International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.02, Issue.11,
December-2014,
Pages:1194-1202

www.ijvdcs.org

Asynchronous NULL Convention Logic Circuits Gate Mapping Automation by using S-Box

BANAVATHU VEERASWAMI NAYAK[1], B. RATNA RAJU[2], NANDIPATI GANNANAGA PRASAD[3]

[1]PG Scholar, Dept of ECE, Kakinada Institute of Engineering and Technology, Kakinada, AP, India.
[2]Associate Professor, Dept of ECE, Kakinada Institute of Engineering and Technology, Kakinada, AP, India.
[3]Associate Professor, Dept of ECE, Kakinada Institute of Engineering and Technology, Kakinada, AP, India.

Abstract: In every VLSI design we mainly go for the reducing the size of the chip by taking consideration of area, power and cost. In this project we go for the reducing the area .Design automation techniques are a key challenge in the widespread application of timing-robust asynchronous circuit styles. A new methodology for mapping multi rail logic expressions to NULL convention logic (NCL) gate library is proposed by using s-box. The new methodology is then compared to another recently proposed mapping approach, demonstrating that the new methodology can further reduce the area and improve the delay of NCL circuits. Also, in contrast to the original approach, only targets area reduction. In order to automate the new methodology and compare it with the original one, both methodologies were implemented in the Verilog programming language and compared in terms of mapping performance and runtime. The results show that, depending on the test circuit.

Keywords: Automation, Factoring, Gate Mapping, Grouping, NULL Convention Logic (NCL), Technology Mapping, S-Box.

I. INTRODUCTION

Design automation techniques are a key challenge in the widespread application of timing-robust asynchronous circuit styles. Asynchronous circuits keep the assumption that signals are binary, but remove the assumption that time is discrete. This has several possible benefits are Clock skew is the difference in arrival times of the clock signal at different parts of the circuit. Since asynchronous circuits by definition have no globally distributed clock, there is no need to worry about clock skew. In contrast, synchronous systems often slow down their circuits to accommodate the skew. As feature sizes decrease, clock skew becomes a much greater concern Standard synchronous circuits have to toggle clock lines, and possibly precharge and discharge signals, in portions of a circuit unused in the current computation. For example, even though a looting point unit on a processor might not be used in a given instruction stream, the unit still must be operated by the clock. Although asynchronous circuits often require more transitions on the computation path than synchronous circuits, they generally have transitions only in areas involved in the current computation.

Note that there are techniques being used in synchronous designs to address this issue as well. Synchronous circuits must wait until all possible computations have completed before latching the results, yielding worst-case performance. Many Asynchronous systems sense when a computation has completed, allowing them to exhibit average-case performance. For circuits such as ripple-carry adders where the worst-case delay is significantly worse than the average-case delay, this can result in a substantial savings In systems such as a synchronous microprocessor, the system clock, and thus system performance, is dictated by the slowest (*critical*) path. Thus, most portions of a circuit must be carefully optimized to achieve the highest clock rate, including rarely used portions of the system. In logic, the converse of a categorical or implicational statement is the result of reversing its two parts. For the implication $P \rightarrow Q$, the converse is $Q \rightarrow P$. For the categorical proposition *All S is P*, the converse is *All P is S*. In neither case does the converse necessarily follow from the original statement. The categorical converse of a statement is contrasted with the contra positive and the obverse. *S* is a statement of the form *P implies Q* ($P \rightarrow Q$), then the converse of *S* is the statement *Q implies P* ($Q \rightarrow P$).

In general, the verity of *S* says nothing about the verity of its converse, unless the antecedent *P* and the consequent *Q* are logically equivalent. For example, consider the true statement "If I am a human, then I am mortal." The converse of that statement is "If I am mortal, then I am a human," which is not necessarily true. On the other hand, the converse of a statement with mutually inclusive terms remains true, given the truth of the original proposition.. NCL circuits are correct-by construction designs, requiring very little timing analysis, if any. In today's nanometer processes where meeting timing closure is becoming more and more difficult due to increasing clock rates and decreasing feature sizes, this quality can be very attractive. NCL is also one of the few asynchronous design paradigms that have been used for a number of industrial designs. The

main obstacle preventing the widespread application of asynchronous design paradigms, such as NCL, in industry is the lack of standard computer-aided design (CAD) tools that support automating the design process. Several NCL design automation flows have been proposed in the literature so far.

II. LITERATURE SURVEY

Technology Mapping and Cell Merger for Asynchronous Threshold Networks Done by Cheoljoo Jeong and Steven M. Nowick. A key challenge in using timing-robust asynchronous circuit styles is the lack of automated optimization techniques. In this paper, technology mapping and cell merger algorithms for asynchronous threshold networks are introduced. The cell merger problem is a restricted form of technology mapping where only adjacent cells are merged. The two algorithms can each target either delay or area, or a combined delay–area cost function. Experiments were performed on substantial industrial design examples (DES and GCD circuits) that had already been optimized by an existing commercial asynchronous synthesis tool flow. Average delay improvements of 31.6% for basic technology mapping and 29.6% for basic cell merger were obtained. Average area improvements of 9.5% for basic technology mapping and 8.5% for basic cell merger were also obtained. Additional experiments were performed on the largest MCNC combinational benchmarks with similar results. Finally, targeting a hybrid cost function, area minimization under specified hard timing constraints, a further area reduction of up to 10.7% on average in technology mapping was recovered without compromising the overall system performance.

The new algorithms are the first systematic and general mapping approach for asynchronous threshold networks, targeting delay or area, which preserve the timing-robustness properties of the initial unoptimized circuits. Design of asynchronous circuits by Synchronous CAD tools done by Alex Kondratyev and Kelvin Lwin. The roadblock to wide acceptance of asynchronous methodology is poor CAD support. Current asynchronous design tools require a significant re-education of designers, and their features are far behind synchronous commercial tools. This paper considers a particular subclass of asynchronous circuits (Null Convention Logic or NCL) and suggests a design flow that is based entirely on commercial CAD tools. This new design flow shows a significant area improvement over known flows based on NCL. Optimization of NULL convention self-timed circuits done by S.C. Smitha, R.F. These results the asynchronous circuits reducing the number of MOS and wire area. We provide two synthesis methods and simulation results of the gates and full-adder. The evaluation results of area dissipation and average delay show the advantages of the proposed circuitry.

