Prioritized Prime Implicant Patterns Puzzle for Novel Logic Synthesis and Optimization

Kuo-Hsing Cheng* and Shun-Wen Cheng
Department of Electrical Engineering, Tamkang University, TAIWAN
*E-mail: cheng@ee.tku.edu.tw**

Abstract

Compare CMOS Logic with Pass-Transistor Logic, a question was raised in our mind: "Does any rule exist that contains all good?" This paper reveals novel logic synthesis and optimization procedures for full swing arbitrary logic function. The novel procedures are called Prioritized Prime Implicant Patterns Puzzle (PPIPP). Following the proposed procedures, we can get a new hybrid high performance logic circuit family, which has low power consumption, low power-delay product, area efficiency and suitable for low supply voltage. It has full swing signal in all nodes and high robustness against transistor downsizing and voltage scaling.

Index Term – Low power design, full-swing logic, hybrid logic, prime implicant, VLSI design.

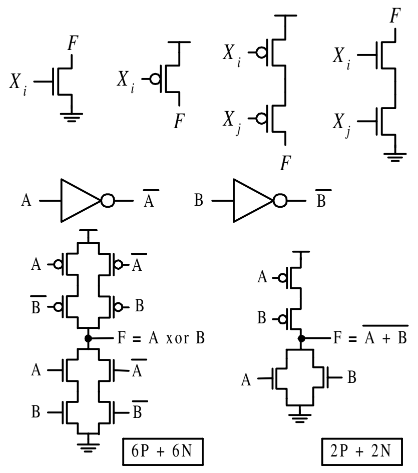
1. Introduction

On logic circuit design level, a proper choice of a circuit style for implementing combinational logic is an important issue. For example, in the NOR gate implementation, as shown in Fig. 1, the static CMOS logic circuit structure seems the better logic circuit family than the DVL [5], DPL [7] or any other logic circuit families.

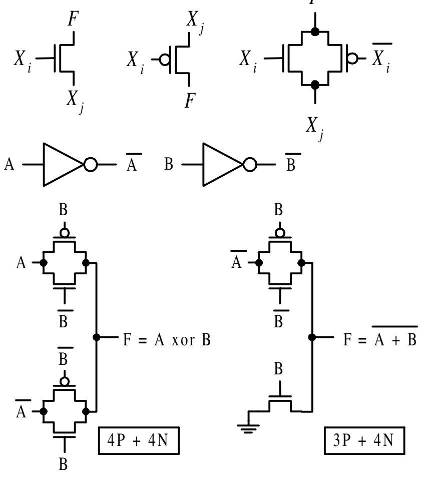
But when it comes to 2-input XOR logic implementation, as shown in Fig. 1, the static CMOS logic circuit family becomes the worst choice. This result may confuse someone in logic circuit family selecting.

In general, the static CMOS logic circuit structure can be seem as a special case of pass transistor logic network that the pass variables input signals are just "1" and "0", and the input signals xi and $\overline{X_i}$ are connected to drive the gate of the MOS transistor as shown in Fig. 1(a).

And shown in Fig. 1(b), the input signals xi and $\overline{X_i}$ can be used as the control variables or pass variables of the pass-transistor network. The control variables are connected to drive the gate of the MOS transistors. The pass variables are connected to the sources/drains of the MOS transistors.



(a). CMOS Logic Structure Style.



(b) Full Swing Pass-Transistor Logic Style.

Fig. 1. Compare CMOS with PTL, a question was raised in our mind: "Does any rule exist that contains all good?"

This paper is organized as follows. In Section 2, we show the fundamental circling concepts, as the background of the proposed method. Then in Section 3, the proposed method is shown and demonstrated by examples. In Section 4, process some comparison. Finally conclude the major findings and outline the future work.

2. Basic Circling Concepts

The ideas are based upon the pass transistor logic circuit implementation. As shown in Fig. 3, we use 2-input XOR function as a circuit implementation example. The detail design flow of the circuit will be shown in the following. In order to describe the basic raw circling procedures clearly, some basic notations and circuit implementation procedures [3] -- *Square*, *Modified K-map*, *Loop Circling*, *Selected Set*, *Implicate Loop* and *Circuit Implementation Methods* are shown as the following:

2.1. Square

The *Karnaugh map* (*K-map*) of a function specifies the value of the function for every combination of values of the independent variables. The *Square* indicates a function output state on the K-map. As shown in Fig. 2(a), the output state of the Square for {A=B=0}, plotted in the upper left on the K-map, is "1".

