Synthesis and Optimization of Multiple Portions of Circuits for ECO based on Set-Covering and QBF Formulations

Masahiro Fujita Yusuke Kimura Xingming Le Yukio Miyasaka Amir Masoud Gharehbaghi
University of Tokyo
Tokyo, Japan

***Abstract*—Engineering Change Order (ECO) and logic debugging problems where multiple locations in the circuit must be modified are formulated with Quantified Boolean Function (QBF) and set-sovering techniques. The formulation is based on the fanin selection method for each gate. Although the resulting formulation for single portion changes is basically equivalent to Sets of Pairs of Functions to be Distinguished (SPFD) [3], the way of its computations is quite different. Moreover, the simultaneous changes for multipl portions becomes Boolean Relation extension of SPFD. Experimental results and applications to various logic optimization problems are also shown.**

***Index Terms*—EDA, logic synthesis, debug, ECO**

I. Introduction

Engineering Change Order (ECO) in logic design [1] [2] happens when the specification changes after its implementation has been generated. In such cases, it may be better to "modify" implementation in such a way that the modified implementation becomes logically equivalent to the new specification, instead of resynthesizing a new implementation from the modified specification, since the resynthesis process may generate structurally and performance-wise (timing, power consumption, and others) very different. Although the resynthesized implementation is logically correct, its performance including timing and power consumption can be very different due to the structural difference from the original implementation,

Logic debugging must be performed when some bug in the specification is found after its implementation has been generated. The situation is similar to ECO, and it is sometimes much better to modify the implementation accordingly to the debugged specification instead of resynthesizing the new implementation from the debugged specification.

In this paper ECO and logic debugging processes are mathematically formulated for combinational circuits or fixed time frame expanded sequential circuits. Current implementations are adjusted by changing a single or multiple target locations in the circuits. That is, first a single or multiple locations are picked up. Then those locations are replaced with another single or multiple sub-circuits so that the resulting implementation becomes logically correct with respect to the modified/new specification. Thus, the structural changes in the implementations can be kept small.

The problem with single location changes is discussed first followed by the case of multiple location changes. The appropriate inputs to the sub-circuit for a single location change are searched out of all internal signals and primary inputs, and a set of inputs with the minimum cost is obtained with set-covering formulation. Once the set of inputs to the sub-circuit is found, the logic function to be realized by the sub-circuit can be searched as Quantified Boolean Formula (QBF) problem.

In the case of multiple location changes, first mathematical formulation is given, although it is computationally very expensive to solve. A simple and fast method is to deal with one location by one location sequentially instead of all together simultaneously. Experimental results are given with various ordering heuristics.

Various application of the proposed method to logic optimization and their experimental results are also given.

The set-covering formulation has been turned to be equivalent to the logic optimization with Sets of Pairs of Functions to be Distinguished (SPFD) [3] [4], although the proposed method has been independently developed. A brief discussion on this issue and the extension of SPFD with Boolean Relation [5] [6] are given.

The paper is organized as follows. In the next section the ECO and debugging problem are mathematically formulated with modifications of single and multiple locations. Then the proposed methods is given in the following section. Discussion with respect to SPFD is also given. Various application of the proposed method including the logic synthesis with multiple-output gates as well as their the experimental results are given. The final section gives concluding remarks and future directions.

II. Problem formulation

There are two situations in terms of mathematical formulation for ECO and logic debugging problems: single target changes and multiple target changes, although the single target change can be considered to be a special case of the multiple target changes. For easy to explain and understand discussions, they are formulated separately.

Figure 1 (a) is the target design to be modified for ECO and logic debugging when we change only one internal location.

