Delay Estimation for Technology Independent Synthesis

Yutaka TAMIYA

FUJITSU LABORATORIES LTD.
4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, JAPAN, 211-88
Tel: +81-44-754-2663
Fax: +81-44-754-2664
E-mail: tamy@flab.fujitsu.co.jp

Abstract

This paper proposes “path mapping”, a method of delay estimation for technology independent combinational circuits. Path mapping provides fast and accurate delay estimation using the common ideas with the tree covering based technology mapping. First, path mapping does technology mapping for all paths in the circuit with minimum delay. Then, it finds the most critical path among all the paths in the circuit. Finally, it answers its path delay as the circuit delay.

Experimental results show path mapping estimates more accurate circuit delay than unit delay, and runs much faster than the technology mapper.

1 Introduction

In a general LSI design flow, a circuit is first optimized at technology independent level, then, is mapped into the technology library cells, and finally optimized at technology dependent level. At all design phases, “delay” plays an important role in measuring goodness of the circuit.

Especially, technology independent synthesis needs both fast and accurate delay estimation. Technology independent synthesis iterates hundreds times of trials that modifies the circuit and estimates its delay. Thus delay calculation time should be small to reduce synthesis time. In addition, the actual circuit delay cannot be calculated unless all nodes in the circuit is made from technology library cells. There are two previously proposed delay estimations for a technology independent circuit: unit delay and technology mapping.

Almost all synthesis tools use unit delay [1] to estimate delay of a technology independent circuit. It first decomposes the circuit into inverter (INV) and 2-input nand (NAND2) nodes. Then it finds the maximum number of NAND2's on a single path from a primary input to a primary output, and recognizes that number as the circuit delay.

Extension of this unit delay for complex logic function nodes has been proposed by [2]. It does not actually decompose the circuit into INV and NAND2 nodes, but it calculates delay of a complex node with a similar way of unit delay. Thus, this method has no essential differences from unit delay.

Run time of unit delay is very small. Accuracy of unit delay, however, is low because unit delay does not consider technology mapping. For example in Fig. 1, nodes on path $A \to X$ can be mapped to a NAND4 cell. In this case both A and C will be an input of a NAND4 cell, and path delays of $A \to X$ and $C \to X$ are almost the same. However, unit delay says path $A \to X$ is 1.5 times larger than path $C \to X$.

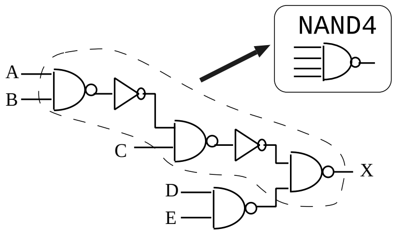


Figure 1: Difference of Unit Delay and Technology Mapping

Another delay estimation of technology mapping answers the exact circuit delay with the actual delay model of the technology library. The most efficient technology mapper is based on tree covering algorithm [3], which runs in linear time to the number of nodes in the circuit. However it is very slow for technology independent synthesis to use it iteratively.

We propose a new delay estimation method of “path mapping”. It provides accurate delay estimation using the common ideas with the tree covering based technology mapping. Also it runs fast, so that it is suitable for technology independent synthesis.

This paper is organized as follows: definitions of path mapping and delay calculation are presented in Section 2. Section 3 presents an efficient algorithm for delay calculation, and is followed by some heuristics for more accurate delay estimation in Section 4. Then we discuss experimental results in Section 5 and conclusion in Section 6.

