Hierarchical Timing Analysis considering Global False Path

Sunik Heo and Juho Kim
Department of Computer Science,
Sogang University C.P.O. Box 1142, Seoul, Korea

Abstract

As the integrated circuit technology gets developed, a circuit size of more than thousands of transistors becomes normal. A hierarchical design is unavoidable due to a huge circuit size. It is important how we can consider hierarchical structure in circuit delay analysis. In this paper we present an accurate method to analyze the delay of circuit with hierarchical structure. Adding the notion of global false path to the hierarchical timing analysis performs more accurate timing analysis.

1. Introduction

A flat analysis is accurate but not efficient in run time since it does not consider hierarchical structure in timing analysis [2]. In addition, it has disadvantage; whenever we modify the circuit, we have to re-analyze the whole circuit. On the contrary, hierarchical timing analysis is very efficient because it is sufficient to analyze just once when the same module is used repeatedly. Also, we have only to re-characterize one module not the whole circuit in case of modification. A hierarchical timing analysis is composed of two steps. In the first step, we characterize leaf modules. In the second step, we compute the circuit delay by using this information [3].

2. Hierarchical Timing Analysis considering Global False Path

We redefine notions of local false path and global false path. If some path is false and included entirely in a module then we call it local false path [4]. Also, if two or more paths are true in each module but not true totally, in other words, if some path becomes false due to connection of modules, then we call it global false path.

A previous hierarchical timing analysis is not accurate in analyzing the circuit delay since it considers only local false paths [3]. In this paper, we calculate accurate delay by applying a global false path to timing analysis. In previous method, input vectors were not used in computing a circuit delay. As a result, many side effects occurred. What is worse, a huge delay over-estimation occurred.

Let's consider the figure 1 for example. Suppose that the delay of MUX is 2 and the delay of INVERTER is 1. Let M1, M2 be leaf modules. The topologically critical delay then becomes 6(3+3). In previous hierarchical timing analysis [3], it is also 6 because no false paths exist in each module M1 and M2. However, the true circuit delay is 5 since the topological critical path is a global false path. In above example we can see that we should consider a global false path to get more accurate circuit delay.

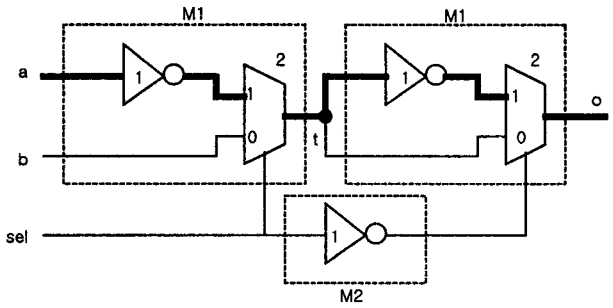

Figure 1 The circuit with global false path

In this paper we use the known X function and the XBD0 model (extended bounded delay-0 model) [1]. We use the library to get gate delay. The X-function $\chi^{t0}_{n,v}$ is a conditional function for a node n to have stable logic signal v (0 or 1) at given time t0 [1].

If x is an input of module then we define X-function such as

$$\text{if } \tau \geq arr(x) \quad \chi^{\tau}_{x,1} = x$$
$$\text{otherwise} \quad = 0$$

$$\text{if } \tau \geq arr(x) \quad \chi^{\tau}_{x,0} = \bar{x}$$
$$\text{otherwise} \quad = 0$$

For comprehension, consider figure 2.

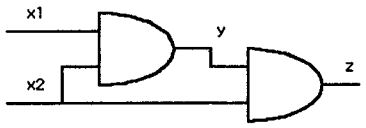

Figure 2 Two AND gate

$$\chi^{1}_{z,1} = \chi^{0}_{y,1} \bullet \chi^{0}_{x_2,1}$$
$$= \chi^{-1}_{x_1,1} \bullet \chi^{-1}_{x_2,1} \bullet \chi^{0}_{x_2,1}$$
$$= 0 \bullet 0 \bullet x_2$$
$$= 0$$

$$\chi^{1}_{z,0} = \chi^{0}_{y,0} + \chi^{0}_{x_2,0}$$
$$= \chi^{-1}_{x_1,0} + \chi^{-1}_{x_2,0} + \chi^{0}_{x_2,0}$$
$$= 0 + 0 + \bar{x}_2$$
$$= \bar{x}_2$$

The value 0 means that node z cannot be stable at time 1 with logic value 1. Also, $\bar{x}_2$ means that node z have stable logic value 0 at time 1 when a primary input *x2* have logic value 0.