978-3-9819263-4-7/DATE20/©2020 EDAA

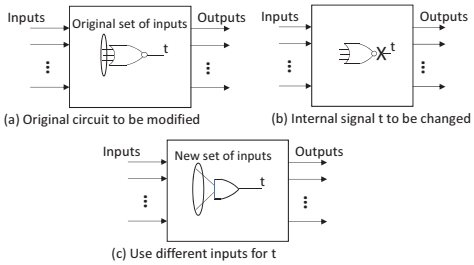


Fig. 1. Target problem with single location

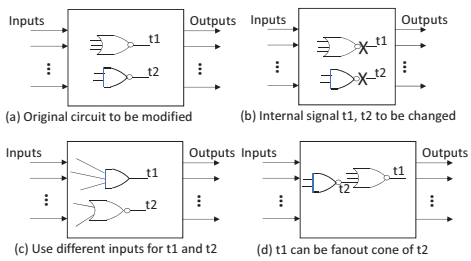


Fig. 2. Target problem with multiple locations

Given a circuit, the target of the gate to be replaced with a gate or sub-circuit is first determined. Such a selection can be made based on signal dependency analysis techniques and is not discussed in this paper. In the figure signal t is the target signal. Please note that there is only one target signal in this case, and the case targeting multiple signals is discussed later. Then signal t is separated from the gate (for example, NOR gate in the figure) as shown in the figure (b). Now the remaining problem is to find a set of inputs to the gate or sub-circuit in general which will be connected to signal t as shown in the figure (c). Once the set of inputs to the sub-circuit is found, there are ways to automatically find the required logic function for the sub-circuits, such as the one through QBF formulation with LUT (Look Up Table) [8], [9].

Corresponding problem for multiple target signals is shown in Figure 2. In the figure there are two target signals, t1 and t2. The way to rectify the circuit is the same as before, but the two signals must be replaced with the new sub-circuits as shown in the figures (a)-(c).

Please note that the functionality to be realized by the sub-circuits for t1 and t2 depends on each other. That is, even if the same sets of inputs are used for t1 and t2, the functionality realized by the sub-circuit for t1 influence the required functionality for the sub-circuit fot t2. If a different function is chosen for the sub-circuit for t1, the function of the sub-circuit for t2 may have to change. This is true not only if one of the two signals is in the fanout code of the other signal, such as the case of the figure (d), but also true when the fanout cones of the two signals somehow overlap.

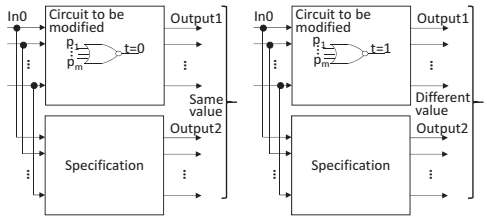


Fig. 3. The case where target signal must be 0

Now let us formulate the case of Figure 1, i.e., single target change. The following formulation is first introduced in [10] [11]. The problem to be solved here is that given a circuit to be modified (for example, the circuit shown in Figure 1 (a)), find a set of inputs for the target signal t (for the example, the set of inputs shown in Figure 1 (c)) in such a way that there exists a logic function of the set of inputs for t by which the entire circuit becomes logically equivalent to the modified/new specification.

There are two sets of primary input values that we need to pay attention to. The first is the set of primary input values under which the value of the target signal t must be 0 in order for the entire circuit to be equivalent to the specification. Such set of values for primary inputs are called "In0" in this paper. The condition for In0 is shown in Figure 3. Under the primary input values of In0, if the target signal t=0, the primary output values are the same as the ones in the specification, and if t=1, the primary output values are different from the ones in the specification.

The other set is the primary input values under which the target signal t must be 1 for the equivalence. Such set of values are called "In1", and the condition for In1 is shown in Figure 4, that is, if the target signal t=1, the primary output values are the same as the ones in the specification, and if t=0, the primary output values are different from the ones in the specification.

In0 and In1 are formally defined as follows. Here the functionality of the specification is given by $Spec(in)$ and that of the implementation is given by $Impl(in, t)$ where in represents the primary inputs and t is the target signal.

$\exists In0, t.(\bar{t} \wedge Impl(In0, t) = Spec(In0))$
$\wedge(t \wedge Impl(In0, t) \neq Spec(In0))$.
$\exists In1, t.(t \wedge Impl(In1, t) = Spec(In1))$
$\wedge(\bar{t} \wedge Impl(In1, t) \neq Spec(In1))$.

If there are don't cares existing in the specification, equality and non-equality above should be based on them. That is, when the primary input values correspond to don't cares, the implementation is always equal to the specification up to the don't cares.

Please note that the definition and notation above are a little bit different from [10] [11]. Then the condition of the existence of a logic function for the target signal t becomes the following:

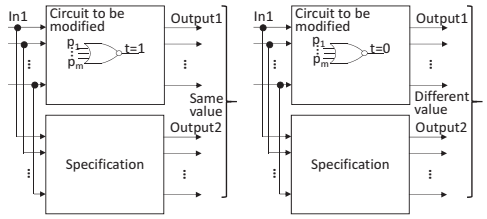


Fig. 4. The case where target signal must be 1

Condition of function existence

For all pairs from In0 and In1, at least one of the inputs to the sub-circuit for t must have different values, as t must produce different values for all of those pairs.