ASP-DAC ’97
0-89791-851-7/$5.00 ©1997 IEEE

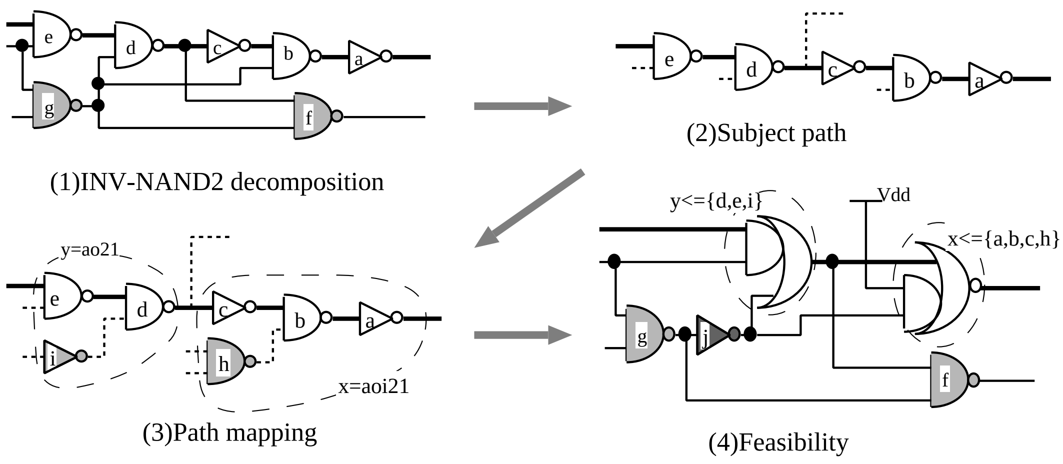


Figure 2: Example of path mapping

2 Path Mapping

Path mapping estimates the circuit delay by the following equation:

$$\max_{\forall p \in Paths} \min_{\forall m \in Mappings} \{\text{Delay of } p \text{ mapped by } m\} \quad (1)$$

The "min" operator of this equation means that path mapping finds the fastest technology mapping for each path in the circuit. And the "max" operator means that path mapping finds the most critical path among all paths in the circuit. Details of Eq. 1 are represented in the following subsections.

2.1 Mapping on a Subject Path

Technology mapping within the "min" operator of Eq. 1 proceeds as follows:

Step 1 First, extract one path from a primary input to a primary output (Fig. 2(1), (2)). We call this extracted path a "subject path". Remove all nodes and nets not on the subject path. Record the number of fanouts and number of inputs at each node on the subject path (They are shown with dotted lines in Fig. 2(2)).

Step 2 Map nodes on the subject path with library cells. In this mapping some extra INV and/or NAND2 nodes can be added to the side inputs of the path, so that original nodes on the path and added nodes will match with some library cell. In Fig. 2(3) an AOI21 (AND-OR-INV21) is mapped to nodes a, b, c, and an extra NAND2 node h. Also AO21 (AND-OR21) is mapped to nodes d, e, and an extra INV node i.

This mapping prohibits to map a single library cell across a fanout point: i.e., two nodes just before and after a fanout point cannot be mapped to the same one cell. That's because tree covering based technology mapping never maps a single library cell across a fanout point.

Note that resultant mapping of Step 2 ignores two kinds of nodes:

- Unused nodes within mapped library cells, which does not map to the subject path.
- Nodes not on the subject path in the circuit.

Even though ignoring such those nodes, the resultant mapping by Step 2 is feasible, if the technology library satisfies certain requirements. The requirements are as shown here.

Definition 1 *"Technology Library Requirements for Feasible Path Mapping"*
The technology library satisfies the following all conditions:

1. *It contains cells of constant 0, constant 1, INV, and NAND2.*
2. *The technology mapping uses only combinational and single-output library cells.*
3. *The technology mapping only uses INV-NAND2 tree decompositions of library cells.*

□

Requirement 1 and 2 are already satisfied by almost all CMOS libraries and tree covering based technology mappers. Requirement 3 means path mapping cannot handle EOR and ENOR cells, whose INV-NAND2 decomposed graphs are not a tree, but a DAG (Directed Acyclic Graph).

All nets of an INV-NAND2 tree decomposition are controllable: i.e., by setting adequate values to some inputs all nets can be set to both 0 and 1. Thus values of any nodes within a library cell that are not used by Step 2, can be arbitrarily set to either a constant value (0/1), or the same (or inverted) value of a some inputs of the cell.

We have proved that if the technology library satisfies requirements of Def. 1, resultant mapping of Step 2 can be replaced with its original path in the circuit by adding INV nodes and/or assigning constant values to unused inputs of mapped library cells. Fig. 2(4) shows adding an INV node j and pulling up one input of the AOI21 cell make the circuit logically equivalent to the circuit of Fig. 2(1).

