Layout-based Logic Decomposition for Timing Optimization *

Yun-Yin Lian† Youn-Long Lin
Department of Computer Science, National Tsing Hua University
Hsin-Chu, Taiwan 300, R.O.C.
{sunny,ylin}@theda23.cs.nthu.edu.tw

Abstract

As feature sizes shrink to deep sub-micron, the performance of VLSI chips becomes dominated by the interconnect delay. In a traditional top-down design flow, logic synthesis algorithms optimize gate area or delay without accurate interconnect delay because of lack of physical design information. Thus, the effectiveness of the optimization techniques is limited.

We integrate logic synthesis and physical design into an iterative procedure for performance optimization. The logic synthesis process can optimize circuit delay based on accurate interconnect delay information extracted from the physical design. The physical design tools can refine the layout incrementally with the engineering change information and changed netlist passed from the logic synthesis process.

In this thesis, we integrate logic decomposition, gate sizing and buffer insertion to work together to improve the circuit speed. Experimental results on a set of benchmark circuits show that the techniques are indeed effective.

1 Introduction

In today's deep sub-micron design, chip delay is dominated by the interconnect delay. When designing high performance circuits, it is very important to make accurate interconnect delay information available in each step of the design flow.

We propose integrating logic synthesis and physical design into an iterative procedure in a flow. The layout-based logic synthesis algorithms can get more accurate interconnect delay information back-annotated from extracting the layout. The delay model we used is the distributed Elmore RC delay model [7]. The interconnect resistance and capacitance are calculated for getting realistic critical paths. The layout-based logic synthesis algorithms reduce the circuit speed by optimizing these realistic critical paths. The physical design tools refine the layout incrementally according to the changed netlist information passed from the logic synthesis process. After placement and routing, the resistance and capacitance of interconnects are extracted again for the next iteration.

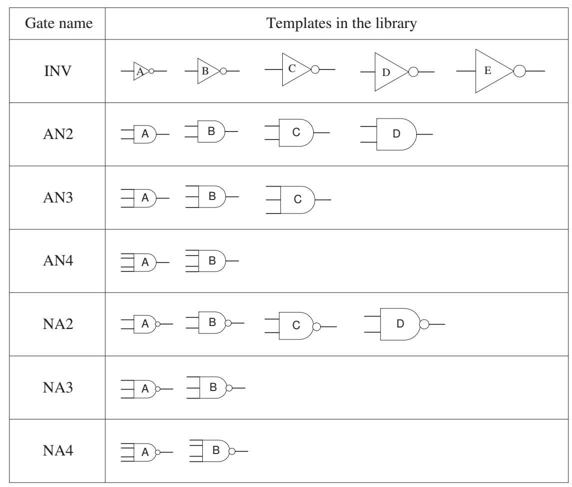

Figure 1: Library content.

There are many optimization techniques in layout-based logic synthesis. Some techniques, such as gate sizing, buffer insertion, and wire sizing, are performed without changing netlist structure. Their effectiveness is limited by the library and the persistent netlist structure. For example, we want to perform gate sizing operation on a circuit composed of only 4-input AND gates and INV gates, from the library shown in Figure 1. We can easily observe that if we only replace the 4-input AND gates with the same type but different template in the library, we have only two choices. If we can decompose the 4-input AND gate as shown in Figure 2(a), we will have more choices to get better result.

Another motivation of logic decomposition is to decompose a cell whose inputs can be divided into two parts: critical inputs and non-critical inputs. For example, as shown in Figure 2(b), the delay of the critical path is reduced because the slow cell has been replaced by a faster one.

In this thesis, we integrate logic decomposition, gate sizing and buffer insertion to simultaneously work together to improve the circuit speed. The rest of this thesis is organized as follow: In section 2, our system flow is introduced. Section 3 describes our proposed methodology. Some ex-

*This work was supported in part by the National Science Council, R.O.C., under a contract no. NSC87-2215-E-007-017.

†Yun-Yin Lian is now with Global UniChip Corp.,Taiwan

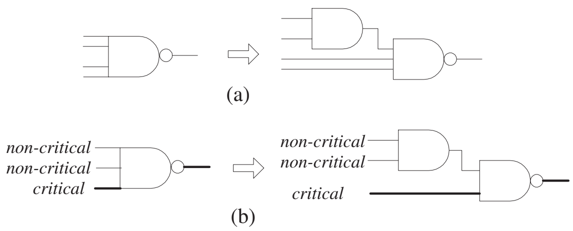

Figure 2: Logic decomposition.

perimental results are reported and analyzed in Section 4. Section 5 gives conclusions.

2 Our System Flow

The flow of our system is shown in Figure 3. Inputs to our system are a mapped netlist, a cell library and several control parameters.