2.2. Modified Karnaugh map (K-map)

The *Modified K-map* is almost the same with the *K-map*, except that not only the power lines ("1"and "0") but also the input variables ("Xi" and "$\overline{X_i}$") are listed in the Square to represent the function result as shown in Fig. 2(b). It is straightforward to implement the circuit based on the Static CMOS Logic according to the *K-map*. And the *Modified K-map* provides us the thoughts of implementation of the new logic synthesis and optimization procedures.

2.3. Loop Circling

The *Loop Circling* is the method to implement the PTL circuit. A loop contains one or more squares on the *Modified K-map*. For example, the Square {A=B=0} and Square {A=1, B=0} combine to form the Loop (i) by looping the corresponding A's on the *Modified K-map* in Fig. 3(a). A loop may contains all squares that are never selected by other loops (Loop(iii) in Fig. 3(a)), or part squares are selected by other loop (*Square* {A=B=0} in Loop(ii) in Fig. 3(a) is selected by Loop(i)).

2.4. Selected Set

We define a set of controlling and passing variables that ever used for circuit implementation, call *Selected Set*. Which means we can choose the variables in the *Selected Set* for implementing new circuit without extra inverters to generate the newly complementary signals. The initial values in the *Selected Set* are {0, 1, Xi}. For example, the initial variable in the *Selected Set* is {0, 1, A, B} in Fig. 3(a). After every loop circling, we put the new selected passing and controlling variables in the Selected Set immediately. For example, the *Selected Set* is {0,1,A, B, $\overline{B}$ } after Loop (i) is circled. The select of variables in the *Selected Set* to implement new circuit is based on the choosing priority "0">"1">"Xi">"$\overline{X_i}$" (in the *Selected Set*) >"$\overline{X_i}$" (not in the *Selected Set*).

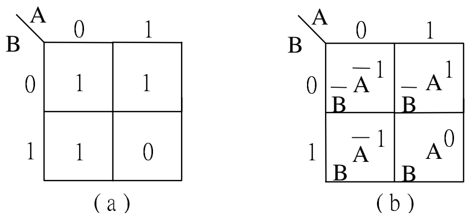


Fig. 2. (a) The K-map of the XOR Function
(b) The Modified K-map of the XOR Function.

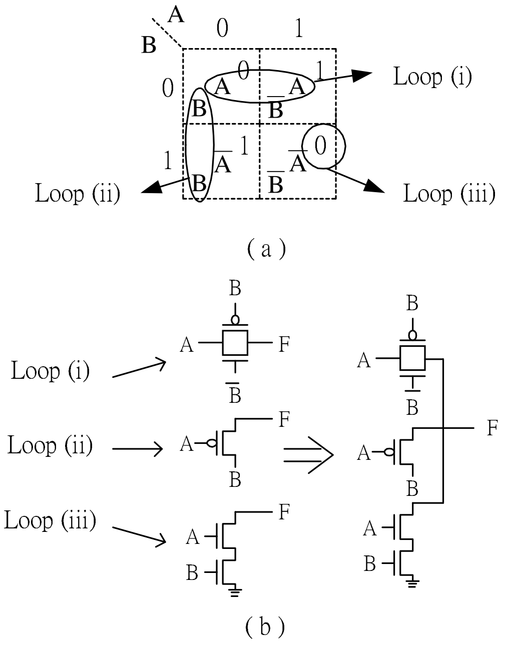


Fig. 3. (a), (b) The original circling procedures of the 2-input XOR Modified K-map.

2.5. Implicate Loop

An *Implicate Loop* may include partial or all squares that are already chosen by a selected Loop. For example as in Fig. 3(a), the Loop (ii) can be seen as a *Implicate Loop* to the Loop (i), cause the *Square* {A=0, B=0} is circled again. And due to the Loop (iii) (the *Square*{A=1, B=1}) is not circled by any other selected loops, so it is a *Non-Implicate Loop*.