Apply this function to Figure 3.

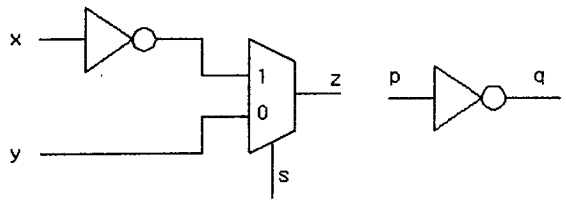

Figure 3 Module M1 and M2

$$\chi^{3}_{z,1} = \chi^{1}_{s,1} \bullet \chi^{1}_{a,1} + \chi^{1}_{s,0} \bullet \chi^{1}_{y,1}$$
$$= \chi^{1}_{s,1} \bullet \chi^{0}_{x,0} + \chi^{1}_{s,0} \bullet \chi^{1}_{y,1}$$
$$= s \bullet \bar{x} + \bar{s} \bullet y$$

$$\chi^{3}_{z,0} = \chi^{1}_{s,1} \bullet \chi^{1}_{a,0} + \chi^{1}_{s,0} \bullet \chi^{1}_{y,0}$$
$$= \chi^{1}_{s,1} \bullet \chi^{0}_{x,1} + \chi^{1}_{s,0} \bullet \chi^{1}_{y,0}$$
$$= s \bullet x + \bar{s} \bullet \bar{y}$$

$$\chi^{1}_{q,0} = \chi^{0}_{p,1} = p$$

$$\chi^{1}_{q,1} = \chi^{0}_{p,0} = \bar{p}$$

We could obtain a contradiction on primary input *sel*. For the topologically longest path being true, logic value of the primary input *sel* should be 0 at second module M1 (because it is propagated through module M2) and 1 at first module M1. Then the value of the *X* function becomes 0. It means that the topologically longest path is a global false path.

We characterize each leaf module by applying *X* function. After we propagate backward the *X* function for the critical path, and a contradiction occurred in primary input relations, we regard it as false path. We can consider the global false path by using *X* function and we can obtain more accurate circuit delay. If the function *fn* is an expression of primary input of node n then the *X* function is generalized such as

$$\chi^{t0}_{n,v} = \sum_{p \in P^{v}_{n}} [\prod_{m_i \in p} \chi^{t0-d(n)}_{m_i,1} \bullet \prod_{m_i \in p} \chi^{t0-d(n)}_{m_i 0}]$$

P^{1}_{n} is a prime set of *fn* and P^{0}_{n} is prime set of $\bar{fn}$. In two input *AND* gate, P^{1}_{n} becomes $\{m_1, m_2\}$ and P^{0}_{n} becomes $\{\bar{m}_1, \bar{m}_2\}$. A sigma function is an *OR* and a pi function is an *AND*. As we apply above equation to the circuit recursively, we get the *X* function of node n.

To get the circuit delay, we set the required time of primary output 0. Through the above equation, we obtain the required time of primary inputs. After converting the sign of primary input required time, the largest value among those is the critical path delay.