After performing the Initial Placement and Routing process, we can get the realistic resistance and capacitance of interconnects. The delay calculator is performed for calculating the realistic critical paths. If the timing constraint is not meet, the system will apply the layout-based optimization procedure on the cells along critical paths. This procedure integrates three techniques: gate sizing, buffer insertion and logic decomposition working together to reduce the delay of critical paths.

The first step of the iterative procedure is to score cells along K-most critical paths by three cost functions which are identified by sizing, buffering and logic decomposition algorithms, respectively. Then, we select N highest buffering score cells, M highest sizing score cells and positive logic decomposition score cells. (M and N are input parameters) Finally, we change the selected cells by gate sizing, buffer insertion and logic decomposition. The engineering change information consists of the name of changed cells and the location of inserted cells.

The incremental placement and routing process are performed in the cells which are changed during the previous process. The layout of the circuit is incrementally refined by the engineering change information. The delay calculator is performed again by using newly extracted resistance and capacitance of interconnects to get new critical paths for next iteration.

The processes of the system are iteratively performed until the delay of the circuit meets the timing constraint or fails to improve further. Finally, the circuit area is reduced by the buffer removal and gate downsizing process if the timing constraint is meet.

2.1 Layout-based Timing Optimization Techniques

There are three layout-based timing optimization techniques we used in this system. (1) Gate sizing: This approach tries to size up or size down a cell with its functionality-equivalent gate from the library. We size up a cell in order to speed up

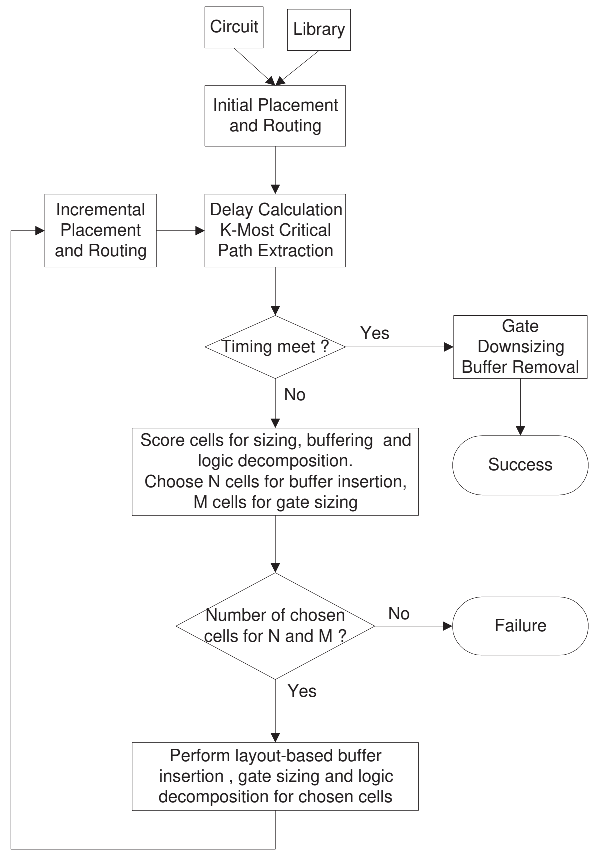

Figure 3: System overview.

the critical paths and size down a cell for reducing the area of the circuit. (2) Buffer insertion: This technique is especially useful for nets with high number of fanouts. We insert buffers into critical nets to decouple fanout loading and remove buffers for area reduction. (3) Logic decomposition: We perform logic decomposition operation on two cases: first, when inputs of a cell contain critical part and non-critical part, we decompose this cell into two smaller cells for reducing the timing of critical paths. Second, we decompose a gate that can not be sized up any more into the other gates which will have chance to size up for getting better results.

2.2 Incremental Placement and Routing

We record the engineering change information in the layout-based timing optimization process for incremental placement and routing. The information written to the ECO database consists two parts: (1) Changed cells: The changed cells can be the sized cells or the decomposed cells. We use a pair ($gate_i$, $template_j$) to represent a cell $gate_i$ that will be replaced by another gate $template_j$. $Template_j$ substitutes for $gate_i$ at the location of $gate_i$ by the placement tool. Cells whose locations behind $gate_i$ in the same row will be shifted as the cell expands or shrinks. (2) Inserted cells: The inserted cells can be the inserted buffers or the decomposed cells. The location of newly inserted cell gatei will be specified in a form of ($gate_i$, row_i , $column_i$). The placement tool will try to place $gate_i$ as close to the specified location as possible.

The placement tool incrementally refines the layout information with the ECO database. Then the routing tool is performed to optimize the critical path delay by the list of the critical paths. Finally, the layout-based logic synthesis extracts the new RC information which is refined by the incremental placement and routing process for the next iteration.

