LP based Cell Selection with Constraints of Timing, Area, and Power Consumption

Yutaka TAMIYA Yusuke MATSUNAGA
FUJITSU LABORATORIES LTD.
1015 Kami-kodanaka, Nakahara-ku, Kawasaki, JAPAN 211

Masahiro FUJITA
FUJITSU LABORATORIES OF AMERICA, INC.
77 Rio Robles, San Jose, CA 95134-1807

Abstract

This paper presents a new LP based optimal cell selection method. Optimal cell selection is useful tool for final tuning of LSI designs. It replaces drivabilities of cells, adjusting timing, area, and power constraints. Using the latest and earliest arrival times, it can handle both setup and hold time constraints. We also make an efficient initial basis, which speeds up a simplex LP solver by 5 times without any relaxations nor approximations. From experimental results, it reduces the clock cycle of a manual designed 13k-transistor chip by 17% without any increase of area.

1 Introduction

In recent LSI design, library cells with several drivabilities of the same logic are available. For example, Fujitsu 0.5 μm CMOS gate array library has four inverters, five for each type of nand gates, three for each type of flip-flops, etc. Those library cells have different properties of drivability, delay, area, and power consumption. Replacing cells of the same logic is useful for tuning of the circuit, which adjusts its timing constraints, total area, and total power consumption. However, optimal cell selection is an NP problem, Thus, it is difficult to tune a large circuit.

Almost all previous synthesis tools, such as DAGON [1] and MIS [2], can only handle circuits of small size at the same time. So they handle a large scale circuit by dividing it into several sub-circuits, and optimizing each of them. Critical paths in the original circuit may be split and scattered into those sub-circuits. Therefore some of sub-circuits may have the separated critical paths and some may not. It is no use to speed up sub-circuits, which have no critical paths. Also critical paths are determined by relative delays of sub-circuits. Thus, no one can know the final critical paths until all sub-circuits are processed. Therefore it is not preferable for timing optimization to divide a circuit into sub-circuits.

One technique, which overcomes this problem, is "gate sizing." Gate sizing does not change any network topologies but changes drivability of each gate. It raise drivabilities of gates on critical paths, and makes them faster. On the other hand, it lowers drivabilities of gates on non-critical paths, and saves area and power consumption. By balancing drivability of each cell, gate sizing tunes up the circuit with respect to delay, area, and power consumption.

Berkelaar [3] and Buurman [4] have proposed LP (Linear Programming) based gate sizing. It represents network delays of all paths in an LP problem. Therefore, it considers all path delays at the same time, and can optimize the circuit globally to meet design requirements, such as delay, area and power consumption. However, it does not consider any hold time constraints.

In this paper, we propose a new cell selection method, which works under both setup and hold time constraints. We introduce two kinds of arrival times [5] : the latest arrival time and the earliest arrival time. With these two arrival times, we can represent setup and hold time constraints. Our method is an extension of the LP based gate sizing [3]. Also we employ a technique to speed up a simplex LP solver. From a result of timing analysis, We make an efficient initial for a simplex LP solver. The initial basis does not change the solution, but speeds up the LP solver.

This paper is organized as follows. In Section 2 we explain our timing model. Linear approximation of a gate delay is illustrated in Section 3. In Section 4 we describe constraints about area and power consumption of a gate. Then we describe other constraints and the efficient initial basis for a simplex LP solver in Section 5. Section 6 gives experimental results. Finally, Section 7 is concluding remarks.

2 Timing Model

Our target circuit is sequential, combinational loop-free, and contains latches and/or flip-flops (we call both FF) synchronized with multi-phase clocks

There are two types of delay propagation: external and internal [5]. External propagation goes through a net and carries a wiring delay. Internal propagation is defined inside each gate, and carries a gate delay. We handles both rising and falling signal transitions. In order to simplify our discussion, we do not distinguish those signal transitions in the following discussion. It is easy to extend our discussion to distinguish them.

We define two arrival time variables at each pin: the latest arrival time and the earliest arrival time. We employs "static" delay calculation, i.e., we ignore logics of all gates, but consider all possible delay propagations. At each pin, we record arrival times of the latest delay propagation and the earliest delay propagation.