2.2 Circuit Delay

There are many combinations of library cells to map the subject path. In order to calculate Eq. 1, we try all possible path mappings and calculate its path delay. Path delay is calculated considering number of fanouts, delay parameters of mapped library cells, and other technology-specific delay parameters. The minimum path delay among all possible path mappings is recorded as the delay of the subject path.

The path of the maximum delay is chosen as the most critical path in the circuit, and its delay will be the estimated circuit delay. This estimated circuit delay is guaranteed to be a lower bound of the actual circuit delay after technology mapping. This is because path mapping calculates the fastest technology mapping for each path. Thus circuit delay obtained by technology mapping is always larger than circuit delay obtained by path mapping.

3 Efficient Algorithm

Properties of Eq. 1 and path mapping gives us an efficient algorithm to obtain the circuit delay. We can calculate circuit delay in linear time to the number of nodes.

3.1 Path Pattern Table

In the mapping process described in Section 2, a library cell can be mapped to the subject path, if one of paths in the library cell can be mapped to the subject path. Using a "path pattern table" we can speed up this mapping process.

A "path pattern" is defined as a string of digit 1 and 2, which represents an order of INV and NAND2 nodes on a path. A digit 1 corresponds to an INV node, and 2 to a NAND2 node. For example in Fig. 1, path $A \to X$ has a path pattern of "21212". Path $C \to X$ and $D \to X$ have path patterns of "212" and "22", respectively.

A "path pattern table" is defined as a collection of a pair of a path pattern and its corresponding cell delay. All entries of path patterns must exist in some technology library cell. A path pattern table can be made by generating all INV-NAND2 tree decompositions of all library cells, and calculating path delays of all paths in those INV-NAND2 tree decompositions. Since Eq. 1 only needs the fastest mapping, it is enough for each path pattern in the path pattern table to have only the minimum cell delay.

Technology mapping generates a buffer or/and INV node tree at fanout points with a large number of fanouts in order to distribute fanouting load and speed up the fanout points. At fanout points path mapping considers both buffering and non-buffering cases, and takes the smaller delay.

Table 1 shows statistics of two CMOS libraries: lib2 (a SIS genlib library [5]) and CG51 (a Fujitsu's 0.5 μ CMOS standard cell library [7]). In the table, "Number" means number of path patterns contained by single-output combinational library cells, excluding EOR and ENOR cells. "Maximum Length" means the maximum number of nodes contained by one path pattern. Lib2 has only 23 path patterns. Although

Library	Number	Maximum Length
lib2 (SIS)	23	7
CG51 (Fujitsu)	63	15

Table 1: Path patterns in CMOS libraries

CG51 is a real library and has 240 combinational single-output cells, it has only 63 path patterns. Those small numbers of path patterns helps us to save memory and run time. The length of the maximum path pattern is 7 for lib2, and 15 for CG51. Since time spent by matching process depends on length of path patterns, path mapping is expected to run in short time.

In addition, we need to make a path pattern table only once for each technology library because the path pattern table does not depend on subject circuits.

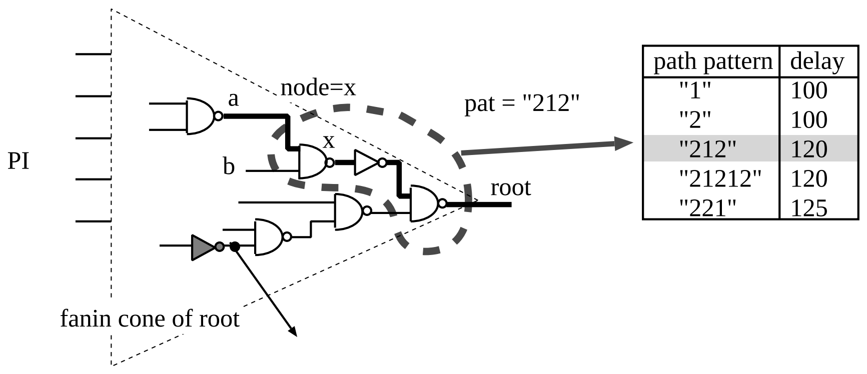
3.2 Dynamic Programming based Arrival Time Calculation

Path mapping can use a dynamic programming paradigm similar to the tree covering based technology mapping[3]. It can avoid to enumerate all paths in the subject circuit, where the number of paths is exponential to the number of nodes at worst.

