Compact yet High-Performance (CyHP) Library for Short Time-to-Market with New Technologies

Nguyen Minh Duc and Takayasu Sakurai

Center for Collaborative Research and Institute of Industrial Science
University of Tokyo
7-22-1 Roppongi, Minato-ku, Tokyo, 106-8558, JAPAN
Tel: +81-3-3402-6226
Fax: +81-3-3402-6227
e-mail: duc@cc.iis.u-tokyo.ac.jp, tsakurai@iis.u-tokyo.ac.jp

Abstract Two compact yet high performance standard cell libraries (CyHP libraries), which contain only 11 and 20 cells respectively, are proposed. The first CyHP library leads to 5% increase in delay compared to a library containing about 400 cells. The second CyHP library suppresses delay increase up to 2%. The compact nature of these libraries not only reduces the cost and time for generation and maintenance substantially (thus enables new technologies to be adopted immediately), but also shortens synthesis time to about a half. Application results of these libraries to standard benchmarks and an industry design of about 80K gates are presented.

I. INTRODUCTION

Standard cell based design has become a mainstream design style for recent VLSI's. Standard cell libraries are getting bigger, containing more than 500 cells in expecting better performance of the resulting VLSI's. Generating, verifying and maintaining these big libraries, however, need lot of time and manpower and errors may be crept into the cell designs and/or cell characterization processes. Moreover, technologies are getting diverse and changing rapidly and a cell library must be generated from scratch more frequently. For example, low-power technologies such as MTCMOS, VTCMOS, DTMOS, and partially-depleted SOI [1] demand new layout and templates for the cells should be redesigned. To make the most of the more interconnection levels, self-aligned contacts, and other technology innovations, new cell design is required. These new cells cannot be generated by just changing the parameters of the parameterized cell templates. In this sense, the burden of the big cell libraries becomes an obstacle to realize short time-to-market of VLSI's.

Considering these issues, this paper proposes two compact yet high performance cell libraries (CyHP libraries) that contain fewer cells and is inherently efficient in generation, verification and maintenance. The drawback in adopting the small libraries may be area increase, delay increase, power increase and synthesis time increase.

Using multiple cell libraries, a widely used synthesis tool (Synopsys Design Compiler), a large set of benchmark circuits and a real industrial design, the paper shows the increase in these indices can be small and the synthesis time is even decreased to about a half. The area and power increase are not very important, because synthesized blocks by standard cells are only a part of the VLSI and memories, I/O's, clock systems and handcrafted datapaths occupy major portion of the area and power in most VLSI designs. The delay is important because the critical path in a synthesized block may determine the clock frequency of the total chip even though it is not major area-wise. The delay with the compact library, however, increases only about 2% even if the number of cells in a library is decreased to 20.

Since it is possible to make use of new technologies earlier with the compact libraries, it may achieve better overall performance at a fixed time point, although small increase in delay is observed if the same technology is used. The compact libraries will be beneficial to semiconductor suppliers as well as design houses that create their own set of standard cells from time to time to be independent from foundries.

In section II, an algorithm used to determine the CyHP libraries are explained. Section III shows the delay, area, power, and synthesis time results achieved with the CyHP libraries produced by this algorithm. An application of the CyHP libraries to an industry design is given in section IV.

II. ALGORITHM TO DETERMINE CELL SET

Generally there is no fixed rule to determine which cells a standard cell library should contain and the organization of a library is often determined in an empirical way. In this study, compact libraries are created by removing cells from existing libraries such that the increase of delay, area and power of the circuits generated automatically is kept as small as possible.

A standard cell library usually contains hundreds of cells. However, most of them do not influence much the performance (i.e. delay, area and power. Hereafter, *performance* will be used as a generic term to indicate these indices) of the circuit being synthesized. While primitive cells such as inverter, NAND, NOR, etc. are used very frequently, cells with relatively complicated functionality such as AND-OR-INV, majority circuit, and 8-input MUX are seldom used in random logic design. The circuit performance is almost unchanged even if these infrequently used cells are eliminated from the library. A more compact library can be obtained by removing these cells from an existing library. The problems is which cells to remove and what is the performance penalty occurs due to the absent of these cells.