III. EXISTING METHOD

Asynchronous logic has drawn more attention. One of the major reasons is the clock management problem for the increasingly complex synchronous circuits. Delay insensitive asynchronous logic uses handshaking protocols rather than clocks to control the circuit behavior. Theoretically, NULL Convention Logic (NCL), a quasi-delay-insensitive logic, is a symbolically complete logic, which expresses process completely in terms of the logic itself. Logic signals in NCL circuits are usually encoded in a multiple-rail format. The simplest encoding scheme is dual-rail logic, which uses two wires to interpret one signal value. There are two valid states DATA state, which including DATA 0 and DATA 1, and NULL, represented by both wires being logic low. The NCL logic family is composed of threshold gates. The main obstacle preventing the widespread application of asynchronous design paradigms, such as NCL, in industry is the lack of standard computer-aided design (CAD) tools that support automating the design process. Several NCL design automation flows have been proposed in the literature so far. Technology mapping, in particular, is the heart of such design flows since the performance of the mapped circuits is directly determined by the efficiency of such mapping.

IV. PROPOSED SYSTEM

Design automation techniques are a key challenge in the widespread application of timing-robust asynchronous circuit styles. A new methodology for mapping multi rail logic expressions to NULL convention logic (NCL) gate library is proposed. The new methodology is then compared to another recently proposed mapping approach, demonstrating that the new methodology can further reduce the area and improve the delay of NCL circuits. Also, in contrast to the original approach, only targets area reduction. The results show that, depending on the test circuit. After several synthesis and optimization steps each combinational module in the initial synchronous RTL design is finally expressed as dual-rail SOP expressions describing its outputs in terms of its inputs. Then, each SOP expression is separately mapped to NCL gates using the grouping algorithm.

The algorithm starts with a multi-rail SOP expression as input and then groups the product terms of the SOP expression such that an area-efficient implementation is obtained. Taking $F1 = A0B1C1 + A0B1D1 + A1B1 + B1C0 + B1D0 + A1C1$ as an example, the first step in the original grouping algorithm is to remove all four-variable terms from the initial SOP expression, because according to the NCL gate library, these terms cannot be grouped with any other terms to make larger groups and are always realized using TH44 gates. Thus, any four variable terms in the initial SOP expression can be individually regarded as a group and moved to the set of final groups. Since $F1$ has no four-variable term, no action is needed. Next, the distinct variables in the SOP expression are counted; if the number is less than 4, the previously grouped terms can be potentially combined with the SOP expression to make 4-feasible groups. A new mapping algorithm was proposed and compared with the original one for mapping multi-rail logic expressions to the NCL gate library. Both mapping

algorithms were implemented in the Perl programming language and tested on some multi-rail logic expressions and circuit components. The proposed mapping algorithm was shown to outperform the original one in terms of area and delay reduction. We showed that the new mapping algorithm could result in up to 10% area reduction and 39% delay reduction. Although the new mapping algorithm performs mapping in one pass requiring less computation in the main loop, its average runtime is higher compared to the original method because the new mapping algorithm incorporates several extra optimization steps. Also, in contrast to the original mapping algorithm, the new mapping algorithm can map a multi-rail expression to a restricted subset of NCL gates and can target any cost function.

A. Module's:

- Original Mapping Algorithm
- Proposed Mapping Algorithm

V. MODULE DESCRIPTION

A. Original Mapping Algorithm

After several synthesis and optimization steps each combinational module in the initial synchronous RTL design is finally expressed as dual-rail SOP expressions describing its outputs in terms of its inputs. Then, each SOP expression is separately mapped to NCL gates using the grouping algorithm. The algorithm starts with a multi-rail SOP expression as input and then groups the product terms of the SOP expression such that an area-efficient implementation is obtained. Taking $F1 = A0B1C1 + A0B1D1 + A1B1 + B1C0 + B1D0 + A1C1$ as an example, the first step in the original grouping algorithm is to remove all four-variable terms from the initial SOP expression, because according to the NCL gate library, these terms cannot be grouped with any other terms to make larger groups and are always realized using TH44 gates. Thus, any four-variable term in the initial SOP expression can be individually regarded as a group and moved to the set of final groups. Since $F1$ has no four-variable term, no action is needed. Next, the distinct variables in the SOP expression are counted; if the number is less than 4, the previously grouped terms can be potentially combined with the SOP expression to make 4-feasible groups.

B. Proposed Mapping Algorithm

The standard NCL gate library is comprised of 27 gates, each of which has several properties. Some of these properties are technology-independent, such as the set function and the number of transistors, while some are technology-dependent, such as area and delay. The main idea of the new grouping algorithm is to sort the list of NCL gates based on a cost function prior to grouping. At the time of grouping, the gates occupying a higher position in the sorted list are considered more important and will have a higher priority in grouping. Moreover, in contrast to the original grouping method, the list of NCL gates is no longer required to include all 27 gates as long as it contains enough gates to cover any given SOP expression. This usually means that the gate list must contain at least all of the NCL AND (THnn) and OR (TH1n) gates. The first step in using the proposed grouping algorithm is to select a set of NCL gates to which the SOP expressions are to be mapped. The selected NCL gates are then sorted on the basis of a cost function, and each gate is assigned a priority number accordingly. At the time of grouping, this priority number is used to determine which groups are most desirable. The cost function is usually area or delay. Comparison of the Original and Proposed Grouping Methods:

TABLE I: Comparison of the Original and Grouping Methods

	Resulted Groups	# T	Area [μm^2]	Delay [ns]	Runtime [ms]
1	$F = A^0B^1 + A^0C^0 + A^0D^1 + B^1C^0 + A^0D^0 + B^0C^1D^1$				
Original	$X = A^0B^1 + A^0C^0 + A^0D^1 + B^1C^0$ (TH44w322) $Y = B^0C^1D^1$ (TH33) $F = A^0D^0 + X + Y$ (TH24w22)	52	271	660	13.5
Proposed	$X = A^0B^1 + A^0C^0 + A^0D^0 + B^1C^0$ (TH44w322) $Y = A^0D^1 + B^0C^1D^1$ (TH54w32) $F = X + Y$ (TH12)	46	254	434	13.8
2	$F = B^1C^0D^0 + C^1D^0 + A^0B^1 + A^0C^1 + A^0D^1 + B^0C^1$				
Original	$X = A^0B^1 + A^0C^1 + C^1D^0$ (THand0) $Y = A^0D^1 + B^0C^1$ (THxor0) $Z = B^1C^0D^0 + X$ (TH34w3) $F = Y + Z$ (TH12)	63	350	734	13.9
Proposed	$X = A^0B^1 + A^0C^1 + A^0D^1$ (TH44w3) $Y = B^1C^0D^0 + C^1D^0$(TH54w32) $Z = B^0C^1 + X + Y$ (TH24w22)	52	293	704	20.7
3	$F = A^0B^1C^0 + A^0B^0C^1 + A^1B^1C^0 + A^1B^0C^1 + B^1C^0D^1$				
Original	$X = A^0B^1C^0 + A^1B^1C^0$ (TH54w22) $Y = A^0B^0C^1 + A^1B^0C^1$ (TH54w22) $Z = B^1C^0D^1 + X$ (TH34w3) $F = Y + Z$ (TH12)	60	368	739	5.6
Proposed	$X = A^0B^1C^0 + A^1B^1C^0$ (TH54w22) $Y = A^0B^0C^1 + A^1B^0C^1$ (TH54w22) $Z = B^1C^0D^1$ (TH33) $F = X + Y + Z$ (TH13)	60	352	447	11.4

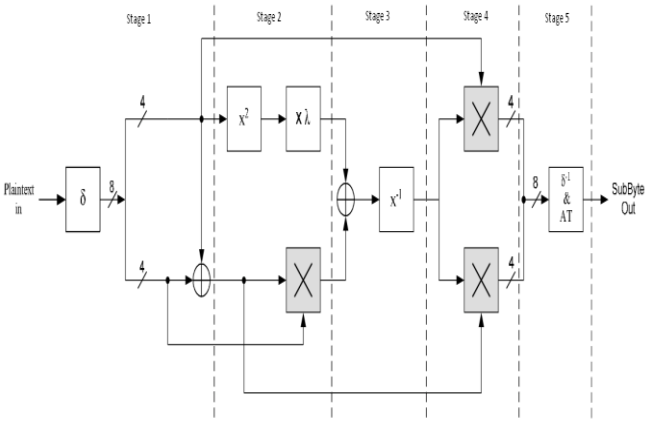

Fig1. S-Box or Subbyte.

C. Sub Bytes

The Sub Bytes operation is a nonlinear byte substitution. Each byte from the input state is replaced by another byte according to the substitution box (called the S-box). The S-box is computed based on a multiplicative inverse in the finite field GF (2^8) and a bitwise affine transformation (fig 1).

Fig2. Internal Blocks

1. Addition In Gf(2^4)

Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation.

D. GF(2^4) Multiplier

Sub Bytes is a nonlinear transformation that uses 16 byte substitution tables (S-Boxes). An S-Box is the multiplicative inverse of a Galois field GF(2^4) followed by an affine transformation**.** Although two Galois Fields of the same order are isomorphic, the complexity of the field operations may heavily depend on the representations of the field elements. Composite field arithmetic can be employed to reduce the hardware complexity.

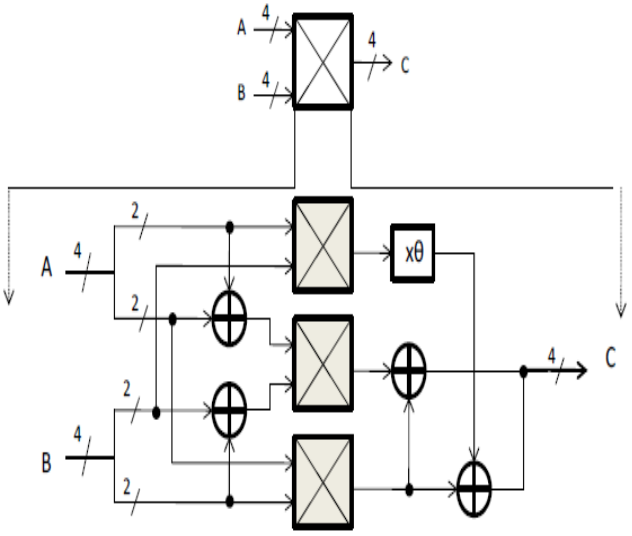

Fig3. GF(2^4) Multiplier.

E. GF (2^2) Multiplier

While each finite field is itself not infinite, there are infinitely many different finite fields; their number of elements (which is also called cardinality) is necessarily of the form p^n where p is a prime number and n is a positive integer (fig 4 to 7).

F. GF(2^4) Squarer

It consists of bitwise xor operation. A bitwise operation operates on one or more bit atterns or binary numerals at the level of their individual bits.

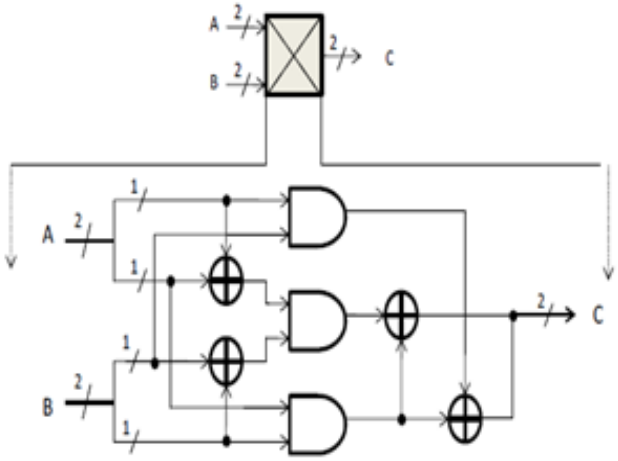

Fig4. Implementation Of The Gf(2^2) Multiplier.