This is a necessary and sufficient condition for the existence of a logic function for t.

So far single target cases, such as the case shown in Figure 1, have been discussed. For multiple target cases, such as the two target case shown in Figure 2, their mathematical formulation is simply extensions for multiple target signals, such as t1 and t2 in the case of Figure 2. It is, however, much more complicated as the required set of inputs for t1 and the required inputs for t2 depend on each other, i.e., depending on the selected inputs for t1, the condition for t2 changes. Because of this, the exact condition for the multiple target singles must include the conditions for all value combinations (which are exponentially many) of the multiple target signals. Therefore, if the number of the target signals becomes large, the size of the entire condition can be very large and become impossible to solve. In such cases, practically the target signals must be processed one by one. Suppose there are n of target signals, t1, t2, ..., tn. First the processing order of t1, t2, ..., tn is determined. Here let us suppose the order is the original order which is t1, t2, ..., tn. Then all signals but t1 are quantified out, and the single target method discussed above is applied. With its result, signal t is replaced with a sub-circuit in the original circuit (the circuit before the quantifying out operations). That has target singles, t2, ..., tn. This process is repeated for all of ti's.

Although this method can find a set of solutions for all the target signals, t1, t2, ..., tn, the solutions fully depend on the order of processing t1, t2, ..., tn. So it is very important to use "good" processing order, and that will be discussed in the section on the experiments below.

III. Proposed methods

Our method is to formulate the searching problem for the sets of inputs to the sub-circuit as a set-covering problem. Let us discuss the proposed method through an example circuit shown in Figure 5. It is the smallest ISCAS85 benchmark circuit, called C17, and the original circuit is shown in the figure (a). We assume this is the target specification. Please

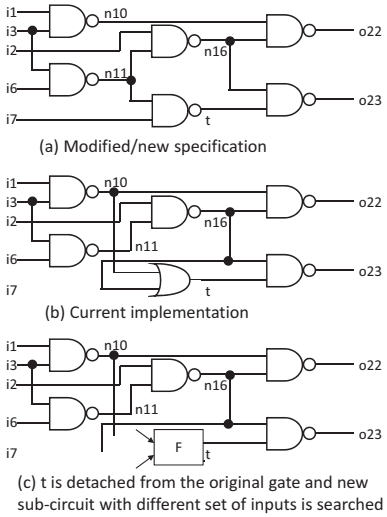


Fig. 5. Example of ECO

note that although the specification is given as a logic circuit, no internal structure is analyzed in the specification. Only the primary input-output relations are used in the following. Therefore, specification can be given in any form as long as that is logically analyzable.

Let us suppose the circuit shown in the figure (b) is given as the current implementation. That is, the circuit in the figure (b) must be debugged/modified so that it becomes logically equivalent to the specification of the figure (a). As can be seen from the figures, the target signal t has a wrong set of inputs. n11 and i7 are the correct inputs for the sub-circuit for t, but currently n10 and n16 are the inputs. So the target signal t is detached from the OR gate in the figure (b), and appropriate sets of inputs and appropriate logic functions for the sub-circuit for t are searched.

Examples of In0 and In1 for the example are shown in Figure 6. When the primary inputs, (i1,i2,i3,i6,i7) are (1,0,0,0,1), if t=0 both of the two primary outputs, o22, o23 are correct whereas if t=1, one of the primary output, o23 becomes incorrect, as shown in the figure (a). Similarly, when the primary inputs, (i1,i2,i3,i6,i7) are (0,1,1,1,1), if t=1, both of the two primary outputs, o22, o23 are correct whereas if t=0, one of the primary output, o23 becomes incorrect, as shown in the figure (b).

These In0 and In1 give a necessary condition on the existence of the logic function for the sub-circuit for t, i.e., at least one of the inputs to the sub-circuit must have different values on In0 and In1. If the two tables shown in Figure 6 are compared, the signal n11 has different values for In0 and In1. Therefore, n11 becomes the candidate for the input to the sub-circuit for t.

