An ILP Formulation for Yield-driven Architectural Synthesis

Zhaojun Wo, Israel Koren and Maciej Ciesielski
Department of Electrical and Computer Engineering
University of Massachusetts,Amherst,MA 01003
E-mail: {zwo,koren,ciesiel}@ecs.umass.edu

Abstract

Data flow graph dominant designs, such as communication video and audio applications, are common in today's IC industry. In these designs, the datapath resources (e.g., adders, multipliers) count more than 90% in area. Different datapath resources have very different properties in terms of area, delay, power and yield. Considering yield during system level design can result in significant benefits. A Mixed Integer Linear Programming (MILP) formulation for yield-aware architectural synthesis is presented in this paper. The proposed approach attempts to maximize the yield of the design while satisfying other constraints like area and delay. Through experiments on several benchmarks, we show that incorporating the yield as an objective during architectural synthesis can significantly improve the yield compared to conventional methods. Transistor sizing at the circuit level can also be incorporated in our method to further improve the yield.

I. Introduction

A common goal in IC industry is the reduction of the manufacturing cost. Two factors affect the die cost: die area and die yield [1]. Considerable effort has been devoted to decreasing manufacturing cost by minimizing the die area. Reducing the die cost through yield maximization, however, is more complex since the relation between yield and die area is not monotonic. The sensitivity of a layout to defects is measured through its *critical area* [2] and by minimizing the critical area the yield can be maximized.

Because the critical area is directly related to the layout of the design, many schemes have been developed for reducing the critical area at the physical design level, such as during placement [3], routing [4], compaction [5], and cell library preparation [6]. Usually, the yield is dealt with as a secondary optimization goal. Current techniques can increase the yield by up to 10% [7].

In [8], the authors claim that somewhere below the $0.18\mu m$ technology node, the old rules-of-thumb cease to apply. The yield drops dramatically and is severely limited by design content. As a result, the yield should no longer be regarded as a secondary goal. To achieve additional improvements in yield it is necessary to modify the circuit topology itself. Synthesis-based yield optimization would allow the designers more flexibility than just layout-based changes. One of the design stages that can prove to be particularly beneficial is high-level synthesis, which is the process to determine the block level (functional units) structure of the circuit. Conventional architectural synthesis focuses mainly on area, delay, or power constraints, but not on yield.

Currently, data flow graph (DFG) dominant designs such as communication video and audio devices are very common. In these designs, the datapath resources (i.e., adders, multipliers, comparators), account for more than 90% of the die area while the controllers in these designs are relatively small and simple. To the best of our knowledge, very little research on improving yield at the architectural synthesis level has been published, since yield is traditionally considered as a secondary objective. In this paper, we show that considering yield during the system level design stage can result in significant benefits.

We present here an approach in which we represent the yield optimization problem as a mixed integer linear programming (MILP) problem. First, we use MILP to formulate the conventional scheduling and

binding during the architectural synthesis for the delay and area constraints. Then, we incorporate the yield objective using the Poisson yield model. Since the yield objective is non-linear, we adopt an approximate yield cost function which is linear. Finally, we utilize a commercial linear programming solver to find the global optimization solution for the synthesis process. Our major contribution in this paper is that the yield is regarded as an objective at the system level. As a result, the yield is maximized under the area, delay and power constraints. We also analyze the problem of minimizing the area with a yield constraint. Compared to the conventional solutions which do not take yield into account, our results show substantial yield improvements. We also show that the well-known method of transistor sizing can be incorporated into our approach to further improve the yield even without extra area penalty.

This paper is organized as follows. In Section II, a yield model at the system level is presented and a yield extraction tool for functional units is introduced. In Section III, the yield as a constraint is discussed and a mathematical formulation is proposed. Both area and delay optimization problems with a yield constraint are formulated. Our experimental results are shown in Section IV. Conclusions and future work are presented in Section V.

II. Manufacturing yield

A. Yield model of RTL-level

Various types of defects are introduced during the manufacturing process and may result in open or short circuits. However, not all the defects will necessarily cause a failure. It is common to quantify layout sensitivity to defects through the critical area [9]. The critical area, $A_i^c(x)$, is defined for a defect of type i and diameter x as the size of the area in which the center of the defect must fall in order to cause a circuit failure. A_i^c is defined as the critical area for a defect of type i averaged over all possible defect diameters.

$$A_i^c = \int A_i^c(x) f_d(x) dx \tag{1}$$

where $f_d(x)$ is the defect size probability density function.