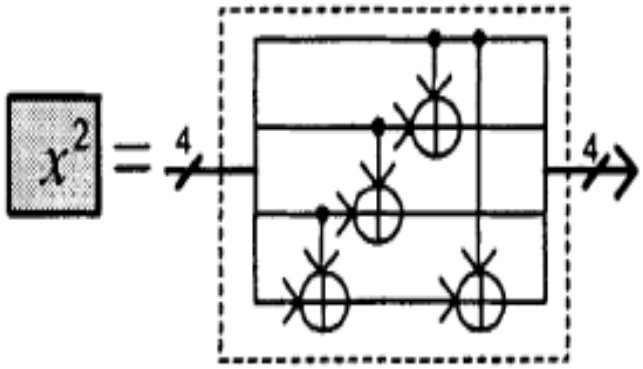

Fig5. Hardware Diagram For Squarer in GF(2^4) Constant Multiplier (Xϕ).

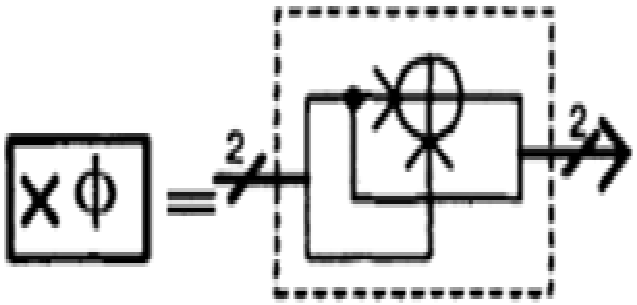

Fig6. Hardware Implementation of Multiplication with Constant Φ Constant Multiplier (L).

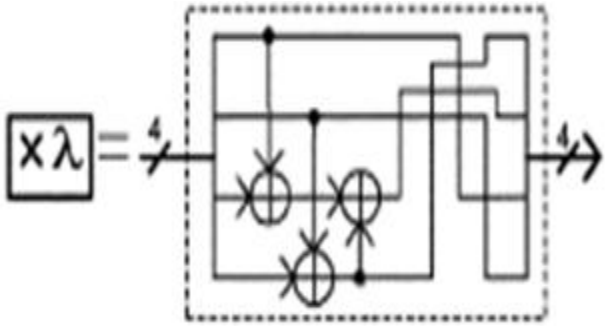

Fig7. Hardware Diagram for Multiplication with Constant.

G. Affine Transform

TABLE II: Boolean Equations for Affine Transformation and Inverse Affine Transformation Components

$q = aff\_trans(i)$	$q = aff\_trans^{-1}(i)$
$q_0 = (i_0 \oplus i_4) \oplus (i_5 \oplus i_6) \oplus (i_7 \oplus 1)$	$q_0 = i_2 \oplus i_5 \oplus i_7 \oplus 1$
$q_1 = i_1 \oplus i_5 \oplus i_6 \oplus i_7 \oplus i_0 \oplus 1$	$q_1 = i_0 \oplus i_3 \oplus i_6$
$q_2 = i_2 \oplus i_6 \oplus i_7 \oplus i_0 \oplus i_1$	$q_2 = i_1 \oplus i_4 \oplus i_7 \oplus 1$
$q_3 = i_3 \oplus i_7 \oplus i_0 \oplus i_1 \oplus i_2$	$q_3 = i_2 \oplus i_5 \oplus i_0$
$q_4 = i_4 \oplus i_0 \oplus i_1 \oplus i_2 \oplus i_3$	$q_4 = i_1 \oplus i_3 \oplus i_6$
$q_5 = i_1 \oplus i_5 \oplus i_2 \oplus i_3 \oplus i_4 \oplus 1$	$q_5 = i_2 \oplus i_4 \oplus i_7$
$q_6 = i_6 \oplus i_2 \oplus i_3 \oplus i_4 \oplus i_5 \oplus 1$	$q_6 = i_0 \oplus i_3 \oplus i_5 \oplus 1$
$q_7 = i_7 \oplus i_3 \oplus i_4 \oplus i_5 \oplus i_6$	$q_7 = i_1 \oplus i_4 \oplus i_6$

H. Standard Ncl Gates

TABLE III: LIBRARY OF THE 27 STANDARD NCL GATES

Threshold Gate	Set Function	No. of Transistor
TH12	$A + B$	6
TH22	AB	12
TH13	$A + B + C$	8
TH23	$AB + AC + BC$	18
TH33	ABC	16
TH23w2	$A + BC$	14
TH33w2	$AB + AC$	14
TH14	$A + B + C + D$	10
TH24	$AB + AC + AD + BC + BD + CD$	26
TH34	$ABC + ABD + ACD + BCD$	24
TH44	$ABCD$	20
TH24w2	$A + BC + BD + CD$	20
TH34w2	$AB + AC + AD + BCD$	22
TH44w2	$ABC + ABD + ACD$	23
TH34w3	$A + BCD$	18
TH44w3	$AB + AC + AD$	16
TH24w22	$A + B + CD$	16
TH34w22	$AB + AC + AD + BC + BD$	22
TH44w22	$AB + ACD + BCD$	22
TH54w22	$ABC + ABD$	18
TH34w32	$A + BC + BD$	17
TH54w32	$AB + ACD$	20
TH44w322	$AB + AC + AD + BC$	20
TH54w322	$AB + AC + BCD$	21
THXOR0	$AB + CD$	20
THAND0	$AB + BC + AD$	19
TH24comp	$AC + BC + AD + BD$	18

For the rest of this paper, for the sake of comparison with the original grouping method, let us assume that the cost function is area and that all of the NCL gates are available for grouping. A typical priority table is shown in Table VII, in which the gates are sorted based on how many three-variable and two-variable terms they cover. If some gates cover the same number of terms, the one that covers more variables i assigned a higher priority; if they also cover the same number of variables, then they are assigned the same priority number.Note that the NCL OR gates (TH1n) are missing from this table because they are used in the last step of the proposed grouping algorithm when the final groups are ORed. This step will be discussed later in more detail.The proposed grouping algorithm shows In summary, the algorithm begins with moving all the four-variable terms to the set of final groups, just as in the original grouping method. Next, all k-feasible groups are found by using a k-feasibility table and combining all parent terms with the other terms, as discussed in the original grouping method. The k-feasible groups are then arranged into priority levels according to Table VII. Next, starting from the highest non-empty priority level, the non-redundant groups in each priority level are found, their terms are removed from all the other groups, and the priority levels are updated accordingly. This procedure continues until all priority levels become empty.Let us take the same SOP expression that was used to explain the original grouping algorithm as an example (fig 8).