First, we set initial arrival times at primary inputs of the subject circuit. Then we process internal nodes one by one in a topological order from primary inputs toward primary outputs, and calculate an arrival time of the node. After all nodes are processed, the maximum arrival time among primary outputs is selected as the circuit delay.

Fig. 3 shows an example of arrival time calculation of node $root$. Since we calculate arrival times of nodes in topological order, we have already processed all nodes in the transitive fanin cone of $root$.

In order to calculate the arrival time at $root$ we trace all paths ended at $root$ by visiting nodes in the fanin cone in a depth-first order. We introduce two variables, $arrvTime$ and $minPath$. $arrvTime$ is initialized with $-\infty$, and answers the arrival time at $root$ after we have visited all nodes in the



path pattern	delay
"1"	100
"2"	100
"212"	120
"21212"	120
"221"	125

Figure 3: Arrival time calculation at “root”

fanin cone. $minPath$ is initialized with $+\infty$, and holds the minimum arrival time of all intermediate nodes on the path from the currently visiting node to $root$.

Suppose we currently visit node x in the fanin cone. We try to map the path $x \rightarrow root$ with a single library cell. We can easily check this trial by searching the path pattern table. If an entry of the path pattern of path $x \rightarrow root$ is found, we know the path can be mapped and its minimum cell delay from the path pattern table. The arrival time of $root$ with this mapping is calculated by the following equation:

$$\text{(Arrival time at } x) + \text{(Delay of path pattern } x \rightarrow root) \quad (2)$$

We substitute $minPath$ with the value of Eq. 2, if it is less than $minPath$. This means we find the fastest mapping ever examined on the path $x \rightarrow root$.

We call a node in the fanin cone a “leave node”, if the node is either a fanout point or at a distance of the length of the maximum path pattern apart from $root$. We don't need to visit any nodes beyond a leave node because no cells can be mapped across the leave node.

When we reach to a leave node, we substitute $arrvTime$ with $minPath$, if $minPath$ is larger than $arrvTime$. Note that this arrival time calculation leads us to further reduction of searching space, i.e., when the arrival time of a currently visiting node is less than $arrvTime$, we don't need to visit any nodes beyond the currently visiting node.

According to discussion above, run time calculating an arrival time of $root$ is bound by a constant value. And an arrival time of each node needs to be calculated only once. Thus time complexity of path mapping is linear to the number of nodes in the subject circuit. Tree covering based technology mapping also has linear time complexity. It is, however, much slower than path mapping because technology mapping examines all matchings between a graph of the subject circuit and all graphs of library cells, while path mapping only has to search an entry of a path pattern in the path pattern table.

4 Heuristics for More Accurate Delay Estimation

Path mapping uses an inverter heuristic as tree covering based technology mapping [3] does. Adding a pair of inverters to every net, it considers both positive and negative phases of library cell mappings.

Since path mapping considers the fastest technology mapping for a subject path, it ignores delay from side inputs of the subject path, and it often answers too small circuit delay. Fig. 4 illustrates two cases of mapping a NAND4 cell. Circuit (1) of the figure has only one critical path. In this case technology mapping may use a cascade INV-NAND2 decomposition of NAND4 in order to cover as many nodes on the critical path as possible. This contributes to speed up the critical path.

On the other hand, Circuit (2) of the figure has all paths equally critical. In this case technology mapping may use a balance INV-NAND2 decomposition of NAND4 in order to speed up all paths equally. If technology mapping uses a cascade decomposition instead, it speeds up only one path, but makes all other paths slow down. As a result the circuit delay obtained by a cascade decomposition becomes larger than one obtained by a balance decomposition. This is why technology mapping does not use a cascade decomposition to Circuit (2).