Then, d_i is defined as the average number of defects of type i per unit area. Finally, the average number of *faults* in the circuit, denoted by λ, is:

$$\lambda = \sum_i A_i^c \cdot d_i \tag{2}$$

For a given circuit block b with an average number of faults λ_b, the yield is expressed using the Poisson model as [9]:

$$Y_b = \exp(-\lambda_b) \tag{3}$$

We assume that, at the architectural level, all the blocks are statistically independent, and the defects are uniformly distributed. Thus, for a system which has N functional blocks, the yield can be expressed as:

$$Y_{system} = \prod_{k=1}^{N} Y_k \tag{4}$$

where Y_k is the yield for block k.

¿From (3) and (4), we have:

$$Y_{system} = \exp(-\sum_{k=1}^{N} \lambda_k) \tag{5}$$

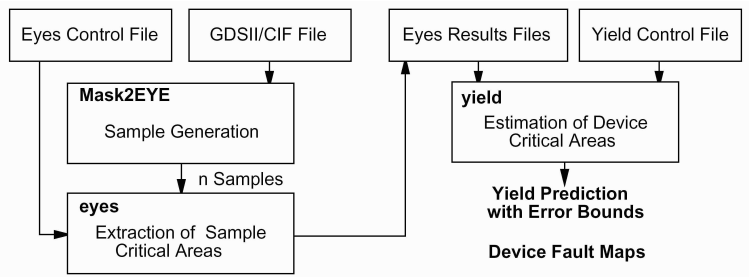

Figure 1. Flow of EYES prediction system

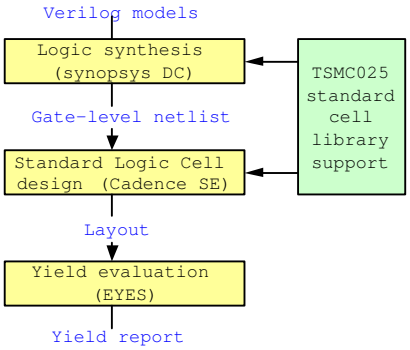

Figure 2. RTL components layout generate flow.

B. Yield simulation package

Our goal is to maximize the yield by selecting functional units from the library while satisfying given constraints. Conventional libraries characterize the area, delay and power properties of their cells/functional units. The yield value is not included and had to be calculated for our purposes. To this end, we use the EYES^{TM} (Edinburgh Yield Estimator Sampling) simulator [10] for calculating the yield of the various functional units. EYES implements a sampling based methodology for critical area estimation and then calculates the yield, based on the layout of the design. The procedure is shown in Figure 1.

C. Yield of library units

Adders and multipliers are the main units optimized in architectural synthesis of signal processing designs. There are various types of adders and multipliers, each with different area, delay and power characteristics. In order to compute the yield for functional units such as adders and multipliers, we generated the layout for each functional unit we needed. In our examples, we prepared a RTL-verilog file [19] for the different implementations of the functional units, synthesized to gate-level using Synopsys Design Compiler^{TM} and then used standard logic cell design methodology to generate the layout. The Cadence Silicon Ensemble^{TM} (Qplace+Wroute) has been adopted for this task. After the layout generation, EYES was used to estimate the yield as well as the area. The delay is estimated by the DC compiler in the gate level netlist. The detailed flow is shown in Figure 2. All the data are obtained from the TSMC 0.25 μm standard cell technology.

In our functional unit library, we implemented two different adders and two different multipliers. The two adders are the Ripple Carry Adder (RCA) and the Carry Look Ahead adder (CLA). The two multipliers are the Overturned-Stairs (OS) multiplier and the Balanced-Tree (BT) multiplier [18]. The measured results are shown in Table I. Column 2 shows the results of the Synopsys Design Compiler after logic synthesis to the gate level netlist. Column 3 and 4 show the results reported by EYES. Note that all inputs and outputs are 16 bits long.

It is not always the case that the smaller the area, the higher the yield. The yield value is highly related to the circuit structure as well as to the physical routing style. For example, the OS multiplier has a smaller delay (12.43ns) compared to the BT multiplier (15.94ns). The OS multiplier also has a smaller area (986 um^2 vs. 1025 um^2), however, it has an irregular structure and needs more routing tracks. Since the number of routing tracks between cell rows is limited, more tracks means that more routing layers are needed. As a result, for the OS multiplier, 3 layers are needed to complete the routing, while only 2 layers are needed for the BT multiplier. The yield value we obtained for the BT multiplier (0.966) is larger than that for the OS multiplier (0.946).