Assume F1 = A0B1C1 + A0B1D1 + A1B1 + B1C0 + B1D0 + A1C1.

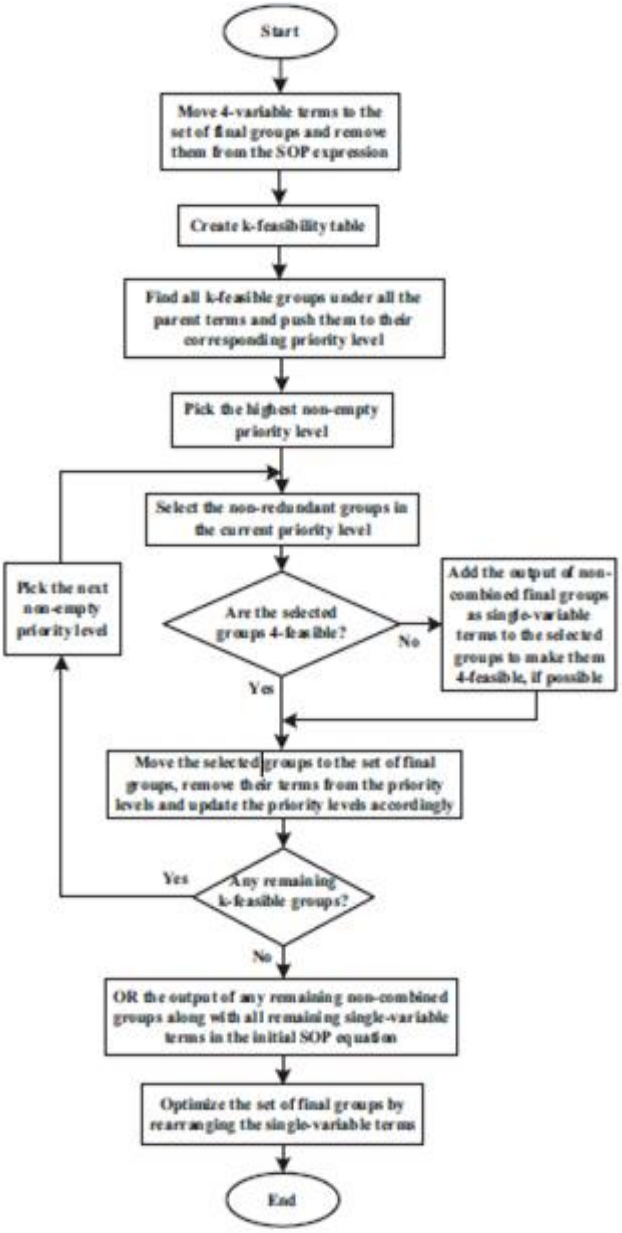

Fig8. Proposed grouping algorithm.

VI. RESULTS

In order to compare the original and the proposed grouping methods, both methods were implemented using the Perl programming language. The developed Perl scripts were then tested on several SOP expressions and circuit components. Table XII shows the results of running the developed scripts on some of the examples used in this paper. For each example, the groups resulting from both grouping methods are shown. Moreover, the corresponding number of transistors (T), area, delay, and runtime are also compared. Both grouping scripts were run on an Intel Core 2 Duo 2.4-GHz machine with 4 GB RAM, and the area and delay information came from an NCL physical cell library realized in a 1.8 V 0.18-µm TSMC CMOS process

A. Technological schematic

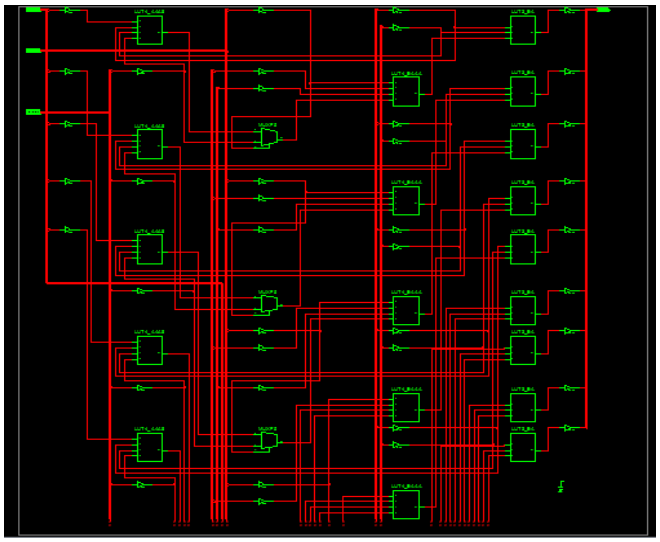

Fig9. Orignal Mapping.

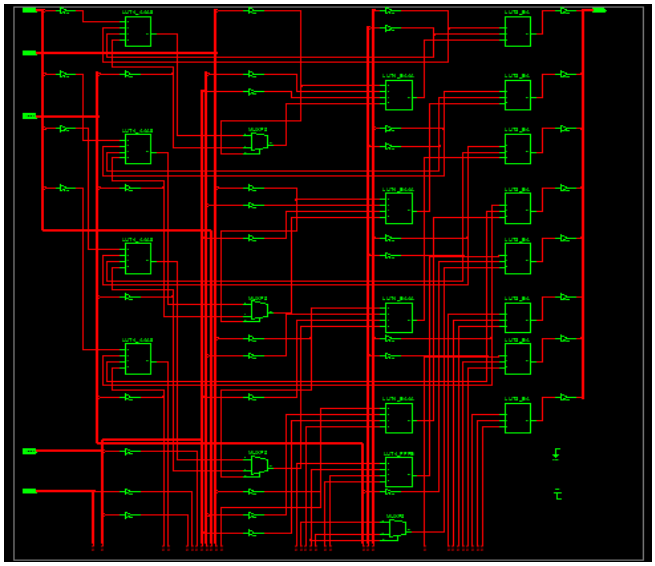

Fig10. Proposed Mapping.