2.6. Circuit Implementation Methods

For pass transistor circuit implementation, we will only concern those squares are newly choose in the current loop. If a loop contains newly outputs states has both 1's and 0's (Loop (i) in Fig. 3(b)), its pass-transistor circuit switch is implemented by both NMOS and PMOS (a transmission gate is used). The PMOS is used to implement all 1's loop (Loop (ii) in Fig. 3(b), the A=0, B=1 *Square* is only concerned) and the NMOS is used to implement all 0's loop (Loop (iii) in Fig. 3(b)).

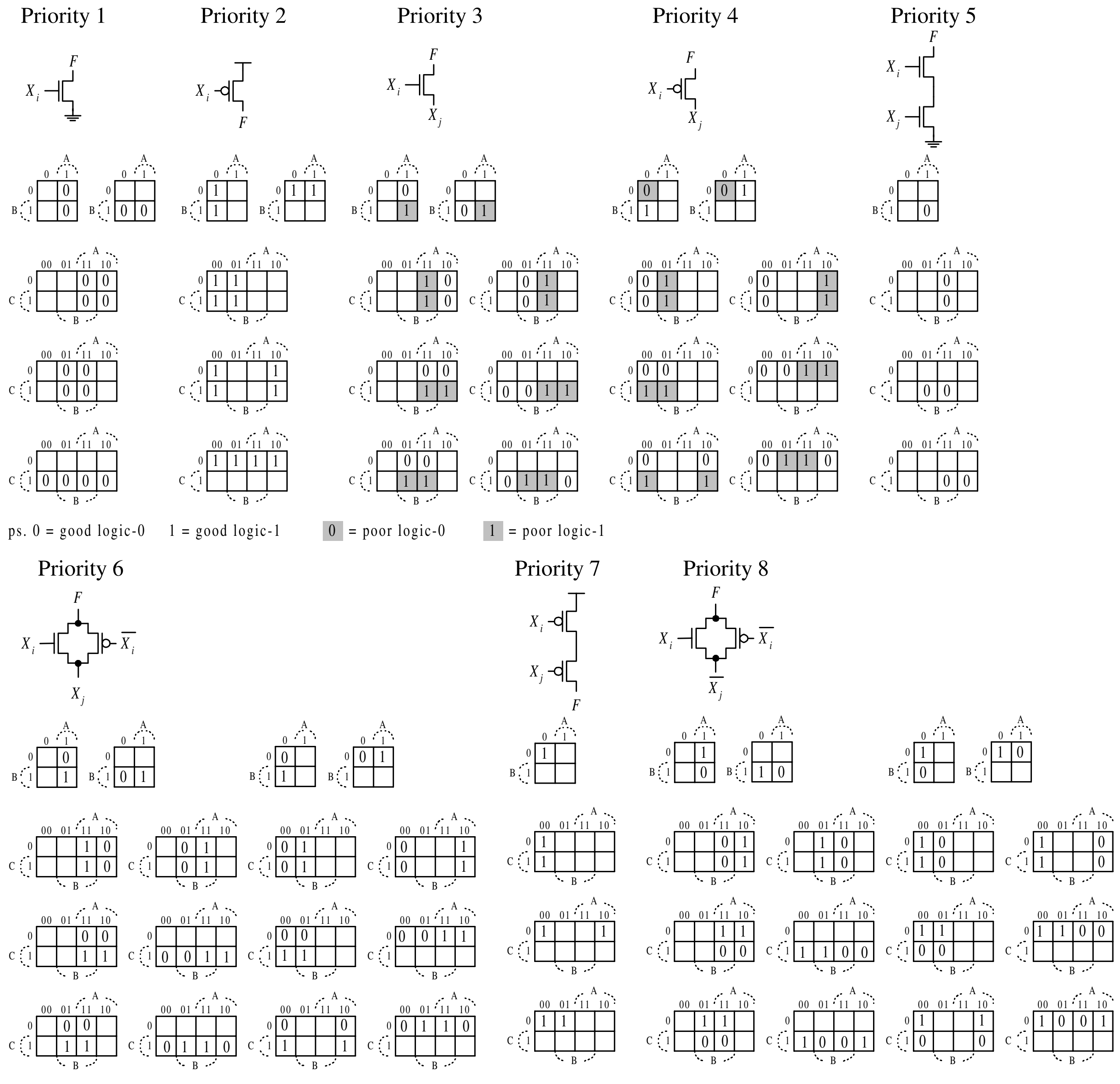


Fig. 4. The priorities of prime implicant patterns are constructed by electrical characteristics.