Let $T(i)$ and $t(i)$ be the latest and earliest arrival times of a signal i, respectively. External propagation (net delay from source i to sink j) is calculated as follows:

$$T(j) = T(i) + NetDelay_{ij} \qquad \forall j \in fanout(i) \quad (1)$$

©1994 ACM 0-89791-690-5/94/0011/0378 $3.50

$$t(j) = t(i) + NetDelay_{ij} \quad \forall j \in fanout(i) \quad (2)$$

Notice that $NetDelay_{ij}$ represents a constant of the wiring delay.

Internal propagation (gate delay from input i to output j) is calculated as follows:

$$T(j) = \max_{i \in fanin(j)} \{T(i) + MaxGateDelay_{ij}\} \quad (3)$$

$$t(j) = \min_{i \in fanin(j)} \{t(i) + MinGateDelay_{ij}\} \quad (4)$$

Variable $MaxGateDelay_{ij}$ and $MinGateDelay_{ij}$ represent maximum and minimum delays of a gate from input i to output j, respectively.

Eq. 3 and Eq. 4 have max or min operators. However, we can get linear inequalities without any relaxations as follows:

$$T(j) \geq T(i) + MaxGateDelay_{ij} \quad \forall i \in fanin(j) \quad (5)$$

$$t(j) \leq t(i) + MinGateDelay_{ij} \quad \forall i \in fanin(j) \quad (6)$$

In order to hold convexity with Eq. 5, $MaxGateDelay_{ij}$ must be convex upwards. Similarly, $MinGateDelay_{ij}$ in Eq. 6 must be convex downwards.

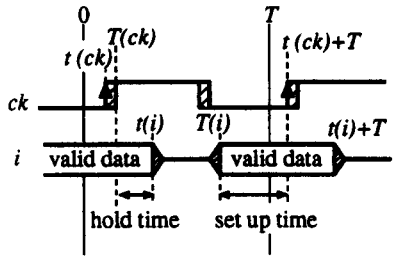

Figure 1: Setup and Hold Constraints of an FF

Our setup and hold time constraints of FFs are based on [6]. Let i in Fig. 1 be the input data signal of an FF, and ck be the control signal of the same FF. At the beginning of the clock period, data of i is valid. At $t(i)$ of the earliest arrival time, the data of i becomes invalid. Then, at $T(i)$ of the latest arrival time, the data of i becomes valid again. Thus, signal i holds valid data between $T(i)$ and $t(i) + T$ (we define T as the clock cycle period). Setup and hold time constraints are represented as follows:

$$t(i) - T(ck) \geq W_{hold}(i, ck) \quad (7)$$

$$T + t(ck) - T(i) \geq W_{setup}(i, ck) \quad (8)$$

In these equations, W_{hold}, W_{setup} represents minimum hold and setup constants, respectively. Eq. 8 has a term of T, because we should compensate one clock cycle for setup time evaluation.

It is trivial that Eq. 7 and Eq. 8 hold convexity with Eq. 5 and Eq. 6.

3 Gate Delay Model

A gate delay model heavily depends on a target technology. Thus we cannot discuss about a general gate delay approximation for LP formulation. We only can illustrate an example of linear gate delay approximation of our target technology.

Our target design is Fujitsu CMOS 0.5 μm gate array technology, and its gate delay calculation needs two factors: total driving capacitance of the output signal and "slew rate" of the input signal. Driving capacitance is a sum of wiring capacitance and loading capacitances of fanout gates. "Slew rate" represents dullness of a signal transition response of 'H → L' or 'L → H'. Let i and j be input and output signals of a gate, respectively. And let $CL(j)$ and $S(i)$ be the total driving capacitance of output j and the slew rate of input i, respectively. The gate delay from i to j is given as follows:

$$GateDelay_{ij} = t0 + \alpha \cdot S(i) + \{kcl + \beta \cdot S(i)\} CL(j) \quad (9)$$

where $t0$, kcl, α and β are characteristic constants of the gate internal path i to j.

"Slew rate" of output (j) is calculated at each gate output signal with the following equation:

$$S(j) = ts0 + kscl \cdot CL(j) \quad (10)$$

where $ts0$, and $kscl$ are characteristic constants of the gate output j.