3 The Proposed Method

We propose a method, logic decomposition, which contains two parts: one is to decompose a larger cell which contains critical inputs and non-critical inputs into two smaller cells for reducing the crtical path delay. The other is to decompose a gate that can not be sized up any more into the other gates that will have chance to size up for getting better results.

Many optimization techniques in layout-based logic synthesis have been proposed. Some techniques, such as gate sizing, buffer insertion and wire sizing are performed without changing the netlist structure. Their effectiveness is limited by the library and the persistent netlist structure.

We can easily obtain an observation when we investigate the library.

Observation 1: The number of the template of every gate in the library is different.

For example, the number of the template of NA2 in Figure 1 is 4 (from NA2A to NA2D) and NA4 is 2(from NA4A to NA4B). When we perform gate sizing process of a circuit, we can only size a NA4 gate up to NA4B if we does not change the structure of the circuit. If we can restructure the NA4 gate to a AN2 gate and a NA3 gate cause these two gates have more templates in the library, the gate sizing space will be increased by logic decomposition process.

When we observe the cells in the critical paths of a circuit, we can get another observation.

Observation 2: Inputs of a cell in critical path can be separated into two parts, critical inputs and non-critical inputs.

The primary goal of timing optimization procedure is to speed up the critical paths of a circuit. If we can replace a big gate by a smaller one whose delay is smaller than the original one. We can speed up this path by logic decomposition process.

As the example shown in Figure 4, we have four two-level candidates. The number in parentheses represents the criticality of the input which is decided by the slack value of fanin cells of the NA4 gates.

The logic decomposition function includes four steps:

- **Cell selection:** First we choose cells which satisfy the following conditions: (a) the cell is in critical paths (b) this cell must satisfy one

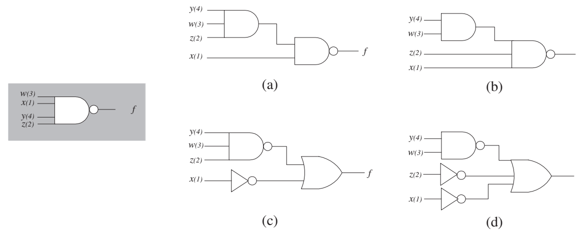

Figure 4: Logic decomposition of an NA4 gate.

  of the following conditions: i.the cell have negative GS_score in this iteration or ii.the cell had been sized up to the largest template before this iteration or iii.the cell's inputs can be divided into critical part and non-critical part

- **Cell substitutes generation:** For each cell chosen by step 1, we use table lookup method to generate some two-level substitutes for replacing the cell. The substitutes are equivalent in functionality with the original cell.

- **Electee Decision:** We estimate the longest path delay passing through original cell and each substitute's paths delay. For cells satisfy condition i and ii, we calculate the substitute's paths delay assuming each cell uses the largest template in the library. Thus, we can see which substitute can increase the gate sizing space most. For cells satisfy condition iii, we assign each cell of the substitute to the smallest template. We select the substitute such that (a) Its longest path delay is less than the longest path delay of the original cell. (b) Its longest path delay is the smallest one among all substitutes'. If such substitute exists, we mark the LD_score for the next step.

- **Cell substitute:** We replace cell by the substitute which is specified by above operation. This step contains four parts: (a) Sort the inputs of original cell by criticality: for example in Figure 5, cell B is decomposed into cell B1 and cell B2. The number in parentheses represents the criticality of the path (or input). We assign the most critical inputs of B to cell B2 and the others to cell B1. (b) Get the proper size of each cell of the chosen substitute: we first assign the smallest template to each cell of the substitute and calculate its paths delay. If some paths delay are greater than the longest path delay of original cell, we size up one of the cells which are passed by these paths and recalculate again. If there are some paths delay still greater than the longest path delay of original cell, we will size up the other cell of the substitute. This operation is repeatedly performed until all paths delay of the substitute are less than the longest path delay of original cell. (c) Replace the cell with the

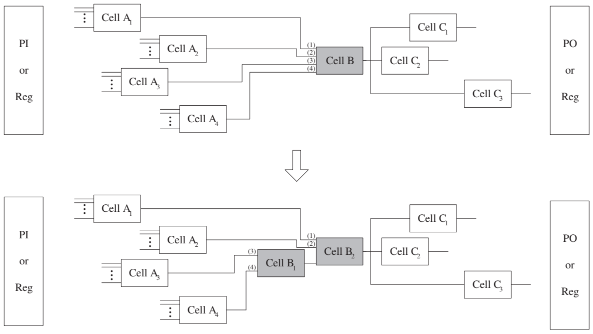

Figure 5: Decompose operation.