When a cell is removed from the library, the target circuit is reconstructed by using other cells. The degree of circuit reconstruction depends on both functionality of the removed cell and its occurrence in the circuit. The more complicated the functionality of the removed cell is, the larger portion of the circuit needs to be reconstructed. Removing a cell that is used frequently will also lead to extensive reconstruction. Since complex cells always have large area, cell area times its occurrence (i.e. the area occupied by a cell in the entire circuit) can be used as a parameter to measure quantitatively its effect on the circuit structure. On the other hand, the circuit performance is almost unchanged if the change in the reconstruction is insignificant. Thus, the smaller the area occupied by a cell in a circuit, the less significant the effect of that cell to the circuit is. The cell with the smallest area is least important in terms of performance of the synthesized circuit.

Removing an appropriate number of least important cells from an existing library results in a new library, which has less cells than the original one but does not deteriorate much the performance of the circuit constructed by using the library. The above-mentioned procedure can be used as a heuristic rule to reduce cells of a library. The area occupied by a cell, however, depends on the nature of the circuit itself as well as the algorithm of the logic synthesis tool. Its average value over a large set of benchmark circuits should be used to guarantee the generality of the resulting library. In this study, the ISCAS89 benchmark set [2] is employed. This benchmark set consists of 31 sequential circuits, whose scales range from hundreds to around 20K gates. The average area portion occupied by a cell is calculated as following.

$$\bar{a} = \frac{1}{N_B} \sum_{i=1}^{N_B} \frac{f_i \times a}{A_i}, \tag{1}$$

where N_B (31 in this case) is the number of benchmark circuits, f_i is the occurrence of the cell in the i-th circuit, a is the cell area and A_i is the total area of the i-th circuit.

The flowchart of the library compacting algorithm is shown in Fig. 1. An existing standard cell library and a set of benchmark circuits are used as the input. The algorithm can be divided into the following steps.

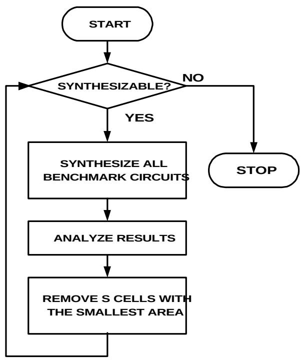
START
SYNTHESIZABLE?
NO
YES
SYNTHESIZE ALL BENCHMARK CIRCUITS
STOP
ANALYZE RESULTS
REMOVE S CELLS WITH THE SMALLEST AREA

Fig. 1 Flowchart of library compacting algorithm.

1. Synthesize (and optimize) all benchmark circuits using the original library *L*.
2. Analyze the circuits generated and use equation (1) to calculate the average area $\bar{a}$ occupied by each cell in *L*.
3. Remove *s* cells with the smallest $\bar{a}$ from *L* (*s* is an appropriate integer).
4. If *L* contains the minimum cell set required for synthesis, go to step 1. Otherwise stop.

The number of cells in *L* is reduced by *s* each time steps 1 through 3 are executed. A minimum cell set (normally an inverter, a 2-input NOR and a flip-flop are required) must exist in *L* to make the synthesis in step 1 possible. *L* becomes more and more compact as long as the execution proceeds. The algorithm will stop when one of these minimum cells is excluded from *L*. The logic synthesizer used in this study is a widely used commercial tool[3]. The area, delay and power used in step 2 are the values produced by the logic synthesizer. These values are used by designers to estimate performance of the synthesized circuits. The optimization in step 1 is done in a delay-oriented way, i.e. constraints are set such that the fastest circuits are generated (however, the result does not change if optimization is done in an area-oriented way, i.e. constraints are set such that the smallest circuits are produced).

With a given library containing a large number of cells, the above algorithm produces (in step 3) a series of libraries which have different number of cells. These libraries can be arranged in the order they are produced as L, L_1, L_2, …, L_m (*m* is the number of times that steps 1 through 3 are iterated). *L* is the original library and also the largest one in this series. L_m is the last and most compact library that is obtained just before the algorithm halts. The number of cells of L_{i+1} is less than that of L_i by *s* at least (since removing a cell from the library generally cause some other cells be unused). Let P_j and P_{ij} be the performance (which is one of delay, area and power as stated above) of the *j*-th benchmark circuit when using the original library *L* and the *i*-th library (i.e. L_i) in this series respectively. The relative performance of the *j*-th circuit achieved with L_i can be defined as,

$$p_{ij} = \frac{P_{ij}}{P_j} \tag{2}$$

The average of relative performance of all benchmark circuits then can be expressed as following.