It is obvious that Eq. 9 is non-linear because of multiplication of variables of $CL(j)$ and $S(i)$. Thus we introduce a first approximation: be $CL(j)$ a constant. In recent sub-micron technology, wiring capacitance has a majority in $CL(j)$. Gate sizing only swaps cells, but does not change layout. So we think wiring capacitance is a constant, and change of $CL(j)$ by swapping cells is negligible.

We approximate Eq. 9 and Eq. 10 with the least squares method. An approximated gate delay functions is a "plane", i.e., convex both upwards and downwards. So we can use this both as $MaxGateDelay$ and $MinGateDelay$.

$$MaxGateDelay_{ij} = MinGateDelay_{ij} = a1_{ij} \cdot Drive_g + a2_{ij} \cdot S(i) + a3_{ij} \quad (11)$$

$\forall i, j$: gate delay from i to j

$$S(j) = b1_j \cdot Drive_g + b2_j \quad (12)$$

$\forall j$: output of the gate

$$drive(min\_lib) \leq Drive_g \leq drive(max\_lib) \quad (13)$$

where $Drive_g$ is the drivability variable of gate g, $a1_{ij}$, $a2_{ij}$, $a3_{ij}$, $b1_j$ and $b2_j$ are constants, which are derived from linear approximation with the method of least squares. $drive(min\_lib)$ and $drive(max\_lib)$ specify the range of the drivability of the gate.

According to our experiments, the approximation error of our gate delay is less than 10%.

4 Gate Area and Power Consumption

Power consumption of a CMOS gate g is calculated by this equation:

$$GatePower_g = \frac{1}{2} \sum_{i \in fanin(g)} CL(i) \cdot f(i) \cdot V^2 \quad (14)$$

where $CL(i)$, $f(i)$ and V are loading capacitance of input i, transition frequency of input i, and the voltage, respectively.

V is a constant. Since we do not change connections of the network, $f(i)$ can be thought as a constant. $CL(i)$ depends on the assigned library cell. Thus, power consumption of a gate is a function of its drivability.

With a linear piece-wise technique, we approximate area and power consumption of a gate as functions of drivability:

$$GateArea_g \geq c1_k \cdot Drive_g + c2_k \quad k = 1 \ldots \mathrm{m} \quad (15)$$

$$GatePower_g \geq d1_k \cdot Drive_g + d2_k \quad k = 1 \ldots \mathrm{n} \quad (16)$$

Notice that $c1_k$, $c2_k$, $d1_k$, and $d2_k$ are constants, and m and n are number of linear piece-wise constraints of area and power consumption, respectively.

5 Optimal Cell Selection

5.1 LP formulation

In the previous sections, we have introduced basic relations among arrival times, slew rates, and drivabilities. In this section, we describe the remaining constraints in our LP formulation.

First, we add boundary conditions: arrival times and slew rates at the primary inputs of the circuit, and required times at primary outputs.

Second, we give limits of the cycle time, total area, and total power consumption.

$$T_{min} \leq T \leq T_{max} \quad (17)$$

$$A = \sum_{\forall g \in Circuit} GateArea_g \leq A_{max} \quad (18)$$

$$P = \sum_{\forall g \in Circuit} GatePower_g \leq P_{max} \quad (19)$$

where T_{max}, T_{min} specify allowable range of the cycle time T, and A_{max} and P_{max} specify maximum limits of the total area and power consumption, respectively.

Finally, we specify the objective function as follows:

$$Obj = a \cdot T + b \cdot A + c \cdot P \rightarrow \min \quad (20)$$

where a, b, and c are non-negative constants, which are given by the designer.

LP optimization answers the optimal drivability of each gate, where the gate is assumed to be assigned an arbitrary continuous drivability. After LP optimization, we assign each gate a library cell with a discrete drivability. First, every gate is assigned the cell of the nearest drivability. Then we iterate replacing cells in such greedy manner that reduces cost function of Eq. 20.

5.2 Initial Basis for an LP solver

A simplex algorithm is the method to find the basis of the optimal solution, by changing basic variables iteratively. If it starts from a basis near to the final one, it may get the optimal solution with small amount of run time. We found timing analysis is useful to generate such a initial basis.