Unfortunately n11 alone cannot guarantee the existence of

i1	i2	i3	i6	i7	n10	n11	n16	t	o22	o23	
1	0	0	0	1	-	-	-	-	0	1	Correct outputs by specification
1	0	0	0	1	1	1	1	0	0	1	Signal values when t=0
1	0	0	0	1	1	1	1	1	0	0	Signal values when t=1

(a) Example of In0

i1	i2	i3	i6	i7	n10	n11	n16	t	o22	o23	
0	1	1	1	1	-	-	-	-	0	0	Correct outputs by specification
0	1	1	1	1	1	0	1	0	0	1	Signal values when t=0
0	1	1	1	1	1	0	1	1	0	0	Signal values when t=1

(b) Example of In1

Fig. 6. Examples of In0 and In1

	i1	i2	i3	i6	i7	n10	n11	
a	1	1	1	1	0	0	1	In0=10001, In1=01111
b	0	0	0	0	1	0	0	In0=10001, In1=10000
c	0	1	1	1	0	0	1	In0=00001, In1=01111
d	1	0	0	0	1	0	0	In0=00001, In1=10000

Fig. 7. Examples of covering table

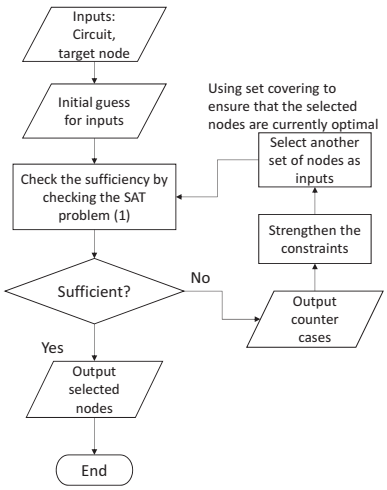


Fig. 8. Overall of the proposed processing flow

the function. If the input to the sub-circuit for t is only n11, there is no way to rectify the implementation, which can be made sure, for example, by the method shown in [8] [9]. That can also be checked by the following SAT problem:

$\exists In0, In1, t.((\bar{t} \wedge Impl(In0,t) = Spec(In0))$
$\wedge(t \wedge Impl(In0,t) \neq Spec(In0))$
$\wedge(t \wedge Impl(In1,t) = Spec(In1))$
$\wedge(\bar{t} \wedge Impl(In1,t) \neq Spec(In1)) \Rightarrow p(In0) \neq p(In1)$ ...(1)

where p represents the values of the current set of inputs to the sub-circuit for t (For the example above, p is just n11) for a primary input value. If this formula is UNSAT, the current set of inputs are sufficient for the existence of a logic function for the sub-circuit for t. If this is SAT, there are more required cases for In0 an In1 which are not yet covered by the In0 and In1 so far accumulated.

By solving the above SAT problem (1) for the example, the following additional In0 and In1 can be found, as (1) becomes SAT:
In0: (i1,i2,i3,i6,i7)=(0,0,0,0,1). In1: (i1,i2,i3,i6,i7)=(1,0,0,0,0). Now there are two of In0 and two of In1, and a set-covering problem can be defined to find another necessary set of inputs to the sub-circuit for the target signal t, as shown in Figure 7.

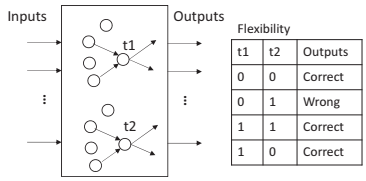
The covering table has four rows which correspond to all combinations of In0 and In1. Those four rows correspond to In0={10001,00001}, in1={01111,10000}: a(10001,01111), b(10001,10000), c(00001,01111), d(00001,10000). For each cell in the table, if the values are different between the two primary inputs (In0 and In1), it is marked as 1. Otherwise it is marked as 0. Now the set-covering problem is to have a minimum set of columns (signals) which cover all rows (for every row, one cell must be marked as 1). In the case of Figure 7, one of minimum covers is (i7,n11). With these signals as p in the above SAT problem (1), it becomes UNSAT. Hence (i7,n11) are the minimum set of inputs to the sub-circuit for the target signal t.

Overall processing flow becomes the one shown in Figure 8. For the case of multiple target signals, as discussed above, the exact formulation becomes exponentially large with respect to the number of target signals. So each target signal is processed one by one with a predetermined order, which may not reach the global optimum. How to decide order or how to process multiple target simultaneously is still an issue to be researched.