TABLE I
AREA, DELAY AND YIELD VALUE FOR ADDERS AND MULTIPLIERS IN OUR LIBRARY.

Resources	delay(ns)	area ($\times 10^{-6} cm^2$)	yield (%)
RCA adder	5.91	90	99.0
CLA adder	1.90	196	97.5
OS multiplier	12.43	986	94.6
BT multiplier	15.94	1025	96.6
register	0.46	55	99.4

III. MILP FOR YIELD CONSTRAINED ARCHITECTURAL SYNTHESIS

A. Problem definition

A data-flow graph (DFG) is a polar directed acyclic graph $G(V, E)$, where the set $V = \{v_i : i = 0, 1, ...n\}$ is the vertex set, and $E = \{(V_i, V_j) : i, j \in 0, 1, ..., n\}$ is the edge set. The vertex set represents the operators, and the edge set represents the data dependencies between operators. Given an unscheduled DFG, with available resources as shown in Table I, we have to find a yield-enhanced high-level synthesis such that the delay and area constraints are met and the overall system yield is maximized.

- **application:** DFG dominant designs (i.e., the datapath consumes most of the circuit area compared to the control logic), such as communication video and audio algorithms.
- **inputs:** Unscheduled DFG, with available resource library (functional units) to provide delay, area and yield properties.
- **outputs:** Scheduled DFG with binding information, such that the delay and area meet the constraints and the overall yield is maximized, or the delay and yield are constrained while the area is minimized.

B. MILP formulation for concurrent scheduling and binding

The parameters used in our MILP model are defined as follows. Let x_{il} denote a binary variable indicating that operator i is scheduled at time frame l, where $1 \leq i \leq n_{ops}$, and $1 \leq l \leq L$, where n_{ops} is the number of operators and L is the system latency. There are n_{res} resources (some may belong to the same functional unit) available. For each resource k, there are a_k implementations, each with delay D_k and area A_k. The binary variable b_{ir} indicates the binding of operator i to resource r, where $1 \leq i \leq n_{ops}$, and $1 \leq r \leq n_{res}$.

The mixed integer programming for concurrent scheduling and binding is shown in Figure 3 where A is the total area and d_i is the actual resource delay for implementation of operator i. Expression (6) is the unique slot scheduling constraint for each operator [11]. (7) is the unique resource binding constraint for each operator. (8) is the actual implemented delay for operator i. (9) is the data dependency relationship of the DFG [11]. (10) means that at most a_r copies of resource r are available. (11) is the total area constraint. (12) states that x_{il} and b_{ir} are binary variables. Finally, (13) indicates that the intermediate variable a_i should be an integer (N denotes the integer set) between 0 and n_{ops}.

Note that (10) is nonlinear. Since both variables x_{il} and b_{ir} are binary, the product of x_{il} and b_{ir} is also binary. There is a common technique [12] for linearizing the product by introducing another intermediate binary variable k_{ilr}, defined such that $k_{ilr} = 1$ if operator i is scheduled at time step l and is bound to the resource r, otherwise $k_{ilr} = 0$. Then, (10) can be rewritten as:

$$\begin{cases} \sum_{i=1}^{n_{ops}} \sum_{m=l-d_i+1}^{l} k_{imr} \leq a_r & \forall r \in [1, n_{res}], l \in [1, L] \\ x_{il} + b_{ir} - 1 \leq k_{ilr} & \forall i, l, r \end{cases} \tag{14}$$

$$\sum_{l=1}^{L} x_{il} = 1 \qquad \forall i \in [1, n_{ops}] \tag{6}$$

$$\sum_{r=1}^{n_{res}} b_{ir} = 1 \qquad \forall i \in [1, n_{ops}] \tag{7}$$

$$d_i = \sum_{r=1}^{n_{res}} D_r \cdot b_{ir} \qquad \forall i \in [1, n_{ops}] \tag{8}$$

$$\sum_{l=1}^{L} l \cdot (x_{jl} - x_{il}) - d_j \geq 0 \qquad \{i, j : (v_i, v_j) \in E\} \tag{9}$$

$$\sum_{i=1}^{n_{ops}} \sum_{k=l-d_i+1}^{l} x_{ik} \cdot b_{ir} \leq a_r \qquad \forall r \in [1, n_{res}], l \in [1, L] \tag{10}$$

$$\sum_{r=1}^{n_{res}} a_r A_r \leq A \tag{11}$$

$$x_{il}, b_{ir} \in \{0, 1\} \tag{12}$$

$$0 \leq a_i \leq n_{ops}, a_i \in \mathbf{N} \qquad \forall i \in [1, n_{ops}] \tag{13}$$

Figure 3. MILP Formulation for concurrent scheduling and binding.