$$\bar{p}_i = \frac{1}{N_B}\sum_{j=1}^{N_B} p_{ij} = \frac{1}{N_B}\sum_{j=1}^{N_B}\frac{P_{ij}}{P_j} \tag{3}$$

The relation between the number of cells and performance is obtained by calculating $\bar{p}_i$ for each library L_i. There exists a trade-off between $\bar{p}_i$ and the number of cells contained in L_i. Generally, decreasing the number of cells will increase $\bar{p}_i$ (i.e. deteriorates the circuit performance). A CyHP library with a small number of cells and an adequate performance can be achieved by examining carefully this trade-off.

III. Results

Based on the algorithm described in the previous section, an automated library generating system was implemented using Perl [4]. The number of cells removed each time, *s*, should be set to 1 in order to achieve a precise performance versus number of cells curve. However, as shown later, the performance stays almost constant when the number of cells is larger than 40. Thus, the calculation time can be reduced greatly by setting *s* to a big initial value (e.g. tens) then decreasing s gradually as the number of cells decreases.

Three libraries A, B and C are used as the starting library. They are all from different companies and have different design rules (1.5[μm], 0.6[μm] and 0.25[μm]). This means that they are very different in nature and the results gained from the experiments using these three libraries will be applying to any other libraries.

All these three libraries contain around 400 cells and are used actually in commercial and academic LSI designs. It takes more than a week to complete calculation for one library using a 400 MHz Sun Enterprise 450 workstation. The variation of delay, area and power versus the number of cells are shown in Figs. 2, 3 and 4 respectively. The horizontal axis signifies the number of cells. Delay, area and power are calculated using equation (3).

In addition to delay, area and power, the average time to synthesize each benchmark is also plotted in Fig. 5 (the synthesis time is calculated in a similar fashion using equation (3)).

Three libraries show the same tendency. In both three cases, the number of cells ranges between around 150 and 10. The maximum number of cells (about 150, which is the right most value of the curves in Figs. 2, 3, 4 and 5) is much less than the number of cells available in the original libraries. This means that the remaining 250 cells are left unused in this case.

It can be observed that the circuit performance is almost constant when the library contains more than 20 cells. Increase in delay is almost negligible, irrelevant to the number of cells. Area and power begin to increase considerably when the number of cells becomes less than 20. Even when the library has only 10 cells, delay, area and power increase no more than 5%, 30% and 40% respectively.

In contrast to performance, synthesis time varies irregularly. However, synthesis time tends to decrease when the number of cells is small.

The curves in Figs. 2, 3, and 4 show that the library with a number of cells between 10 and 20 is the appropriate CyHP

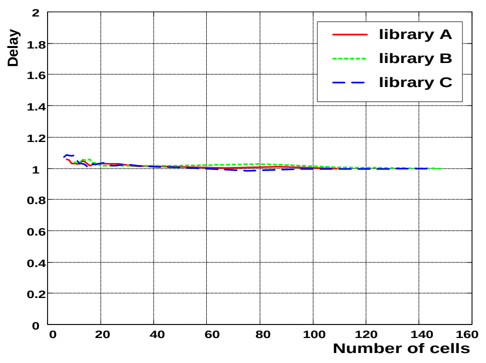

Fig. 2 Average of relative delay versus number of cells.

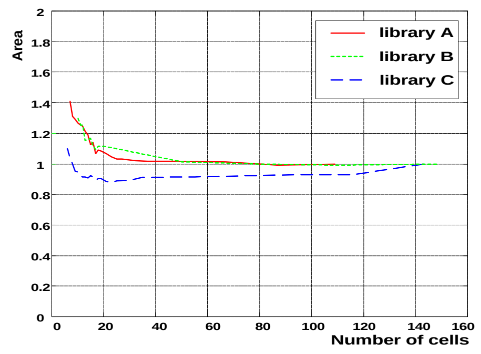

Fig. 3 Average of relative area versus number of cells.

