. Classical vs. Switch-Level Mapping.

4. Performance evaluation

Switch-level logic synthesis allows simplified models for performance evaluation. Detailed evaluation is not required because the module generator will adjust performance afterwards, mainly during transistor sizing. Therefore, good accuracy is not critical; the important point is to provide good *correlation* with the performance models used by the module generator. The performance evaluation models applied during switch-level logic synthesis may be quite technology independent, as long as they are well correlated with the cost functions used by their corresponding module generator.

It is worth to say that this simplified performance evaluation models can be used in the early stages of logic synthesis, during minimization/optimization, if gates are directly generated from the BDDs using the method analyzed in the previous section. The BDD structural information is enough to give the correct parameters to the models: switching activity can be directly calculated from BDDs and the BDD edges are related to transistors, so area and timing information can be obtained.

4.1 Area Model

During the technology independent phases of traditional logic synthesis, area has been measured by counting the literals appearing on the circuit functions. It has been shown experimentally by many authors that this estimator is well correlated with the final area of the circuit, because literals are related to the number of transistors required to implement these functions [1]. Even better, switch-level logic synthesis may use directly the number of transistors. This information is known during technology mapping and can be estimated in the early stages of logic synthesis [10]. So, area can be evaluated in a quasi-technology independent way during switch-level logic synthesis.

4.2 Timing Model

The simplified gate delay model used in library-based logic synthesis may follow an equation similar to:

$$t^{i}_{delay} = t^{i}_{intrinsec} + \Delta t^{i} \cdot C^{i}_{load} \tag{1}$$

where $t^{i}_{\mathrm{intrinsec}}$ and Δt^{i} are constant parameters characteristic of the i^{th} gate and C^{i}_{load} is the driven external capacitance [13]. But this scenario changes radically in switch-level logic synthesis. Effectively, in library-based logic synthesis, gates characteristics are fixed *a-priori*. By contrast, gates are generated on-the-fly during switch-level logic synthesis. So, its final characteristics are not established until transistor sizing is done by the PDMG. During technology mapping C^{i}_{load} is unknown and, moreover, $t^{i}_{\mathrm{intrinsec}}$ and Δt^{i} can not be considered anymore as constant values. Transistor sizing can speed up a cell trying to compensate variations on C^{i}_{load} while decreasing $t^{i}_{\mathrm{intrinsec}}$ and Δt^{i}. Bus this can be done only up to a given lower limit, which is primarily fixed by the number of series transistors and also by internal capacitances and technology related performance constraints [13]. This may be expressed, for the i^{th} gate, using the following basic timing equation:

$$t^{i}_{min\_delay} = K \cdot max(T^{i}_{serial}) + k^{i}_{min} \tag{2}$$

where K is a technology dependent constant, related to the transistor's timing; T^{i}_{serial} is the maximum number of serial transistors (P-type for rise, N-type for fall transitions); finally, k^{i}_{min} represents the minimum delay due to parasitic capacitances on internal gate nodes and it is related to the gate complexity. So, using equation (2), propagation delay can be computed for each circuit gate.

4.3 *Power Dissipation model*

The switching activity, associated to power consumption, may be derived from the BDDs which represent the circuit functions [14]. This activity can be reduced mainly during circuit optimization/minimization, before technology mapping, because major transformations are performed on the circuit structure, so giving more chances to reduce its activity.

By contrast, no big power savings can be achieved by simply trying to reduce the switching activity during technology mapping, because there is no much scope for big circuit transformations. However, some authors claim for power savings when moving nodes with high switching activities to internal gate nodes [15]. These savings come from the fact that internal gate capacitances are usually smaller than external ones. Although this is true, most reported works rely on power models which does not take into account the power dissipation due to internal nodes. So, the reported gains should be taken with care.

Another important source for power consumption comes from glitch activity [16]. This is very dependent on the delay model being used [17,18], and so it is difficult to handle during switch-level technology mapping. Moreover, slight timing variations can cause dramatic increases/decreases in power dissipation [19]. We propose a different kind of measure, the *glitch sensitivity*. A glitch could be only generated in a gate if there is a significant difference between the arrival times of its inputs. The maximum difference can be easily calculated and will measure the possibility of a glitch being generated by the gate.

5. Cell-library vs. Switch-Level Technology Mapping

In this section we shall show the advantages of using switch-level logic synthesis tools instead of library-based tools, when targeting module generators. Since critical differences appear in the technology mapping step, we shall compare only results at this level. Benchmark circuits are minimized/optimized by SIS, an evolution of MIS [2]. The optimized circuits are then both fed to our switch level mapping tool, *synthetic*, and also to the library-based SIS mapping tool using the cost functions described in section 4. Only area and timing results are compared. Unfortunately, power comparison was not possible because SIS does not allowed it.

Table 1 summarizes the results obtained when the primary optimization criteria was the area and timing was secondary. As shown, the area saved by *synthetic* is only marginal. This result was expected and confirms that the technology independent minimization/optimization of the circuits is effective. But the flexibility of the switch-level mapping tool is used to improve the timing results by more than 16% compared to the library-based approach.

When the design objectives are time and area in this order, *synthetic* reduces the delay by 17% while saving 6% area with respect to SIS (see details in [9]). Consequently, specific switch-level logic synthesis tools take full advantage of a design flow with performance-driven module generators.