The overall algorithm of hierarchical timing analysis considering global false path is described in figure 2.

```
for ( every leaf module )
    {while ( output is stable )
        { find topological k-th critical path length ( =t0 )
          set output stable time with t0
          backward propagation of condition for output stable at
              t= t0  using χ  function.
          k++ }
      determine pin to pin require time & delay time tuple
    }
 set primary input arrival time t=0
 propagate by topological order using above tuple.
 while (contradiction does not exist in primary input relation)
        { find k-th critical path
          backward propagate condition that make this path be true
              using χ  function.
          k++}
 determine delay of total circuit
```

Figure 4 The Overall algorithm of hierarchical timing analysis considering global false path

3. Experimental results

The table 1 illustrates the result of this paper. We applied our algorithm to various benchmark circuits and carry skip adders. The CSA32(2) means that a 32bit carry skip adder is composed of leaf modules of 2bit carry skip adder. We divided the benchmark circuit randomly. The HA is a hierarchical timing analysis approach without global false path and the HAGF is the hierarchical timing analysis considering global false paths. As you can see in the table, the HAGF is more accurate than HA.

4. Conclusions

In this paper, we applied the concept of the global false path to hierarchical timing analysis for accuracy. We could obtain a reasonable circuit delay by considering global false path. If we have more information of meaningful leaf modules and a proper partition algorithm we can obtain better results.

5. References

1. P. C. McGeer, A. Saldanha, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Delay models and exact timing analysis," In *T.Sasao, editor, Logic Synthesis and Optimization*, pages 167-189. Kluwer Academic Publishers, 1993.

2. H. Yalcin and J. P. Hayes, "Hierarchical timing analysis using conditional delays," In *proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 371-377, November 1995.

3. Y. Kukimoto and R. K. Brayton, "Hierarchical functional timing analysis," In *Proceedings of 35th Design Automation Conference*, pages 580-585, June 1998.

4. Y. Kukimoto and R.K.Brayton, "Exact required time analysis via false path detection," In *Proceedings of 35th Design Automation Conference*, pages 220-225, June 1997.

5. S. Devadas, K. Keutzer, and S. Malik, "Computation of floating mode delay in combinational circuits: Theory and algorithms," *IEEE Transactions on Computer-Aided Design*, 12(12): pages 1913-1923, December 1993.

6. H.-C. Chen and D. H. -C. Du, "Path sensitization in critical path problem," *IEEE Transactions on Computer-Aided Design*, 12(2): pages 196-207, February 1993.

7. K. Keutzer, S. Malik, and A. Saldanha, "Is redundancy necessary to reduce delay?," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(4): pages 427-435, April 1991.

8. D. Brand and V. S. Iyengar, "Timing analysis using functional analysis," *IEEE Transactions on Computers*, 37(10): pages 1309-1314, October 1988.

9. C. -J. Seger, "A bounded delay race model," In *proceedings of IEEE International Conference on Computer-Aided Design*, pages 130-133, November 1989.

10. K. P. Belkhale and A. J.Suess, "Timing analysis with known false sub graphs," In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 736-740, November 1995.

circuit	# of gate	topological	HA	HAGF		Flat analysis	
		delay	delay	delay	CPU time	delay	CPU time
C7522	3274	1482	178	166	189.3	157	352.8
C5315	2123	1315	136	127	103.6	118	162.5
C3540	1321	1280	101	95	58.2	87	83.3
C432	222	88	23	23	7.1	23	2.3
C499	602	124	38	36	11.8	34	12.1
C880	378	108	34	31	7.2	29	3.7
C1908	672	154	45	40	10.5	37	12.9
C2607	1024	1207	78	69	27.3	65	37.8
C829	978	829	69	63	29.8	58	32.5

circuit	topological	HA	HAGF		Flat analysis	
	delay	delay	delay	CPU time	delay	CPU time
CSA32(2)	98	26	26	0.7	26	10.1
CSA64(2)	194	48	48	0.7	48	73.2
CSA128(2)	386	82	82	0.7	82	279
CSA32(4)	82	26	26	1.3	26	8.7
CSA64(4)	162	40	40	1.3	40	67
CSA128(4)	322	74	74	1.3	74	217
CSA32(8)	74	30	30	3.2	30	6.4
CSA64(8)	146	44	44	3.2	44	51
CSA128(8)	290	80	80	3.2	80	167
CSA32(16)	70	34	34	8.3	34	4.8
CSA64(16)	138	48	48	8.3	48	37
CSA128(16)	274	84	84	8.3	84	137

Table 1 Results of our algorithm