3. The Prioritized Prime Implicant Patterns Puzzle (PPIPP)

Thus, it is possible to develop a synthesis and optimization procedure of the pass transistor logic circuit for arbitrary logic function and high performance applications. Originally circling procedures are restricted by K-map, only works under four variables [3]. And the circles are difficulty to pin down. So the paper improve the previously work then proposed the Prioritized Prime Implicant Patterns Puzzle (PPIPP). It clearly handles the higher variables problem.

The priorities of prime implicant patterns are constructed by electrical characteristics, as shown in the previous section, ex. NMOS logic is better than PMOS, and the fewer input/control signals, the higher priority the prime implicant. In Fig. 4, it just briefly shows some template patterns. The priority order is priority 1>2>…>8. The proposed PPIPP arrange the prime implicant priority following the physical consideration, so it is superior to any other symbolic logic optimization and/or logic minimization methods.

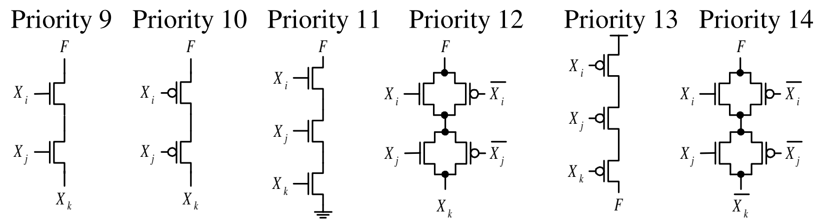
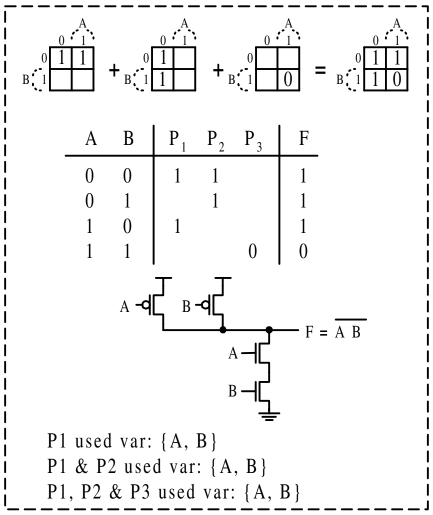


Fig. 5. The priority of rest other 3-input variable prime implicant patterns.

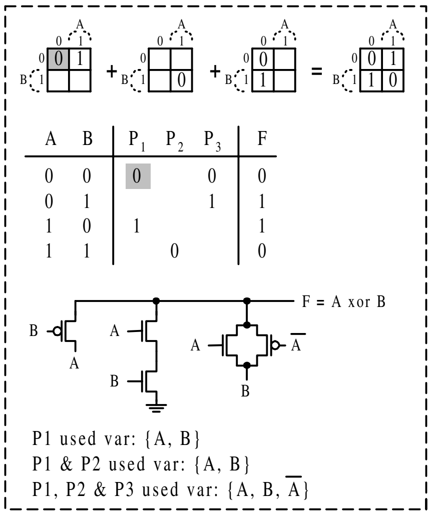
A	B	1		2		3		4		5	6				7	8			
0	0			1	1			0	0		0	0			1	1	1		
0	1		0	1			0	1			1			0		0			1
1	0	0			1	0			1			1	0				0	1	
1	1	0	0			1	1			0			1	1				0	0