By testing the circuit we acquire the result as follows Original ncl method 1

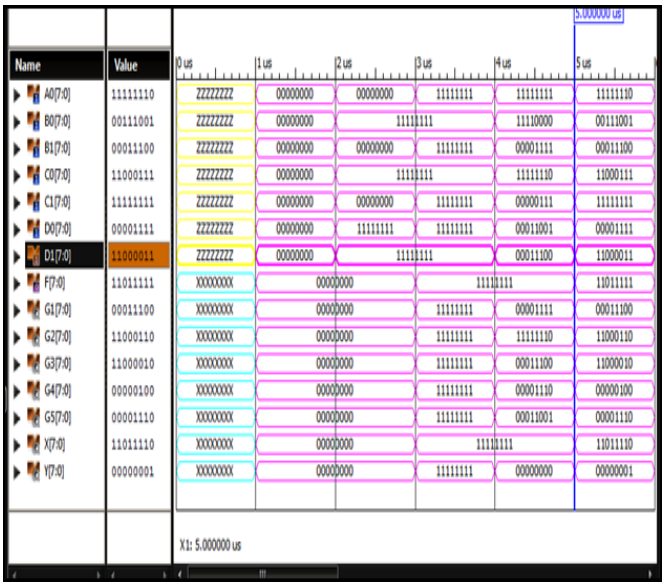

Fig11. Proposed Ncl Method1.

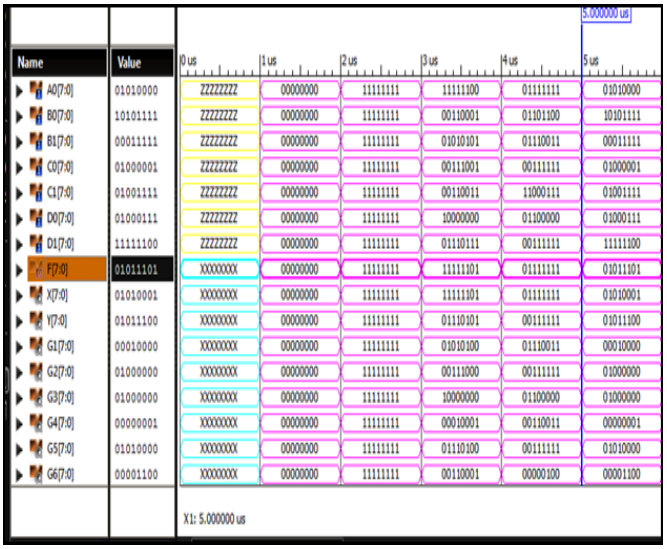

Fig12. Top module of s-box ncl.

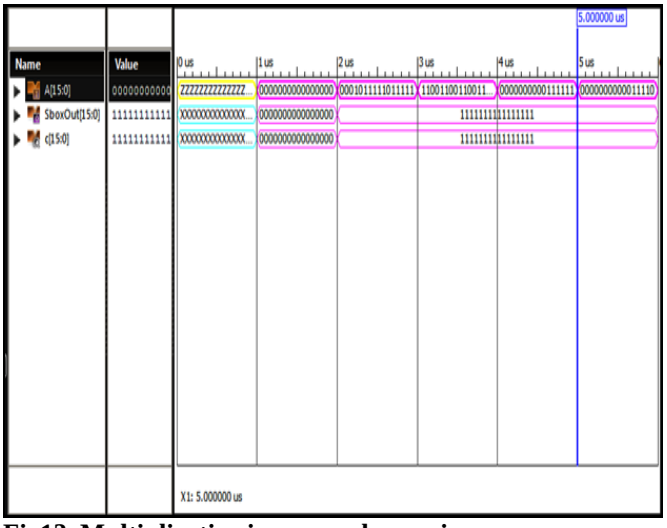

Fig13. Multiplicative inverse ncl mapping.

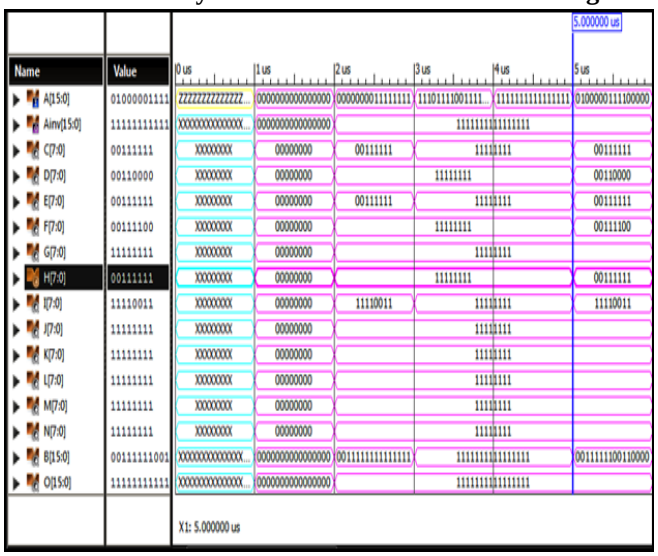

Fig14. Affine ncl mapping.

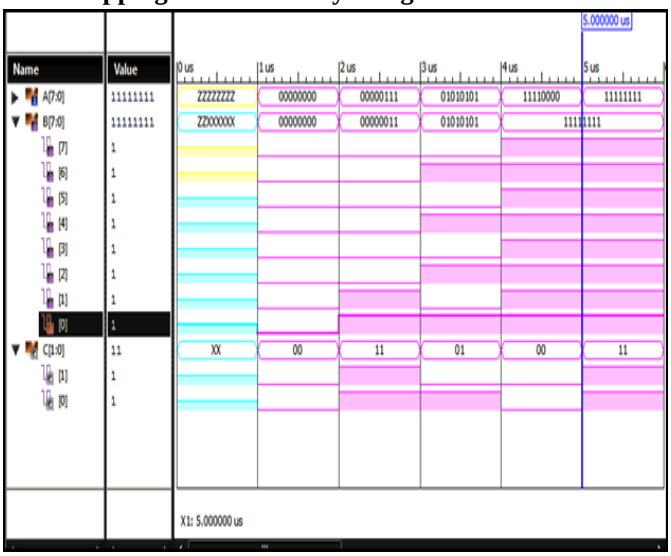

Fig17. 4 bit multiplication ncl.