Path mapping mentioned in Section 2 maps each path as fast as possible. In other words, path mapping always uses cascade INV-NAND2 decompositions. For the Circuit (2) in Fig. 4 path mapping applies a cascade decomposition, while technology mapping never does so. Moreover, it ignores arrival times from side inputs. This results that path mapping answers smaller circuit delay than actual mapping.

In order to avoid this problem, we use both cascade and balance decompositions for each library cell. And we use either of them according to difference of arrival times at NAND2's inputs. In Fig. 4, for example, if difference of arrival times at input A and B is large, a cascade decomposition

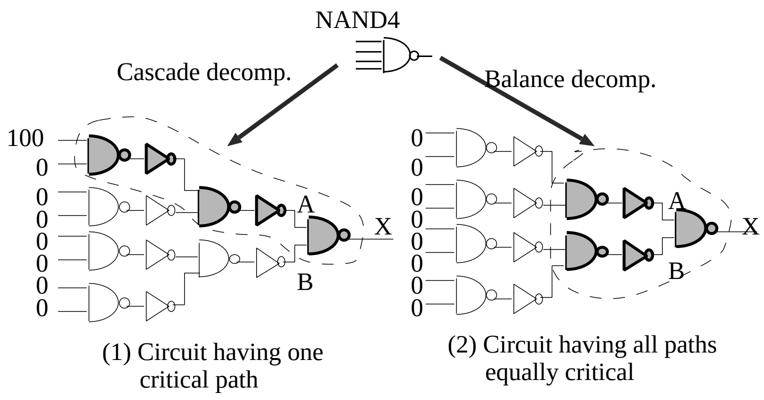

NAND4
Cascade decomp.
Balance decomp.
100
0
A
B
X
(1) Circuit having one critical path
(2) Circuit having all paths equally critical

Figure 4: Cascade decomposition and balance decomposition

is applied, and a balance decomposition is applied otherwise.

Note that this heuristic does not any more guarantee that path mapping gives a lower bound of the actual circuit delay.

5 Experiments

We have applied path mapping with heuristics discussed in Section 4 to 27 combinational MCNC benchmark circuits [6]. These circuits are optimized with script.rugged by SIS[5] twice. In addition, we get another 27 circuits by applying timing optimization script[1] of "speed_up -i; speedup_alg 2c_kernel divisor; speed_up -T -m unit - d 3". Note that this timing optimization uses unit delay model, not path mapping.

We measure circuit delays of those 54 circuits by technology mapping, path mapping, and unit delay. We use lib2 and CG51 for technology libraries. Technology mapper is SIS by the command of "map -AFG -n 1" for lib2, and our boolean matching based in-house mapper for CG51. We cannot use the SIS mapper for CG51 because CG51 employs a complicated delay model the SIS mapper cannot handle.

We don't distinguish rise and fall signal transitions because it does not significantly improve accuracy of delay estimation, but doubles run time. In the path pattern table, we record the average delay of all signal transitions.

Table 2 shows results of lib2 and CG51. The second column shows circuit names, where the suffix of ".opt" means the circuit is optimized by the timing script. The third column shows delays obtained after technology mapping. The fourth and fifth columns represent estimated delays by path mapping and unit delay, respectively. The last two columns mean CPU times of technology mapping and path mapping on a SUN 4/20 Workstation (60 MHz). Numbers in the parentheses are ratios of those CPU times.

Path mapping runs more than 150 times faster than technology mapping for almost all circuits. This means path mapping is enough fast to be used iteratively in technology independent synthesis. Path mapping says C1908 is 10 % faster than C880 with CG51. However, technology mapping says those circuits have almost the same delay. We need better heuristics to this problem.

Table 3 compares accuracies of path mapping and unit delay. In the table, we assume a linear relationship between estimated and actual (after technology mapping) delays, as follows:

$$\text{(Actual Delay)} = K * \text{(Estimated delay)} \qquad (3)$$

Coefficient K is a constant that depends both on a technology mapper and a technology library. We calculate K with regression analysis for both libraries of lib2 and CG51. Then we calculate average errors and maximum errors of Eq. 3.