We recall the latest arrival time constraint:

$$T(j) \geq T(i) + MaxGateDelay_{ij} \quad (21)$$

A simplex LP solver introduces a new slack variable $s_{ij} (\geq 0)$ for this inequality and change it an equation:

$$T(j) = T(i) + MaxGateDelay_{ij} + s_{ij} \quad (22)$$

An ordinary simplex algorithm starts with a trivial initial basis, that is, all slack variables are basic, and and non slack variables (such as $T(i), T(j), MaxGateDelay_{ij}$ in Eq. 22) are non-basic. With the following consideration, we can obtain much better initial basis.

In real design, $GateDelay_{ij}$ is larger than zero. According to Eq. 22, arrival time variable $T(j)$ is larger than zero. Since the value of a non-basic variable is not zero, $T(j)$ must be a basic variable ([7]).

If the path $i \rightarrow j$ is on the critical path, the slack variable s_{ij} equals to zero. Otherwise, s_{ij} is non-zero. Thus, a slack variable is a non-basic variable if its corresponding path is critical, and is a basic variable if its corresponding path is not critical.

Of course, we cannot know the actual critical paths, unless we get the optimal solution. But we can obtain hints of them from timing analysis of the initial circuit. Because replacing cells never causes drastic delay change, critical paths are assumed to not change drastically. Thus we consider the critical paths of the initial circuit is similar to ones of the optimal solution.

We make the initial basis by the procedure as follows:

1. Do timing analysis and mark slack variables on the critical paths.
2. Make slack variables marked above non-basic variables.
3. Make all arrival time variables basic variables.

We employ the Dijkstra method for our timing analysis. Its run time order is $O(E)$: where E is number of edges of the network.

6 Experiments

First, we evaluate the efficiency of speed-up described in Subsection 5.2. Table 1 shows the results. "No init basis" uses a default initial basis of LP solver, and "Init basis" uses our initial basis according to timing analysis of the initial circuit. The speed up ratio is over 4.8. Especially for L6 of a 21-bit multiplier, "No init basis" does not terminate in two weeks, but "Init basis" terminates in only 5 hours.

Second, we have done an experiments for some of benchmark circuits. The optimization condition is to minimize the cycle time with no increase of the total area, no hold time constraints at primary outputs, and no power consumption constraints. Initial circuits are mapped by SIS in delay mode (with option of "-n 1 -AFG"). Library cells used in technology mapping consists of four inverters, five 2-input NANDs, five 2-input NORs, and three D-type flip flops. Since the technology mapper of SIS does not consider slew rate, we have set α and β in Eq. 9 to 0, when we calculate delays.

The experimental results are illustrated in Table 2 both for combinational and sequential circuits. We ran these experiments with an LP solver of simplex method on a SUN SPARC 670. The maximum memory usage was 32MB. We also compare our method with simulated annealing (denoted as "SA" in the tables).

For all circuits, our method speeds up the initial circuits. The reason is that our method consider all path delays at the same time, while SIS does not.

Furthermore, we have done experiments of circuits of manual designs. The experimental circuits are taken from real designs. Especially, "L4" is a circuit of whole one chip. The optimization condition is the same at the a previous experiment. The target library contains 390 cells. Each cell has three or four cells with the same logic but with different drivabilities.

Table 3 illustrates the results. For all circuits, our method "LP" sped up the circuit more than 11% with almost no increase of area. As for "L4", we achieved 17% speed-up, by replacing 1,375 gates. Also results of our method are superior to ones of simulated annealing. However, our method need much more run time. Almost all run time of our method is spent for the LP solver.

Fig. 2 shows a time-power trade-off of a bus logic circuit. Transition frequency of each net is calculated by random pattern logic simulations. The graph presents an inverse proportional curve of cycle time and power consumption.

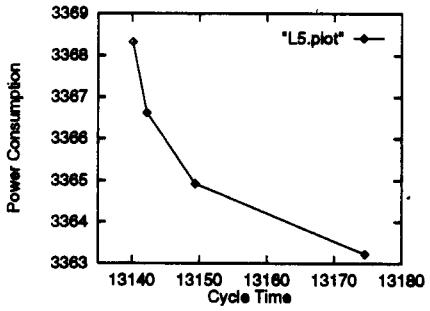

Figure 2: Time-power trade-off curve

7 Conclusion

We have proposed a new LP based optimal cell selection method, which considers all path delays at the same time, guarantees both setup and hold time constraints, and minimizes the cycle time, the total area, and the total power consumption. We conclude that our gate sizing method is very powerful for the use of tuning up a circuit of whole chip after physical layout design.