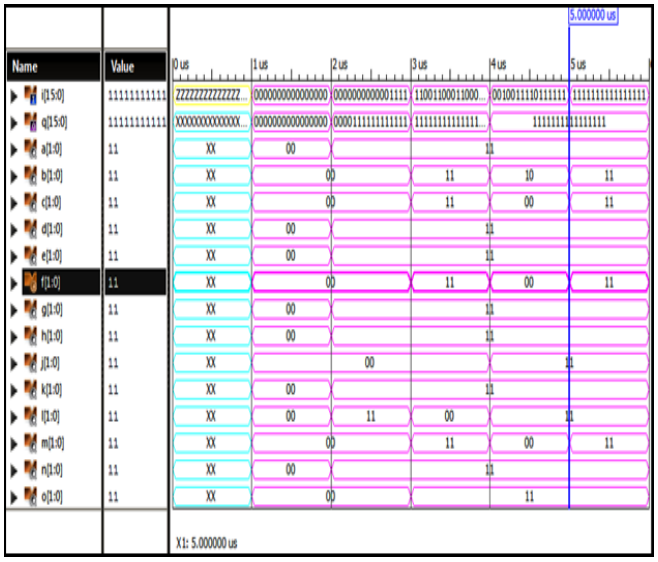

Fig15. Iso ncl mapping.

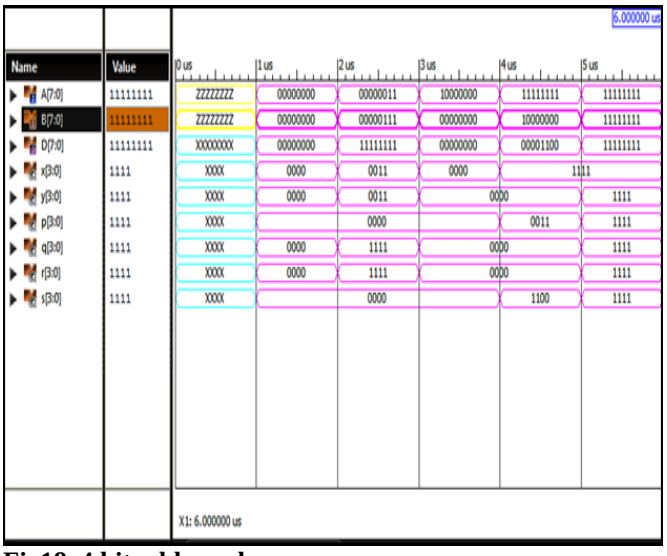

Fig18. 4 bit adder ncl.

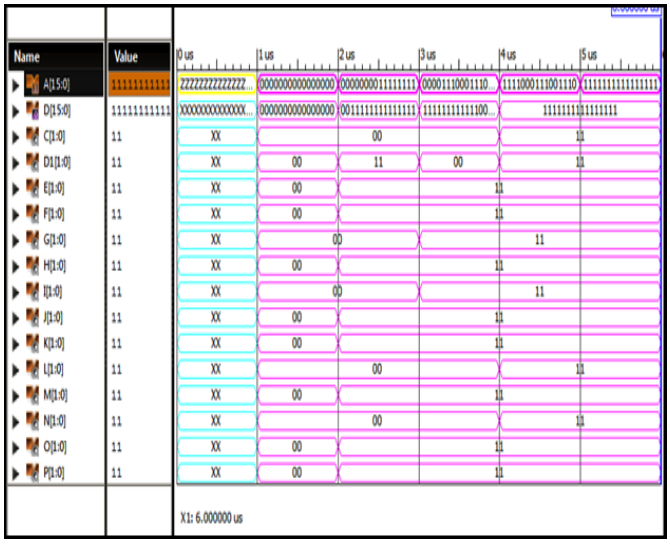

Fig16. Square block ncl.

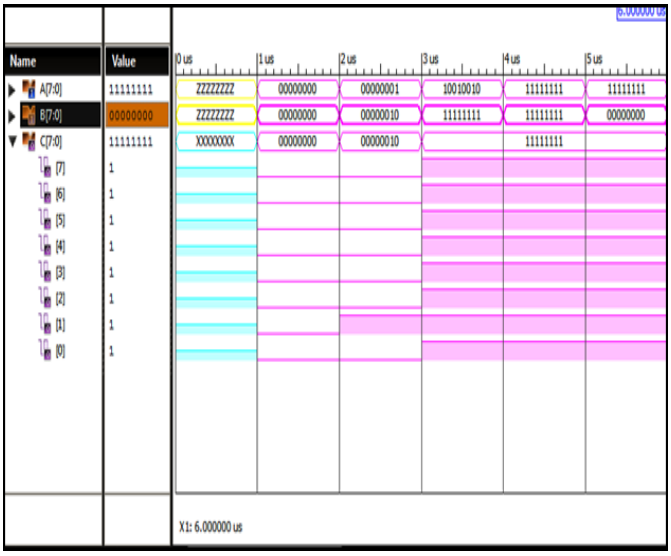

Fig19. Inverse iso ncl mapping.

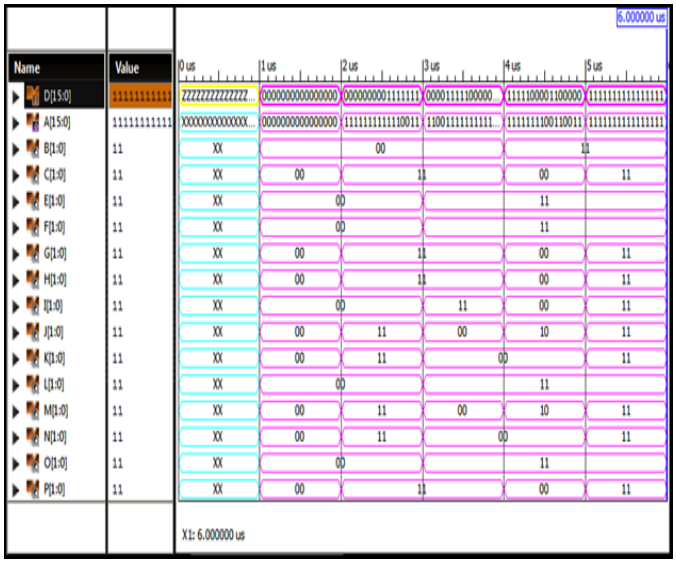

Fig20. Sxor ncl.