B.1 Mixed Integer Linear Programming for Area constrained yield optimization

¿From (5), the yield maximization problem can be converted to the following:

$$minimize \qquad \sum_{r=1}^{n_{res}} a_r \lambda_r \tag{15}$$

The Mixed Integer Linear Programming model consists of (15), subject to (6)-(9), (11)-(14).

B.2 MILP for yield constrained area minimization

In this problem, the yield constraint is obtained from (5):

$$\sum_{r=1}^{n_{res}} a_r \lambda_r \leq -\log(Y_{bound}) \tag{16}$$

where $0 < Y_{bound} < 1$, is a parameter preset by the users. Again, the optimization objective is:

$$minimize \qquad \sum_{r=1}^{n_{res}} a_r A_r \tag{17}$$

subject to(6)-(9),(12)-(14) and (16).

IV. Experimental results

GAUT [17] was used to generate unscheduled DFG. We solved the MILP problems using a commercial MILP solver CPLEX [13]. The AMPL [13] language was used to model the formulations listed above.

A. Verification of area-constrained maximal yield synthesis

In this section, we illustrate the benefits of the proposed approach by comparing the results of the area-constrained maximal yield optimization with the conventional minimal area synthesis. Figure 4 shows the

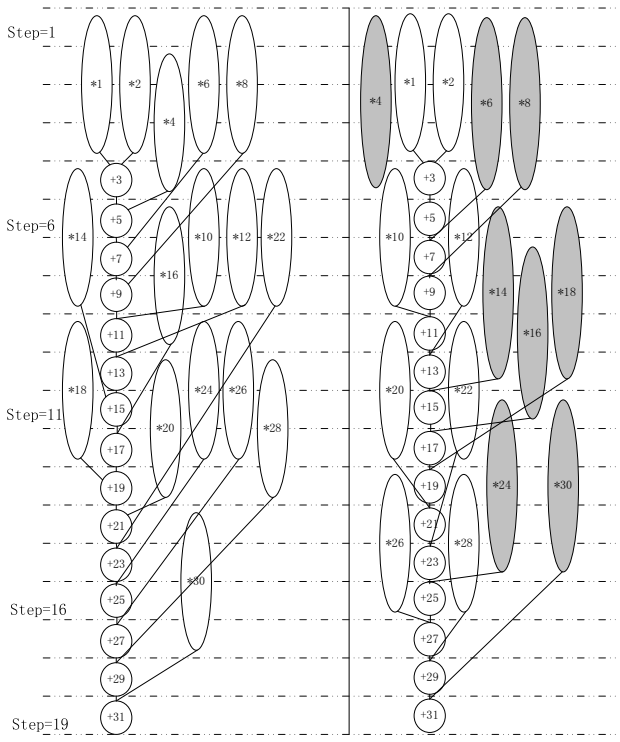

Figure 4. Two schedules for FIR16 benchmark with L=19. (a) minimal area solution: five fast OS multipliers are needed and the total yield is 0.716. (b) yield driven solution with a constrained area: the total yield is 0.760.

scheduled DFG for the FIR16 [17] benchmark, with the latency requirement of 19 cycles using the resource library specified in Table I. The left part is the minimal area solution which uses a fast adder (CLA) and five fast multipliers (OS). We assume that each functional unit (adder, multiplier) mapped needs a 16-bit register. The total area of this design is 6226 units and the total yield is 0.716. The right part of Figure 4 shows the result of the maximum yield synthesis with an area constraint of 120% of the minimal area solution. A fast adder (CLA), two fast multipliers (OS) and three slow multipliers (BT) are used. The total yield of this design is 0.760, improving the yield by 6.0% at the expense of a 4.5% extra area.

In order to verify our yield driven architectural synthesis, we generated the final layout for the above example and then used the EYES tool for final yield evaluation. The left part of Figure 5 is the layout we manually generated according to the scheduled DFG result in Figure 4. The right part is the critical area map reported by EYES. In this case, the reported yield value for Figure 4 is 0.717 for the minimal area solution and 0.773 for the yield driven solution. These are quite close to the predicted values, and indicate a possible increase of 7.8% in the yield. Note that the layout we generated ignores the controller.