Fig. 6. 2-input Variables Prime Implicant Patterns' Priority



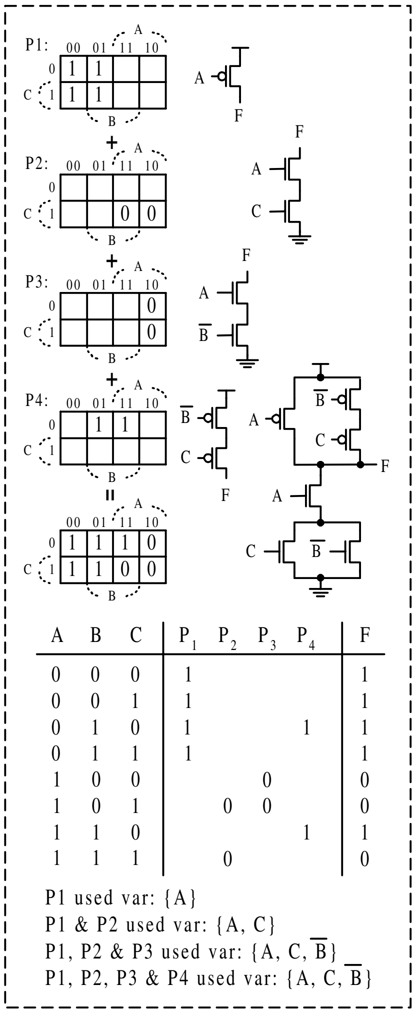
A	B	P_1	P_2	P_3	F
0	0	1	1		1
0	1		1		1
1	0	1			1
1	1			0	0

P1 used var: {A, B}
P1 & P2 used var: {A, B}
P1, P2 & P3 used var: {A, B}



A	B	P_1	P_2	P_3	F
0	0	0		0	0
0	1			1	1
1	0	1			1
1	1		0		0

P1 used var: {A, B}
P1 & P2 used var: {A, B}
P1, P2 & P3 used var: {A, B, $\overline{A}$}



A	B	C	P_1	P_2	P_3	P_4	F
0	0	0	1				1
0	0	1	1				1
0	1	0	1			1	1
0	1	1	1				1
1	0	0			0		0
1	0	1		0	0		0
1	1	0				1	1
1	1	1		0			0

P1 used var: {A}
P1 & P2 used var: {A, C}
P1, P2 & P3 used var: {A, C, $\overline{B}$}
P1, P2, P3 & P4 used var: {A, C, $\overline{B}$}

Fig. 7. 2-input NAND by PPIPP.

Fig. 8. 2-input XOR by PPIPP.

Fig. 9. Function F= $\overline{A}+B\overline{C}$ by PPIPP.

And notice the proposed PPIPP has no sneak path. It is either not like a binary decision diagram (BDD) tree. In Fig. 5, the priorities of rest other 3-input variable prime implicant patterns are shown. Notice the prime implicant patterns can be shown as in Fig. 6. If the proposed PPIPP needs to process n-input (n>4) signal function, the K-map is no more a limitation.

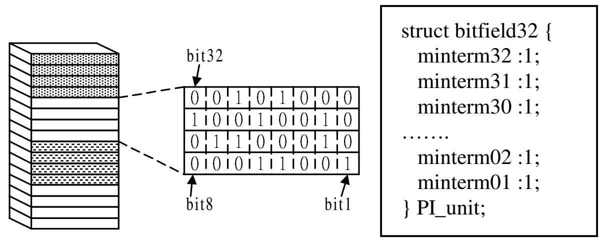
In Fig. 7, it processes 2-input NAND function then gets pure CMOS logic style circuit. And in Fig. 8, it processes 2-input XOR function then gets pass-transistor logic style circuit.

From Fig. 7, Fig. 8 and Fig. 9, it reveals the proposed PPIPP produce hybrid logic style circuit. It combines the advantage of CMOS logic and pass-transistor logic. It has full-swing signal in all nodes and high robustness against transistor downsizing and voltage scaling.

In the proposed PPIPP method, many prime implicant patterns may need some memory space. As shown in Fig. 10, using bit field structures reduce to one-eighth-memory space effectively.

4. Comparisons

Comparisons of the DVL, DPL, CMOS and new logic family through 2-input XOR logic functions are listed in Table.1. The comparisons are based on 0.35μm CMOS technology and post layout simulation for supply voltage at 1.5V. Possible transition combinations are simulated, and the time taken of the worst-case signal transition from input (50% level) to output (50% level) worst-case gate delay is applied as delay value. Power-delay product is calculated as a quality measure for power efficiency.



```
struct bitfield32 {
  minterm32 :1;
  minterm31 :1;
  minterm30 :1;
.......
  minterm02 :1;
  minterm01 :1;
} PI_unit;
```

Fig. 10. Using bit field structure to reduce the memory requirement effectively.