The covering table just discussed has basically the same information on the freedom of circuit transformations as Sets of Pairs of Functions to be Distinguished (SPFD) [3]. Each row in a covering table shows the set of pairs of functions to be distinguished. During the selection process of the inputs to the sub-circuit for the target signal t, the covering table grows each time the SAT problem (1) becomes SAT. That is the covering table before the SAT (1) becomes UNSAT corresponds to a subset of SPFD for the target signal t. Also, the algorithm discussed above is to compute the SPFD for the target signal t in a pinpoint way whereas the algorithm to compute SPFD in [4] is incremental following the topological order from the primary outputs.

From the viewpoint of the logic optimization using the SPFD or our covering table, the two algorithms, one discussed above which directly computes SPFD for the target signal, and the other shown in [4] which incrementally and topologically computes SPFD, may be combined for efficient processing. How to combine them would be an interesting problem but is beyond the scope of this paper.

Also, in the case of multiple target signals, the exact freedom for selecting sets of inputs to the sub-circuits becomes Boolean Relation [5] [6] instead of don't cares or simple



Flexibility

t1	t2	Outputs
0	0	Correct
0	1	Wrong
1	1	Correct
1	0	Correct

Fig. 9. Need formulation with Boolean Relation

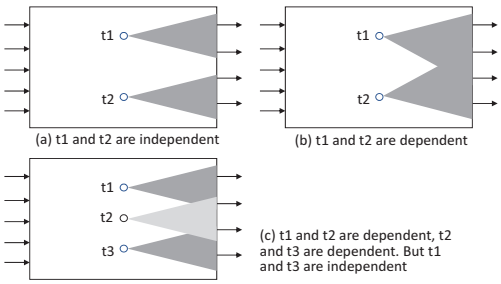


Fig. 10. Target signals may be grouped together

SPFD. Figure 9 shows an example of Boolean Relation. Here there are two target signals, t1 and t2. The allowed values of t1 and t2 for some primary input values are shown in the table of the figure. They are the values by which all of the primary output values become equivalent to the specification with the primary input values. From the table, if (t1,t2)=(0,0), (1,0) or (1,1), the primary output values are correct whereas if (t1,t2)=(0,1), the primary output values become wrong. This relation cannot be captured as don't cares nor SPFD, but can be represented as Boolean Relation. Under Boolean Relation, the allowed values for the target signals must be listed up instead of merging as don't cares. Logic optimization under Boolean Relation has been explored for two-level in [5] and for multi-level in [6]. For the input selection problem, SPFD or covering table must be extended to be able to deal with Boolean Relation version of them.

When dealing with multiple targets, depending on the dependence of the fanout cones of the target signals, more efficient processing order can be considered. Basically if the two targets fanout cones are independent as shown in Figure 10 (a), they can be simply processed one by one in any order. On the other hand, if they are dependent as shown in the figure (b), processing order influences the results. Moreover, when some of the target are depending and some are independent, such as the case of the figure (c), a subset of targets which are independent each other can be processed in any order. When determining the processing order of multiple targets, it is important to take these situations into account as discussed in the experiments shown later.

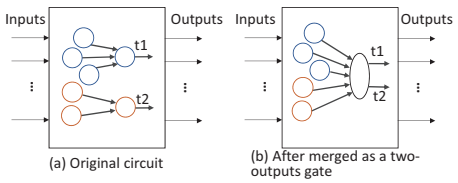


Fig. 11. Logic synthesis with multiple-outputs gates

IV. Various application

The proposed method gives ways to perform ECO and logic debugging as has been discussed, and it can also be applied to logic optimization in general. By changing the sets of inputs to gates in the given circuit, circuit topology can be largely modified which may lead to more optimized circuits.

Apart from regular ways to optimize multi-level logic circuits, the proposed method may work more effectively on the following logic optimization problems.

- Optimization of LUT based circuits, such as FPGA
- Logic synthesis and optimization with multiple-outputs gates

With the proposed method minimum sets of inputs to LUTs in networks of LUTs or FPGA circuits can be searched out of all primary inputs and internal signals. With associated covering tables, minimum numbers of inputs to LUTs can be found.