Another example is shown in Figure 6. FFT is widely used in compression and decompression algorithms [17]. Our latency requirement is 13 clock cycles. The yield for the minimal area solution is 0.656, using two CLA adders and six OS multipliers. The yield for our yield-driven solution is 0.738, using two CLA adders and six BT multipliers. The relative yield improvement is 12.4% at an expense of 7.9% extra area.

B. Results for area-constrained maximal yield synthesis

We have selected several architectural synthesis examples [16] to evaluate our approach. We first performed a minimal area synthesis disregarding the yield (Algorithm B.2 without Equation (16)). Then, we generated an area constrained maximal yield synthesis (Algorithm B.1). For each example, 3 to 5 points were selected with different latency requirements. We set the area constraint to be 1.2 times the minimal area. Figures 7 and 8 show the detailed synthesis results with respect to the different latency requirements for the AR

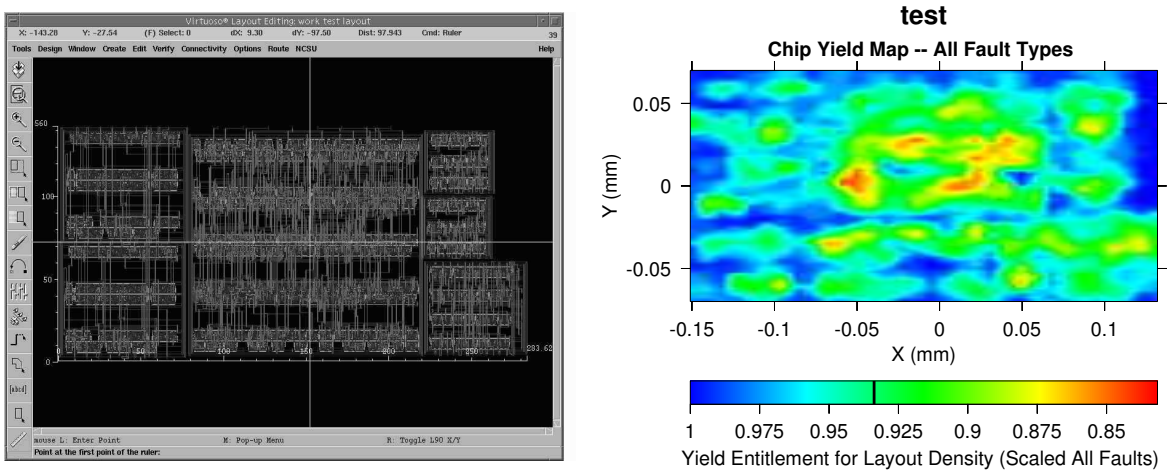

Figure 5. (a) Manually generated layout for Figure 4(b); (b) Yield analysis result using EYES

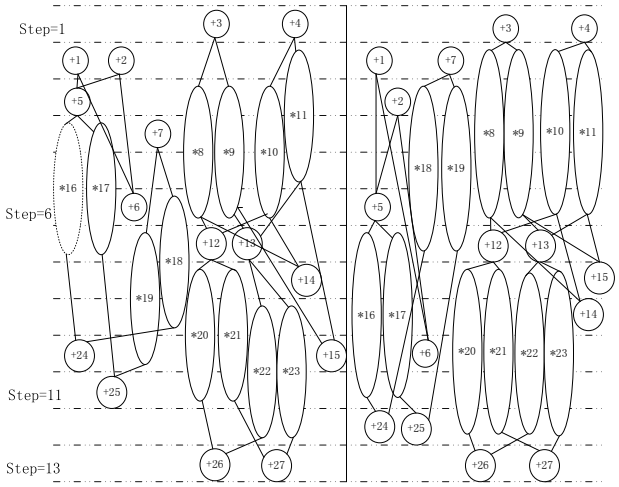

Figure 6. Two schedules for FFT benchmark with L=13. (a) minimal area solution: six fast OS multipliers are needed and the total yield is 0.656. (b) yield driven solution with constrained area: the total yield is 0.738.