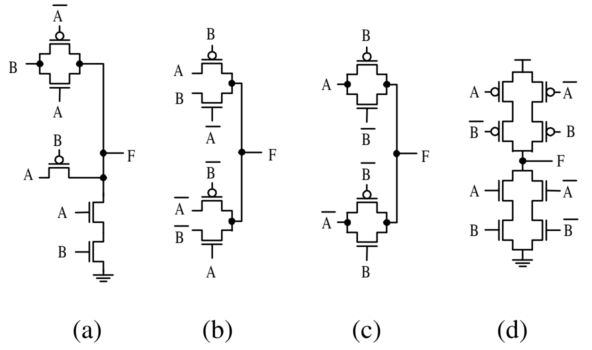


Fig. 11. Full swing 2-input XOR functions:
(a) The proposed logic style. (b) The DPL structure.
(c) The DVL structure. (b) The static CMOS structure.

	Delay-time (ns)	**Power (μW)**	**Normalize power-delay product**	**Global size of transistors.**
Fig. 11a	0.446	7.946	1.00	3P + 4N
Fig. 11b	0.366	11.35	1.17	4P + 4N
Fig. 11c	0.432	10.55	1.79	4P + 4N
Fig. 11d	0.643	15.05	2.72	6P + 6N

Table.1. Various logical circuits comparison results of the full swing 2-input XOR function.

Due to the pass transistor circuits using the passive MOS switches to implement a given logic function, in order to measure the average power dissipation of the original circuit, some inverters are added in front of the input of the original circuits. For a special 2-input XOR function in Fig.11, the new circuit shown in Fig. 11(a) and also proven in literature [8], has advantages over DVL, DPL and static CMOS logic families in power, power-delay product and area.

Hundreds of circuit experiments have ever been processed and found it has the best performance in almost all aspects.

5. Conclusions

In this paper, a novel logic circuit synthesis and optimization procedure, Prioritized Prime Implicant Patterns Puzzle (PPIPP), for arbitrary full swing logic function is proposed. The new proposed logic family proves to be superior to DVL, DPL and CMOS in all aspects with only a few exceptions. The advantages of the propose logic family are low power consumption, low power-delay product and area efficiency. It can clearly handle higher variables, is not limited by Karnaugh map. It's robustness against transistor downsizing and voltage scaling makes it good for deep sub-micron VLSI usage.

6. Refferences

[1] P. Buch, A. Narayan, and A. R. Newton, A. Sangiovanni-Vincentelli, "Logic synthesis for large pass transistor circuits," *ICCAD*, 1997, Page(s): 663 –670.

[2] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-power CMOS digital design", *IEEE Journal of Solid-State Circuit*, Volume: 27 Issue: 4, April 1992, Page(s): 473 – 484.

[3] Kuo-Hsing Cheng and Ven-Chieh Hsieh, "A new logic synthesis and optimization procedure", in *Proc. IEEE Int. Symp. on Circuits and Systems*, 2001.

[4] Hanho Lee and Gerald E. Sobelman "New Low-Voltage Circuit for XOR and XNOR," *Southeastcon '97 Engineering New Century, Proceeding IEEE*, 1997, Page(s): 225-229.

[5] V. G. Oklobdzija, B. Duchene, "Pass-Transistor Dual Value Logic For Low–Power CMOS," *Proceedings of the 1995 International Symposium on VLSI Technology,* Taipei, Taiwan, 1995.

[6] V. G. Oklobdzija, B. Duchene, "Development and Synthesis Method for Pass-Transistor Logic Family for High-Speed and Low Power CMOS," *Proceedings of the 38th Midwest Symposium* on Volume: 1, 1996, Page(s): 298-301 VOL.1.

[7] M. Suzuki, N. Ohkubo, T. Shinbo, T. Yamanaka, et. al. "A 1.5 32-b CMOS ALU in Double PASS-Transistor Logic," *IEEE J. Solid-State Circuits,* vol. 28, no. 11, pp. 1145-1151, November 1993.

[8] J. Wang, S. Fang and W. Feng, "New Efficient Designs for XOR and XNOR Function on the Transistor Level," *IEEE Journal of Solid-State Circuit*, 29(7):780-786, July 1995.

[9] K. Yano; Y. Sasaki; K. Rikino and K. Seki, "Top-down pass-transistor logic design," *IEEE Journal of Solid-State Circuit*, Volume: 31 Issue: 6, June 1996, Page(s): 792 –803.

[10] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE Journal of Solid-State Circuit*, Volume: 32, Issue: 7, July 1997, Page(s): 1079 –1090.