The proposed method can also efficiently find a set of gates which can be merged as a single gate with multiple outputs. Logic synthesis has been studied for a long time, but almost all of such effort is spent for circuits having only single output gates or cells. For more effective synthesis, however, it may be better to try to use multiple-output gates or cells. For example, integer array multipliers consist of a number of adders which are considered as multiple-outputs gates. The proposed method can find opportunities for multiple single-output gates to be merged as a multiple-outputs gate as shown in Figure 11. By solving the set-covering problem appropriately the circuit transformation shown in the figure can be performed.

V. Experimental results

Preliminary experimental results have been shown in [10] [11]. Here additional results are presented for:

- Various processing order of multiple target signals
- Fanin (numbers of inputs to gates) minimization for FPGA
- Logic optimization for multiple-output gates

The result of various processing orders of multiple target signals on the benchmarks used in ICCAD 2017 CAD contest [7] are shown in Figure 12. The results are measured by the cost function (weights) given by the CAD contest. The smaller the better. Three results are reported here for the multiple target signals. The first one (columns 2-5) is the result by processing all targets simultaneously. Please note that this

Circuit	Simultaneous processing				One-by-one (optimum)				One-by-one (heuristics)			
	#Itera-tions	Weight	Total time (s)	Ser-cover time (s)	#Itera-tions	Weight	Total time (s)	Ser-cover time (s)	#Itera-tions	Weight	Total time (s)	Ser-cover time (s)
Unit5	899	47	11	0	621	47	24	0	418	59	25	0
Unit6	753	N/A	>7,200	69,181	150	142	524	22	150	142	504	0
Unit9	870	50	312	3,031	1,008	358	41	16	1,559	445	27	4
Unit10	647	135	122	0	1,124	135	62	0	1,147	135	57	0
Unit11	42,528	N/A	>7,200	6,671	41,102	N/A	>7,200	6,129	2,877	710	472	17
Unit14	209	94	294	0	105	1,932	474	1	72	1,970	431	0
Unit16	527	204	203	191	165	360	12	0	223	468	10	0
Unit17	257	434	27	6	205	440	44	0	367	716	39	0
Unit19	14,653	4,104	7,204	5,592	2,148	20,258	2,989	151	1,049	16,994	3,007	150
Unit20	912	120	43	10	2,248	302	146	88	1,015	170	48	1

Fig. 12. Logic synthesis with multiple-outputs gates

Circuit	Original				After Optimization		
	# LUTs	# LUT5s	# LUT6s	# LUT5s*	# LUT5s	# LUT6s	# LUT5s*
spi	902	621	281	1183	742	160	1062
des_area	991	260	731	1722	486	505	1496
wb_dma	997	824	173	1170	867	130	1127
systemcaes	1916	1311	605	2521	1595	321	2237
tv80	1988	1355	633	2621	1597	391	2379
mem_ctrl	2445	1766	679	3124	1994	451	2896
ac97_ctrl	2785	2133	652	3437	2166	619	3404
usb_funct	3236	2153	1083	4319	2345	891	4127
aes_core	4577	2047	2530	7107	3508	1069	5646
pci_bridge32	5038	3669	1369	6407	4038	1000	6038
des_perf	6094	4147	1947	8041	4370	1724	7818
wb_conmax	10855	7481	3374	14229	8093	2762	13617
ethernet	12170	11431	739	12909	11559	611	12781
Avg.				1			0.9

* LUT6 -> 2 X LUT5

Fig. 13. Logic synthesis with multiple-outputs gates

needs exponentially many case analysis, and because of that, it takes much longer time, and for some benchmarks it does not finish. The second one (column 6-9) shows the result by trying all possible orders of multiple targets. The last one (column 10-13) is the result using a heuristic for ordering the multiple targets.b The simultaneous method always gives the best results if it ever finished. That is the reason why the heuristic method sometimes gives better results. There are still rooms for improvements.

The proposed method has also been applied to logic optimization targeting networks of LUTs, such as FPGA. The given circuits are first converted into networks of LUTs by using the logic synthesis and verification tool, ABC [12] through appropriate scripts. Then the circuits are further optimized by reducing the numbers of inputs to LUTs by the proposed method. As can be seen from the experimental results shown in Figure 13, 10% of further reduction is observed in average. The area is measured normalized to the size of 5-input LUT.

With the proposed method, multiple single-output gates are tried to be merged as a single multiple-outputs gate. The experimental results are shown in Figure 14. As can be seen from the results, significant numbers of single-output gates can be merged.