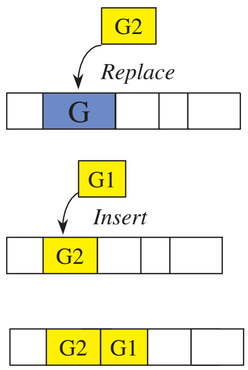

Figure 6: Placement and routing operation.

substitute: for example in Figure 5, cell B2 replaces cell B in the same location and the location of cell B1 is near cell B2. (d) Write engineering change information into the E-CO file for incremental placement and routing process.

Figure 6 shows the replaced operations in incremental placement and routing process of logic decomposition. Cells G is the selected cell of logic decomposition process. G2 and G1 are the proper sized gates of the substitute for replacing cell G.

4 Experimental Results

We conduct experiments using a set of benchmark circuits from MCNC and ISCAS which are mapped by SIS's [8] technology mapper. The SIS's logic gate sizing technique is performed simultaneously with the mapper. The technology we used is the TSMC 0.6 μm CMOS cell library [9]. The initial placement is generated by TimberWolfSC v6.0 [10].

The notations of G, D and B in Figure 7 represent gate sizing, logic decomposition and buffer insertion, respectively. The average delay reduction of G is 16.39%, GB is 19.96%, GD is 24.24% and GBD is 25.43%. The number of inserted buffer we limited is two in each iteration.

5 Conclusions

We have proposed a method, logic decomposition for improving the performance of circuit in this paper. There are two goals of logic decomposition: one uses logic decomposition to break the limitation of gate sizing. Another divides the critical inputs and non-critical inputs of a cell for speeding up critical paths. Logic decomposition can fully utilize the library we used. It also can improve the performance of circuits due to the constraints we specified is very strictly.

Circuit	SISGS	G		GD		GB		GBD	
	delay	delay	improve	delay	Improve	delay	Improve	delay	Improve
C1355	11.94	11.11	6.95%	9.92	16.92%	10.34	13.40%	9.5	20.44%
C5315	19.13	15.24	20.33%	14.73	23.00%	15.19	20.60%	14.48	24.31%
s298	5.56	4.56	17.99%	4.48	19.42%	4.56	17.99%	4.5	19.06%
s1423	14.23	13.29	6.61%	12.12	14.83%	13	8.64%	12.1	14.97%
s5378	13.27	10.11	23.81%	9.75	26.53%	9.75	26.53%	9.57	27.88%
s9234	10.93	8.84	19.12%	8.25	24.52%	8.77	19.76%	8.1	25.89%
s13207	14.65	13.83	5.60%	12	18.09%	13.7	6.48%	11.84	19.18%
s15850	21.4	16.93	20.89%	15.51	27.52%	16.08	24.86%	15.3	28.50%
s35932	108.18	97.29	10.07%	88.62	18.08%	91.38	15.53%	83.22	23.07%
s38417	43.52	29.34	32.58%	20.24	53.49%	23.57	45.84%	21.35	50.94%

Figure 7: Experimental results.

References

[1] K-J. Hwang and Y-L. Lin, *"Post-Layout Performance Optimization Using Gate Sizing and Buffer Insertion"*, Department of Computer Science NTHU,May 1997.

[2] M. Pedram and N. Bhat, *"Layout Driven Logic Restructuring/Decomposition"*, Proceedings of the 30th Design Automation Conference, pp. 230-235, 1993.

[3] K. Singh, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli, *"Timing Optimization of Combinational Logic"*, ICCAD, pp. 282-285, November 1988.

[4] S. Malik, E. M. Sentovich, R. K. Brayton, and A. Sangiovanni-Vincentelli, *"Retiming and resynthesis: optimizing sequential networks with combinational techniques"*, Proceedings of the Twenty-Third Annual Hawaii International Conference pp. 397-409, 1990

[5] Y. M. Jiang, A. Kristic, K.T. Cheng, and M. Marek-Sadowska, *"Post-Layout Logic Restructuring for Performance Optimization"*, Proceedings of the 34th Design Automation Conference, pp. 662-665, June 1997.

[6] K. Sato, M. Kawarabayashi, H. Emura and N. Maeda, *"Post-Layout Optimization for Deep Submicron Design"*, Proceedings of the 33th Design Automation Conference, 1996.

[7] W. C. Elmore, *"The transient response of damped linear networks with particular regard to wide-band amplifiers"*, Journal of Applied Physicals, pp. 55-63, 1948.

[8] E. M. Sentovich, et al., *"SIS: A System for Sequential Circuit Synthesis"*, Department of Electrical Engineering and Computer Science, UC Berkeley, May 1992.

[9] *"COMPASS 0.06u, 5-Volt, High-Performance, Standard Cell Library"*, COMPASS, Corp., June 1994.

[10] C. Sechen, *"TimberWolf6.0: Mixed macro standard cell floor planning, placement and routing package"*, User's manual, Yale University, Sep., 1991.