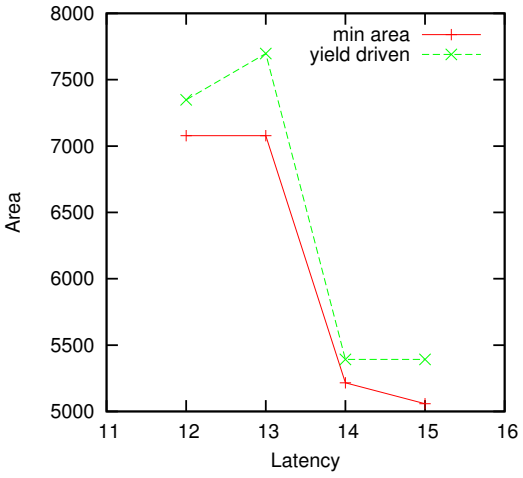

Figure 7. Area vs. latency for the FFT benchmark.

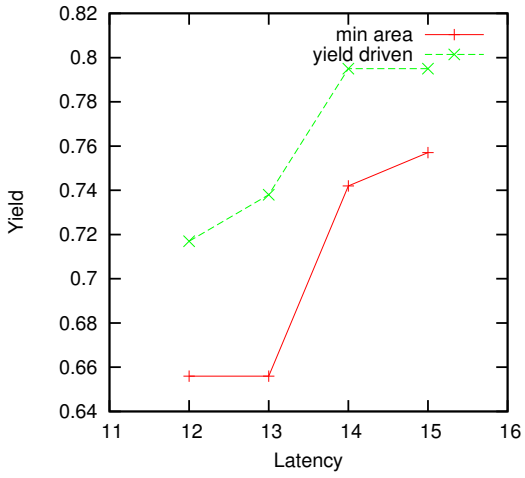

Figure 8. Yield vs. latency for the FFT benchmark.

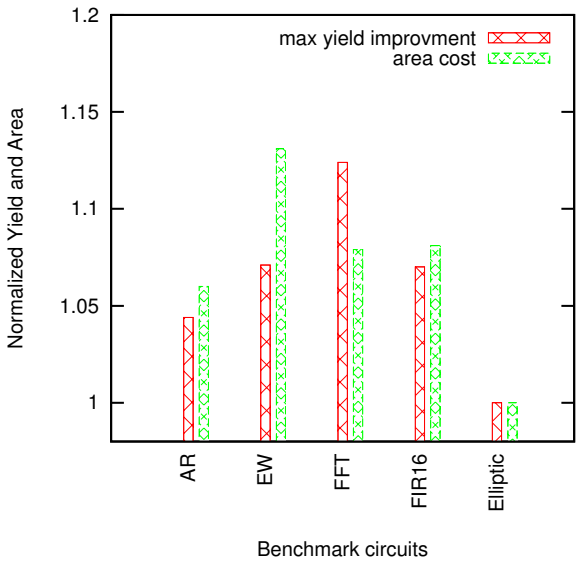

Figure 9. Benchmark results comparing area-constrained yield driven synthesis with minimal area synthesis (the results are normalized to the minimal area solution).

filter. Figure 7 shows the area for both the minimal area solution and the constrained area maximal yield solution. As expected, the required resource numbers decrease because of the increase in latency. Figure 8 shows a steady improvement of the yield.

Figure 9 shows the maximal yield improvement relative to the yield of the minimal area solution. The area is also normalized to its value for the minimal-area solution. From the figure, we see that the yield can be improved by as much as 12.4% with an area penalty of 7.9% (FFT). Note that for the Elliptic benchmark circuits, the minimal area solution and the area-constrained yield driven solution are the same, since only adders are used in this circuit. In our library (RCA,CLA), the RCA adder is better in terms of both area and yield. The solution which reduces the area will also improves the yield, and as a result, the synthesis solutions are the same.

C. Transistor sizing effect in library design

Transistor sizing can not only change the delay and area, but also affect the yield. For example, by carefully sizing the RCA adder we can pay little in terms of the area and yield and greatly improve the delay, as shown in Table II.

From the table above, the sized RCA_2 adder makes a trade-off between the original RCA_1 adder and CLA adder. By slightly sizing the transistors, the delay can be reduced by half with the yield only slightly affected. We incorporate the sized RCA adder into our resource library and re-synthesize the Elliptic circuit. Table III shows the results. In both latency cases, the yield is further improved (2.5% to 3.2%) because of the newly introduced resources while the area can be reduced by as much as 38.6%. We may conclude that by carefully sizing the resources, the yield can be further improved even without extra area penalty.