6. Conclusions

In this paper we have presented the main characteristics of a switch-level logic synthesis methodology. We have exposed the simplified performance models to be used during the logic synthesis process. These models are quasi-independent, allowing their use during the very first stages of logic synthesis. Moreover, if our on-the-fly gate generation method is applied, the information required by the performance models can be easily extracted from the BDDs representing the circuit functions.

Also, library-based and switch-level logic synthesis methodologies targeting module generators have been compared. Results show that the major flexibility

Table 1. *Library-based (SIS) vs. synthetic comparison.*

	SIS - Area			Synthetic - Area + timing		
Circuit	Area	Rise	Fall	Area	Rise	Fall
cm82a	64	19	14	68	12	9
cm151a	84	19,5	20,5	66	15,5	14
parity	180	24	20	180	24	20
cmb	150	15	21	164	40,5	35
cm163a	118	36,5	39	134	16,5	16,5
cm162a	120	20,5	23,5	130	17,5	20,5
cm150a	132	19,5	18	110	16,5	21,5
cm85a	140	26,5	18,5	116	22,5	25
cu	158	28,5	19	176	23	18,5
pm1	126	23	24	130	17	10,5
pcler8	268	29,5	32,5	246	20	19,5
cc	182	18,5	15,5	182	13	9,5
Total	1722	296,5		1702	248,5	
synth. Vs SIS				-1,16%	-16,19%	

of on-the-fly gate generation is exploited by our switch-level synthesis tool, *synthetic*. Even when the main objective is already close to the optimum, this extra flexibility is used to optimize secondary objectives.

Finally, we can conclude that switch-level logic synthesis is really needed to take full advantage of the flexibility introduced by a design flow including a performance-driven module generator. Currently, we are developing a PDMG to be used in this design methodology. Preliminary results are encouraging [4].

7. Acknowledgments

We would gratefully acknowledge the collaboration, discussion and support of Andre Reis, Daniel Auvergne, Ricardo Jacobi, Fernando Moraes, Anne Marie Trullemans and Ramon Capillas.

8. References

[1] R.K.Brayton, G.D.Hachtel, A.L.Sangiovanni-Vincentelli, *Multilevel Logic Synthesis,* Proc. of the IEEE, vol. 78, No. 2, February 1990.

[2] R.Brayton, R.Rudell, A.Sangiovanni-Vincentelli, *MIS: A Multiple-Level Logic Optimization System*, IEEE trans. on CAD, vol. CAD-6, No. 6, November 1987.

[3] H.N.Guyen, L.Ducousso, D.Schnapp, *Design of a Standard Cell Library for Synthesis*, Proc. of the PATMOS, pp. 194-199, 1994.

[4] A.J.Velasco, X. Marín, J.Riera, R.Peset, J. Carrabina, *Performance Driven Layout Synthesis :Optimal Pairing & Chaining*, Proc. of the 5th ACM/SIGDA Physical Design Workshop, pp. 176-182, 1996.

[5] F. Moraes et al., *A Transparent Macrocell Layout Methodology*, Proc. of the PATMOS, pp. 42-54, 1993.

[6] H.R. Lin et al., *Cell Height Driven Transistor Sizing in a Cell Based Module design*, Proc. Of the EDAC, pp. 425-429, 1994.

[7] H.Y.Chen et al. *iCOACH : A Circuit Optimization Tool for CMOS High-Performance Circuits*, ICCAD 88, pp 372-375.

[8] C.Tsareff et al, *An Expert System Approach to Parametrized Module Synthesis*, IEEE Circ and Dev mag, Jan 1988, pp 28-35.

[9] J.Riera, *Mapatge tecnològic orientat a generadors de mòduls*, Ph.D. Thesis, Autonomous University of Barcelona, 1996.

[10] J.Riera, L.Ribas, A.J.Velasco, J.Carrabina, *From BDDs to CMOS Complex Gates*, IWLAS, Grenoble 1996.

[11] R.K.Brayton, *Factoring of Logic Functions*, IBM Journal of Research and Development, vol. 31, no. 2, March 1987.

[12] A.Reis, M.Robert, D.Auvergne, R.Reis, *From TBDD Functional Representation to CMOS Complex Gates,* X DCIS, pp. 159-164, 1995.

[13] M.Shoji, CMOS Digital Circuit Technology, Prentice-Hall Editors, 1988.

[14] F.N. Najm, *Transition Density: a New Measure of Activity in Digital Circuits*, IEEE Trans. On CAD of Integrated Circuits and Systems, Vol.12,No.2, Feb.93.

[15] V.Tiwari, P.Ashar, S.Malik, *Technology Mapping for Low Power*, Proc. of the 30th DAC, pp. 74-79, 1993.

[16] A.Shen, A.Gosh, S.Devadas, K.Keutzer, *On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks*, Proc. of the ICCAD, pp. 402-407, 1992.

[17] A.Gosh, S.Devadas, K.Keutzer, J.White, *Estimation of Average Switching Activity in Combinational and Sequencial Circuits*, Proc. of the 29th DAC, pp. 253-259, 1992.

[18] C.Metra, M.Favalli, B.Riccò, *Glicth Power Dissipation Model,* Proc. of the PATMOS'95 Conf., pp. 176-189.

[19] F.Najm, M.Zhang, *Extreme Delay Sensitivity and the Worst-Case Switching Activity in VLSI Circuits*, Proc. of the 32nd DAC, pp. 623-627, 1995.