VI. Conclusions and Future Direction

The selection method for sets of inputs to gates in a circuit has been discussed with application to not only ECO and logic debugging, but also FPGA circuit optimization and logic

Circuit	Original #gates	#Gates after merging	Ratio	Time (sec)
c880	98	72	0.73	20
c1355	67	52	0.78	17
c1908	108	74	0.69	20
c2670	164	126	0.77	67
c5315	343	302	0.88	700
c6288	612	251	0.41	9,961
c7552	409	238	0.58	785
s6669	577	309	0.54	737
s9234	537	350	0.65	690
s15850	1,117	730	0.65	7,874
Average			0.67	

(a) Circuits with 5-input 3-output gates

Circuit	Original #gates	#Gates after merging	Ratio	Time (sec)
c880	89	56	0.63	19
c1355	70	46	0.66	15
c1908	95	53	0.56	20
c2670	120	88	0.73	69
c5315	279	215	0.77	630
c6288	524	215	0.41	10,139
c7552	351	181	0.52	745
s6669	534	270	0.51	511
s9234	475	204	0.43	607
s15850	1,028	525	0.51	7,334
Average			0.57	

(b) Circuits with 6-input 4-output gates

Fig. 14. Logic synthesis with multiple-outputs gates

synthesis with multiple-outputs gates. The methods use the same freedom of SPFD and the optimum sets of inputs to gates can be obtained by solving set-covering problem.

For the case with multiple targets, there can be more effective ways to deal with, which is one of the future issues. Also, the way to compute SPFD equivalent freedom of circuit transformations may become more efficient by combing the set-covering formulation and the method discussed in [4], which is also another future issue.

References

[1] Daniel Brand, Anthony Drumm, Sandip Kundu, Prakash Narain: Incremental synthesis *IEEE/ACM international conference on Computer-aided design*, 1994.

[2] M. Fujita, T. Kakuda, Y. Matsunaga: Redesign and Automatic Error Correction of Combinational Circuits, *Logic and Architecture Synthesis, ed. G. Saucier*, North-Holland: Elsvier Science Publishers B.V., pp. 253-262.

[3] S. Yamashita, H. Sawada, and A. Nagoya: A new method to express functional permissibilities for LUT based FPGAs and its applications, *International Conference on Computer Aided Design (ICCAD)*, pp. 254–261, Nov. 1996.

[4] S. Sinha and R. K. Brayton: Implementation and use of SPFDs in optimizing Boolean networks, *International Conference on Computer Aided Design (ICCAD)*, pp. 103–110, Nov. 1998.

[5] Y. Watanabe, L. Guerra, and R. K. Brayton: Logic optimization with multi-output gates, *International Conference on Computer Design (ICCD)*, pp. 416–420, Oct. 1993.

[6] K.C. Chen, M. Fujita: Network Optimization Using Don't-Cares and Boolean Relations, In: Sasao T. (eds) Logic Synthesis and Optimization, The Kluwer International Series in Engineering and Computer Science (VLSI, Computer Architecture and Digital Signal Processing), vol 212. Springer, Boston, MA, 1993.

[7] Ching-Yi Huang, Chih-Jen Hsu, Chi-An Wu, Kei-Yong Khoo: ICCAD-2017 CAD contest in resource-aware patch generation, *International Conference on Computer Aided Design (ICCAD)*, Nov. 2017.

[8] Masahiro Fujita, Satoshi Jo, Shohei Ono, Takeshi Matsumoto: Partial synthesis through sampling with and without specification, *International Conference on Computer Aided Design (ICCAD)*, pp.787-794, Nov. 2013.

[9] Masahiro Fujita: Toward Unification of Synthesis and Verification in Topologically Constrained Logic Design. *Proceedings of the IEEE* 103(11): 2052-2060 (2015).

[10] Amir Masoud Gharehbaghi, Masahiro Fujita: A New Approach for Debugging Logic Circuits without Explicitly Debugging Their Functionality, *Asian Test Symposium*, pp.31-36, Nov. 2016.

[11] Amir Masoud Gharehbaghi, Masahiro Fujita: A new approach for selecting inputs of logic functions during debug, *International Symposium on Quality Electronic Design (ISQED 2017)* March 2017.

[12] Robert K. Brayton, Alan Mishchenko: ABC: An Academic Industrial-Strength Verification Tool, *22nd International Conference on Computer Aided Verification (CAV 2010)*, pp.24–40, 2010.