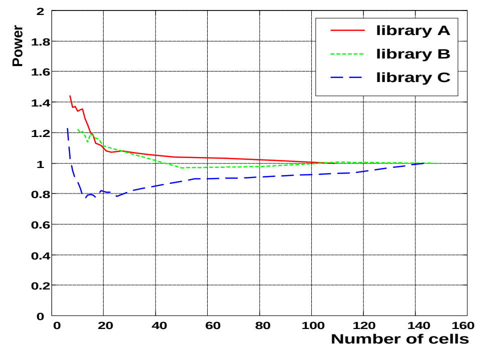

Fig. 4 Average of relative power versus number of cells.

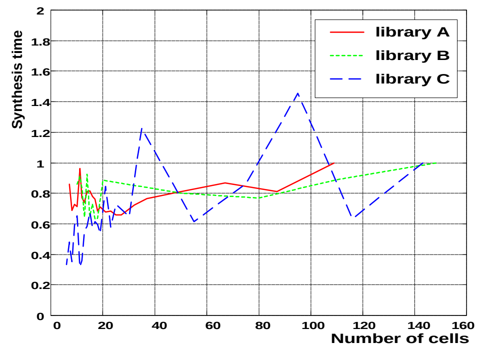

Fig. 5 Average of relative synthesis time versus number of cells.

library, which contains a small number of cells and gives a sufficient performance. Given the number of cells n, a CyHP library $L(n)$ and the performance $\bar{p}(n)$ achieved with it can be determined easily from these graphs. However, the compact libraries determined in this way are extracted from original libraries which are not totally identical, thus their cells are slightly different. Three different libraries used in this study result in three different CyHP libraries $L^{(A)}(n)$, $L^{(B)}(n)$, and $L^{(C)}(n)$ (the superscript denotes the original library). Most cells are common in both $L^{(A)}(n)$, $L^{(B)}(n)$, and $L^{(C)}(n)$ though a few cells exist in only one library. Nevertheless, the common part of these three libraries can be considered as the generic CyHP library.

As mentioned previously, there is a trade-off between the number of cells (which has direct impact on library generation and maintenance cost and time) and performance. Smaller number of cells can be achieved by sacrificing performance. In some cases, short time-to-market is much more crucial than performance, then a compact library is required. However, when time-to-market is not that critical, using a larger library will give better performance. Considering these issues, two CyHP libraries with different number of cells are proposed. The first one contains 11 cells, which are the common part of $L^{(A)}(15)$, $L^{(B)}(15)$, and $L^{(C)}(15)$. The second one contains 20 cells, which are the common part of $L^{(A)}(25)$, $L^{(B)}(25)$, and $L^{(C)}(25)$. The cells of these two compact libraries are shown in Tables I and II. Since the 20-cell library is a super set of the 11-cell library, only cells that do not exist in the 11-cell library are listed in Table II.

For the sake of reading, cells are classified in five categories: flip-flops, inverters, primitive gates, compound gates and multiplexers. Cell names are preceded by a number representing their fanin. The driving ability of each cell is expressed as x1, x2, etc. 2-InvNAND (2-InvNOR) is a 2-input NAND (NOR) with one inverted input. 2-MUXInv is a 2-input multiplexer with inverted output. D-FFN is a D flip-flop activated by clock's negative edge. 3-AND-NOR (3-OR-NAND) is a 3-input compound gate formed by cascading a 2-input AND (OR) into

TABLE I

CONTENTS OF THE 11-CELL CYHP LIBRARY

Flip-flops	D-FF x1, D-FF x2
Inverters	INV x1, INV x2, INV x4
Primitive gates	2-NAND x2 2-NOR x2 2-XNOR x1
Compound gates	2-InvNAND x2 2-InvNOR x2
Multiplexers	2-MUXInv x1

TABLE II

CONTENTS OF THE 20-CELL CYHP LIBRARY

Flip flops	D-FFN x1
Inverters	INV x8, INV x16
Primitive gates	2-NAND x1 2-NOR x1 3-NAND x1 3-NOR x1
Compound gates	3-AND-NOR x1 3-OR-NAND x1

(only cells that not in Table I are listed)

one of the two inputs of a 2-input NOR (NAND).

Since none of the benchmark circuits used in this study requires tri-state buffers and flip-flops with set/reset, these cells do not exist in the proposed CyHP libraries. However, they are often used in real design and should be contained in the CyHP library.