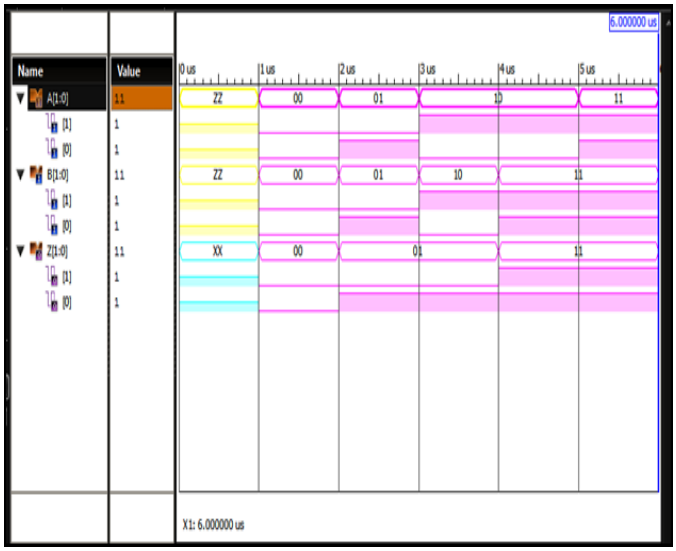

Fig21. Th24comp ncl.

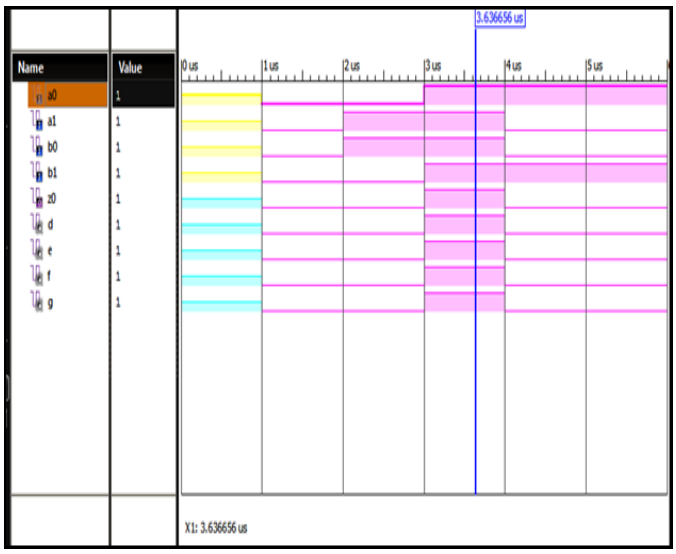

Fig22. Area utilization.

TABLE III: Original Mapping Device Utilization Summary

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	31	7,168	1%	
Logic Distribution				
Number of occupied Slices	18	3,584	1%	
Number of Slices containing only related logic	18	18	100%	
Number of Slices containing unrelated logic	0	18	0%	
Total Number of 4 input LUTs	36	7,168	1%	
Number used as logic	31			
Number used as a route-thru	5			
Number of bonded IOBs	72	141	51%	
Total equivalent gate count for design	213			
Additional JTAG gate count for IOBs	3,456			

TABLE IV: Proposed Mapping Device Utilization Summary

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	28	7,168	1%	
Logic Distribution				
Number of occupied Slices	16	3,584	1%	
Number of Slices containing only related logic	16	16	100%	
Number of Slices containing unrelated logic	0	16	0%	
Total Number of 4 input LUTs	32	7,168	1%	
Number used as logic	28			
Number used as a route-thru	4			
Number of bonded IOBs	64	141	45%	
Total equivalent gate count for design	192			
Additional JTAG gate count for IOBs	3,072			

VII. CONCLUTION

A new mapping algorithm was proposed and compared with the original one for mapping Multi-rail logic expressions to the NCL gate library. Both mapping algorithms were implemented in the Verilog programming language and tested on some multi-rail logic expressions and circuit components. The proposed mapping algorithm was shown to outperform the original one in terms of area reduction. Although the new mapping algorithm performs mapping in one pass requiring less computation in the main loop, its average runtime is higher compared to the original method because the new mapping algorithm incorporates several extra optimization steps. Also, in contrast to the original mapping algorithm, the new mapping algorithm can map a multi-rail expression to a restricted subset of NCL gates and can target any cost function.

VIII. REFERENCES

[1] K. M. Fant and S. A. Brandt, "NULL convention logic: A complete and consistent logic for asynchronous digital circuit synthesis," in Proc. Int. Conf. Appl. Specific Syst., Archit. Process., Aug. 1996, pp. 261–273.

[2] K. M. Fant, Logically Determined Design: Clockless System Design with NULL Convention Logic. New York: Wiley, 2005.

[3] S. C. Smith and J. Di, "Designing asynchronous circuits using NULL convention logic (NCL)," in Synthesis Lectures on Digital Circuits and Systems. San Mateo, CA: Morgan & Claypool, 2009.

[4] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," in Proc. 6th Int. Symp. Adv. Res. Asynchron. Circuits Syst., Apr. 2000, pp. 114–125.

[5] C. Jeong and S. M. Nowick, "Technology mapping and cell merger for asynchronous threshold networks," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 27, no. 4, pp. 659–672, Apr. 2008.

[6] B. Bhaskaran, "Automated synthesis and cycle reduction optimization for asynchronous NULL convention circuits using industry-standard CAD tools," Ph.D. dissertation, Dept. Comp. Eng., Univ. Missouri - Rolla, Rolla, 2007.

[7] R. Reese, S. C. Smith, and M. A. Thornton, "Uncle—an RTL approach to asynchronous design," in Proc. 18th IEEE Int. Symp. Adv. Res. Asynchron. Circuits Syst., May 2012, pp. 65 72.

[8] J. McCardle and D. Chester, "Measuring an asynchronous processor's power and noise," in Proc. Synopsys Users Group Conf., 2001, pp. 66–70.

[9] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL convention self-timed circuits," Integr. VLSI J., vol. 37, no. 3, pp. 135–165, Aug. 2004.

[10] T. Verhoeff, "Delay-insensitive codes—an overview," Distrib. Comput., vol. 3, no. 1, pp. 1 8, 1988.