However, the problem of our method is large run time necessary to solve the LP problem. We may resolve this problem using a fast LP solver, such as an interior point method. Moreover, reduction of variables and constraints is effective for speed-up, since the number of variables and constraints in LP are linearly increasing with the number of gates. If we can know gates, which never become on critical paths, we can remove variables and constraints about arrival times of those gates.

References

[1] K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching," 24th DAC, pp. 341–347, 1987.

[2] R. K. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A Multiple-Level Logic Optimization," IEEE Trans. CAD, pp.1062-1081, Nov. 1987.

[3] M. R. C. M. Berkelaar, "Area-Power-Delay-Trade-off in Logic Synthesis," Ph.D. Thesis Eindhoven University of Technology, Eindhoven, The Netherlands, 1992.

[4] P. Buurman, M. Berkelaar, and J. Jess, "Computing the Entire Active Area versus Delay Trade-off Curve for Gate Sizing with a Piecewise Linear Simulator," Proc. of International Workshop on Logic Synthesis, 8c, 1993.

[5] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Minimum Padding to Satisfy Short Path Constraints," IWLS '93, 4a, 1993.

[6] K. A. Sakallah, T. N. Mudge and O. A. Olukotun, "$checkT_c$ and $minT_c$: Timing Verification and Optimal Clocking of Synchronous Digital Circuits," Proc. of ICCAD-93, pp.552–555, 1993.

[7] L. W. Wang, Y. T. Lai, B. D. Liu, and T. C. Chang, "A Graph-Based Simplex Algorithm for Minimizing the Layout Size and the Delay on Timing Critical Paths," Proc. of ICCAD-93, pp.703–708, 1993.

Table 1: Speed-up of an LP solver

Ckt	#gate	No init basis	Init basis (ratio)
C880	657	1h48m55s	22m47s (4.8)
L2	552	6h18m55s	52m57s (7.2)
L4	2088	108h5m	22h46m (4.8)
L6	5140	$\gg$ 2 weeks	4h55m ($\gg$ 50,000)

Table 2: Results for MCNC benchmarks

Ckt	Opt	Cycle[ps]	Area[Tr]	Run
C432	orig	9,917.8(1.00)	1,647.5	–
415 gates	SA	9,024.8(0.91)	1,646.7	8m46s
451 nets	LP	8,062.0(0.81)	1,647.4	17m25s
C880	orig	9,392.6(1.00)	2,568.4	–
657 gates	SA	8,459.8(0.90)	2,567.9	22m28s
717 nets	LP	6,839.8(0.73)	2,568.3	57m19s
S208	orig	5,793.7(1.00)	639.4	–
156 gates	SA	4,575.0(0.79)	638.8	1m14s
167 nets	LP	3,729.0(0.64)	639.2	2m40s
S400	orig	6,604.6(1.00)	1,340.5	–
326 gates	SA	6,104.5(0.92)	1,340.5	5m30s
330 nets	LP	4,589.5(0.69)	1,340.5	12m07s

Table 3: Results of Real Designs

Ckt	Opt	Cycle[ps]	Area[Tr]	Run
L1	orig	5,695.4(1.00)	2,166.9	–
303 gates	SA	5,305.0(0.93)	2,166.1	7m24s
722 nets	LP	5,084.9(0.89)	2,166.4	7m27s
L2	orig	17,559.0(1.00)	3,374.4	–
552 gates	SA	16,497.9(0.94)	3,373.6	25m31s
1732 nets	LP	14,522.9(0.83)	3,375.3	1h06m
L3	orig	8,212.1(1.00)	1,072.7	–
167 gates	SA	7,019.2(0.85)	1,072.9	2m13s
461 nets	LP	6,951.4(0.85)	1,070.6	2m49s
L4	orig	17,729.4(1.00)	13,404.0	–
2088 gates	SA	18,081.4(1.02)	13,415.8	3h19m
5308 nets	LP	14,653.6(0.83)	13,380.7	22h47m