The average of relative delay, area, power and synthesis time (which are calculated using equation (3)) achieved with these two CyHP libraries are shown in Table III and IV. Adopting the 11-cell library increases delay by only 5% in average. This delay penalty can be reduced to just 2% if the 20-cell library is used. Average increase of area and power is 35% and 58% respectively, when using the 11-cell library. Since synthesized blocks are only a small part of the entire chip, this results in just a few percent of increase in total area and power. Furthermore, if the 20-cell library is in use, the increase of area and power is only 5% and 17% respectively.

Another important point should be noted is that using these CyHP libraries reduce synthesis time to a half. Large designs often take long time to synthesize, thus this can help to accelerate

TABLE III

AVERAGE DELAY, AREA, POWER AND SYNTHESIS TIME OF THE 11-CELL CyHP LIBRARY

Original library	Delay	Area	Power	Time
A	1.05	1.37	1.60	0.28
B	0.99	1.27	1.25	0.71
C	1.11	1.40	1.89	0.46
Average	1.05	1.35	1.58	0.48

(the numbers show the average of relative delay, area, power and synthesis time, which are calculated using equation (3))

TABLE IV

AVERAGE DELAY, AREA, POWER AND SYNTHESIS TIME OF THE 20-CELL CyHP LIBRARY

Original library	Delay	Area	Power	Time
A	1.00	1.08	1.19	0.31
B	0.98	1.04	1.07	0.43
C	1.07	1.03	1.26	0.48
Average	1.02	1.05	1.17	0.41

the design phase, since multiple runs of synthesis are customary in real designs.

IV. APPLICATION TO INDUSTRY EXAMPLE

This section demonstrates the effectiveness of the proposed CyHP libraries by applying them to an industry example. An 80K gate industry design that is written in RTL, was synthesized using both the original libraries (which have about 400 cells) and two proposed libraries. Performance obtained with the CyHP libraries is divided by that with the big libraries. The results are shown in Table V and VI. Adopting the 11-cell library does not increase but even decrease delay by 4% in average. Area and power increase by 54% and 58% respectively. The 20-cell library provides further improvement of delay (8%), while suppresses the increase of area and power at 20% and 38% respectively. Synthesis time is reduced to nearly a half in both cases. The 20-cell library takes longer synthesis time, but gives better performance.

TABLE V

DELAY, AREA, POWER AND SYNTHESIS TIME OF THE 11-CELL CyHP LIBRARY

Original library	Delay	Area	Power	Time
A	0.88	1.55	1.51	0.57
B	1.04	1.46	1.25	0.47
C	0.97	1.60	1.99	0.65
Average	0.96	1.54	1.58	0.56

TABLE VI

DELAY, AREA, POWER AND SYNTHESIS TIME OF THE 20-CELL CyHP LIBRARY

Original library	Delay	Area	Power	Time
A	0.86	1.28	1.29	0.59
B	1.01	1.19	1.21	0.51
C	0.89	1.14	1.64	0.71
Average	0.92	1.20	1.38	0.60

V. CONCLUSIONS

The paper shows that the number of cells in a library can be reduced greatly without deteriorating much performance of the circuits generated with it. An algorithm to determine cells to include in a library is presented. Two compact yet high performance (CyHP) libraries are proposed. Using the first one, which contains 11 cells, the delay increases by 5%, while with the second library which has 20 cells the increase of delay is only 2%. The compact nature of the proposed CyHP libraries not only reduces the cost and time required for generation and maintenance substantially, but also shortens synthesis time to a half, thus helps to accelerate the design phase.

The merit of the CyHP libraries is proved by the application to an industry design of about 80K gates.

ACKNOWLEDGEMENTS

This work was partly supported by a grant from Toshiba Corporation. Useful discussions with T.Kuroda, T.Takayanagi and M. Hamada are appreciated.

REFERENCES

[1] Takayasu Sakurai (Ed.), *Low Power High Speed LSI Circuits & Technology*, Realize Inc., 1998.

[2] Franc Brglez, David Bryan, Krzysztof Kozminski, “Combinational profiles of sequential benchmark circuits”, *Proc. of ISCAS*, pp. 1929-1934, 1989.

[3] *Design Compiler Reference Manual*, Synopsys, 1998.

[4] Larry Wall and Randal L. Schwartz, *Programming Perl*, Reading, O’Reilly & Associates Inc., 1991.