According to Table 3 path mapping has K of 1.01 for lib2. This means path mapping predicts almost the same circuit delay as the SIS mapper. For CG51 path mapping has K of 1.54. This 54 % difference is caused by the complicated delay model of CG51 and our in-house technology mapper. With these K's we can predict circuit delay with average error of 5.79% and 10.57% for lib2 and CG51, respectively. Comparing with unit delay in terms of average and maximum errors, path mapping is more accurate both for lib2 and CG51.

6 Conclusion

This paper proposes "path mapping" as a both fast and accurate delay estimation for technology independent combinational circuits. According to our experimental results, we conclude path mapping is more accurate than unit delay, and much faster than technology mapping. We consider that technology independent synthesis using path mapping can estimate more accurate circuit delays and get better results with small increase of run time.

In our future work, we will study other heuristics to improve accuracy of path mapping. And we will also incorporate path mapping into technology independent timing synthesis, and evaluate its efficiency.

Lib	Ckt	Circuit Delay			CPU Time [s]	
		Tech Map [ns]	Path Map [ns]	Unit [levels]	Tech Map	Path Map (∗)
lib2	C499	19.23	17.83	19	26.2	0.141 (186)
	C499.opt	17.97	15.77	15	63.3	0.162 (391)
	C880	31.00	31.93	35	23.5	0.123 (191)
	C880.opt	19.60	19.85	20	59.6	0.180 (331)
	C1355	19.23	17.83	19	27.1	0.129 (210)
	C1355.opt	17.97	15.77	15	63.2	0.160 (395)
	C1908	27.83	29.54	30	26.4	0.138 (191)
	C1908.opt	24.96	23.64	23	61.7	0.146 (423)
	C5315	26.91	30.10	32	98.5	0.517 (191)
	C5315.opt	22.18	21.98	21	210.5	0.573 (367)
	C6288	94.02	95.62	92	166.6	0.605 (275)
	C6288.opt	76.74	70.31	68	374.7	0.800 (468)
CG51	C499	3.99	2.74	19	138.1	0.378 (365)
	C499.opt	3.62	2.47	15	188.2	0.454 (415)
	C880	6.24	5.09	35	210.1	0.368 (571)
	C880.opt	3.88	2.99	20	192.1	0.538 (357)
	C1355	3.99	2.74	19	157.7	0.363 (434)
	C1355.opt	3.62	2.47	15	188.3	0.436 (432)
	C1908	6.35	4.35	30	184.1	0.364 (506)
	C1908.opt	5.00	3.46	23	179.8	0.566 (318)
	C5315	6.58	4.58	32	700.2	1.462 (479)
	C5315.opt	4.62	3.23	21	665.1	1.667 (399)
	C6288	22.70	13.33	92	783.4	2.165 (362)
	C6288.opt	16.00	10.10	68	1066.1	2.257 (472)

∗: Ratio of CPU times of technology mapping and path mapping

Table 2: Results of lib2 and CG51

Library	Delay Model	K of Eq.3	Ave. Error (%)	Max. Error (%)
lib2	Path Mapping	1.01	5.79	23.34
	Unit Delay	1.01	10.57	31.26
CG51	Path Mapping	1.54	9.56	30.07
	Unit Delay	0.227	12.80	40.91

Table 3: Comparison of delay estimations

References

[1] K. J. Singh, et al. "Timing Optimization of Combinational Logic", Proc. of ICCAD-88, pp. 282–285, 1988.

[2] D. E. Wallace, M. S. Chandrasekhar, "High-Level Delay Estimation for Technology-Independent Logic Equations", Proc. of ICCAD-90, pp. 188–191, 1990.

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

[4] H. J. Touati, "Performance-Oriented Technology Mapping", pp.31–90, Memorandum No.UCB/ERL M90/109, Univ. of California, Berkeley, 1990.

[5] E. M. Sentovich, et al., "SIS: A System for Sequential Circuit Synthesis", Memorandum No.UCB/ERL M92/41, Univ. of California, Berkeley, 1992.

[6] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0", MCNC International Workshop on Logic Synthesis, 1991.

[7] Fujitsu Semiconductor, "CMOS Gate Array CG51 Series, Unit Cell Specification", 1994.