V. Conclusions and Future work

We have formulated in this paper the yield optimization problem as an MILP problem during the architectural synthesis procedure. In contrast to previous publications, the yield is optimized prior to the conventional circuit or layout design stages. From our experimental results, the yield can be improved by as much as 12.4% with an area penalty of 7.9%. The yield may be further improved by incorporating the conventional method of transistor sizing.

TABLE II
AREA, DELAY AND YIELD FOR TRANSISTOR SIZING OF THE RCA ADDER.

Resources	delay(ns)	area ($\times 10^{-6} cm^2$)	yield (%)
RCA_1	5.91	90	99.0
RCA_2	3.0	107	98.9
CLA adder	1.90	196	97.5

TABLE III
YIELD IMPROVEMENT BY TRANSISTOR SIZING FOR THE ELLIPTIC CIRCUIT WITH AREA CONSTRAINED MINIMAL YIELD SYNTHESIS.

Latency		no sizing	sizing	change
L=11	Area	898	631	-29.7%
	Yield	0.909	0.932	2.5%
L=12	Area	792	486	-38.6%
	Yield	0.918	0.947	3.2%

An important concern that may arise when using MILP is that the number of variables grows exponentially as the problem size increases. However, in some recent papers, relaxing the MILP problem to a linear programming (LP) problem is presented [14, 15] resulting in a polynomial rather than exponential growth.

VI. ACKNOWLEDGMENT

The authors wish to thank Dr. Gerard Allan from Predictions Software Ltd. for providing the EYES^{TM} software [10].

REFERENCES

[1] C. Ouyang and H. T. Heineken, "Maximizing Wafer Productivity Through Layout Optimizations," *13th Int'l Conf. VLSI Design*, pp. 192-197, 2000.

[2] I. Koren and Z. Koren, "Defect Tolerant VLSI Circuits: Techniques and Yield Analysis," *Proc. of IEEE*, Vol. 86, pp. 1817-1836, Sept 1998.

[3] R. K. Prasad and I. Koren, "The Effect of Placement on Yield for Standard Cell designs," *ISQED*, pp. 3-11, 2000.

[4] A. Venkataraman, H. Chen and I. Koren, "Yield Enhanced Routing for High-Performance VLSI designs," *Proc. of IEEE Intl. Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 97-105, Oct. 1994.

[5] V.K.R. Chiluvuri and I. Koren, "Layout Synthesis Techniques for Yield Enhancement," *IEEE Trans. on Semiconductor Manufacturing*, Vol. 8, pp. 178-187, May 1995.

[6] P. D. Dood, "Impact of DFM and RET on Standard-Cell Design Methodology," *Proc. Electronic Design Processes Workshop, IEEE CS Press*, pp. 62-69, 2003.

[7] S. A. Shaikh, J. Khare and H. T. Heineken, "Manufacturability and Testability Oriented Synthesis," *13th International Conference on VLSI Design*, pp. 185-191, 2000.

[8] R. Radojcic and M. Rencher, "Old Rules No Longer Apply," *EETimes*, April, 2003.

[9] A. Nardi and A. L. Vincentelli, "Synthesis for Manufacturability: a Sanity Check," *DATE*, pp. 192-199, 2004.

[10] http://www.icyield.com/eyes.html

[11] G. De Micheli, "Synthesis and Optimization of Digital Circuits," *McGraw-Hill Higher Education*, 1994.

[12] S. Tosun, *etl.*, "An ILP Formulation for Reliability-Oriented High Level Synthesis," *ISQED*, pp. 364-369, Mar. 2005.

[13] http://www.ilog.com/products/cplex

[14] J. Um, J. Kim and T. Kim, "Layout-Driven Resource Sharing in High-Level Synthesis," *ICCAD*, pp. 614-618, 2002.

[15] M. Ekpanyapong, etl. "Profile-Guided Microachitectural Floorplanning for Deep Submicron Processor Design," *DAC*, pp. 634-639, 2004.

[16] W. H. Wong and Rajiv Jain, "PARAS: System-Level Concurrent Partitioning and Scheduling," *ICCAD*, pp. 440-445, 1995.

[17] http://web.univ-ubs.fr/lester/www-gaut

[18] I. Koren, "Computer Arithmetic Algorithms," *A.K. Peters*, Second Edition, 2002.

[19] http://www.aoki.ecei.tohoku.ac.jp/arith/mg/index.html