Realizing Many-valued Logic for Computation

Ben Choi, Rong Zheng, Kankana Shukla
Computer Science
Louisiana Tech University, USA
pro@BenChoi.org

Abstract— **This paper proposes to use many-valued logic as new potentials for improving computation speeds. To facilitate the use of many-valued logic for computation, this paper describes a simple four-step process for designing many-valued circuits to implement any many-valued functions. The aim is to design and implement digital circuits entirely within the domain of many-valued logic. In a four-valued logic circuit, each wire carries two bits at a time, each logic gate operates two bits at once, and each memory cell records two bits at one time. To be able to implement four-valued logic circuits in hardware, this paper also contributes new CMOS designs of all the necessary logic gates, including disjoint unary gates, n-input AND gates, and n-input OR gates. The many-valued circuit design methodology and the many-valued logic gates provide the necessary and sufficient tools and components for exploiting the many-valued computational paradigm.**

Index Terms— **Many-valued Logic, Fuzzy Control, Circuit Design, CMOS Design, Fuzzy System**

I. Introduction

The performances of current computers are reaching their limits. Almost all present day computers are built based on two-valued logic. In two-valued logic, each wire can have two states. The performance of current computer depends mostly on how quickly the states can be changed, which determines the clock speed. During the past decades, the clock speed for CPU had doubled almost every year. In recent years, the clock speed doubled every 18 months. Now, it has become progressively difficult to increase the clock speed. The limit is approaching. Recently, CPU manufacturers are trying to circumvent the limitation of clock speed by packing more and more "cores" into a chip, which has resulted in dual-core or quad-core CPUs. However, this multi-core approach does not greatly improve the performance. This is due in part by the limit of the amount of data that can be transferred between the CPU and its connected components, which is determined by the number of pins on the CPU. Using two-value logic each pin on the CPU can have at most two states, and again the amount of data that can be transferred is determined by the clock speed. Thus, the multi-core approach does not circumvent the limitation.

Thus, there is a need for an innovative approach in order to push the speed limit of computing. Now is the time to depart from the two-valued logic to venture into many-valued logic and even into infinite-valued (Fuzzy) logic. Advancing from two-valued to four-valued logic provides an progressive approach [1]. Four symbols {0, 1, 2, 3} are needed to distinguish the four values, as shown in Table 1. The four values might represent anything, for example, the four bases {A, T, C, G} found in DNA, or probability {0, 1/3, 2/3, 1}. These four values can be converted to binary numbers {00, 01, 10, 11}, or they can simply represent integers {0, 1, 2, 3}. It is also possible to start from the ground up by designing components needed for constructing four-valued logic circuits. Each four-valued logic gates will operate two bits of data at a time, and each memory cell will record two bits at once. Now, each wire or CPU pin can have four states, which could double the amount of data that can be transferred between the CPU and its connected components without increasing the number of pins on the CPU. With eight-valued logic, each logic gate operates three bits of data and CPU pin carries three bits of data. The extreme case will be the infinite-valued or Fuzzy logic. Now, a different limit is being pushed.

The approach for using many-valued logic is in fact currently being employed in building higher capacity flash memory. The industry is pushing to allow each memory cell to store not just one bit, but two bits, three bits, and even four bits [2][3][4]. Now, we are proposing to push these limits not just in memory technologies, but also in computation.

To make the many-valued computation possible, this paper provides the necessary and sufficient tools and components for designing many-valued systems entirely within the domain of many-valued logic. We describe a simple four-step process for designing many-valued circuits to implement any many-valued functions. The design of a four-valued adder is provided as an example. By following the simple four-step process, it becomes very convenient to design many-valued circuits to implement any many-valued functions. We also provide new CMOS designs for many-valued disjoint unary gates, n-input AND gates, and n-input OR gates, for fully exploiting many-valued logic and fuzzy paradigm in hardware.

The remaining of this paper is organized as follows. Section II outlines the related research and their limitations. Section III describes using Post Algebra as

Table 1: Representations for a Four-valued Variable

Symbol	DNA	Probability	Bits	Integer
0	A	0	00	0
1	T	1/3	01	1
2	C	2/3	10	2
3	G	1	11	3

the mathematical foundation that facilitates the design process of many-valued circuits. Section IV outlines our simple four-step process for designing many-valued circuits to implement any many-valued functions. Section V shows our CMOS designs and implementations of the necessary and sufficient many-valued logic gates, which serve as the building blocks for implementing the designed many-valued circuits in hardware. Section VI gives the conclusion and outlines the future research.

II. RELATED RESEARCH

To exploit the many-valued computation in hardware, we need the fundamental building blocks for many-valued logic circuits: many-valued logic gates, memory cells, and flip-flops. However, even these essential logic gates and memory cells are not yet fully developed. Currently, many-valued and fuzzy systems [5],[6],[7],[8],[9],[10],[11] are usually simulated or implemented by using a fuzzifier to convert the inputs, using a set of fuzzy rules for processing and inferring, and using a defuzzifier to convert the results to outputs. To go a step further, researchers are now researching on many-valued and fuzzy logic circuits that can fully implement fuzzy systems.

To make the transition from two-valued to many-valued logic circuits, researchers were attempting to adapt CMOS [12],[13] technologies to implement the many-valued and Fuzzy logic gates. The design of the AND gate and the OR gate using CMOS technology was reported [1],[14],[15],[16]. Other technologies, including carbon nanotube and single electron transistors, have been attempted to build many-valued logic circuits [17][18][19]. Other researchers used analog circuits to implement the many-valued and fuzzy logic gates [20],[21],[22]. However, these analog circuits were more difficult to be fabricated.

Many-valued and fuzzy memory cells or fuzzy flip-flops were proposed in [11],[23],[24],[25],[26],[27],[28],[29],[30],[31]. Concept of fuzzy flip-flop was first mentioned by Hirota [23]. They used analog gates [32],[33],[34] for the design their JK-type flip-flop as discussed in [19]. Hirota [23] defined fuzzy JK flip-flop based on the binary JK flip-flop but using fuzzy operators. Their design was based on fuzzy operators such as t-norm, s-norm, and fuzzy negation [35]. Virant et al. [29] proposed a design of T-type fuzzy flip-flop. The researchers adapted a strategy similar to Hirota [23] in the design of the T fuzzy flip-flop. However, we found that the fuzzy memory cells or flip-flops reported previously, such as JK-type flip-flop [23][24][25] and T-type flip-flop [29], have their limitations and cannot fully be used as general fuzzy memory cells. The flip-flops would not produce the correct results under certain input conditions [37].

In this paper, we focused on the design methodologies of many-valued combinatorial circuits that does not require memory cells while that of many-valued sequential circuits (that require memory cells) will be reported elsewhere.

Table 3. Postulates for disjoint system of Post algebras of order $n \geq 2$

P1	$A \cdot B = B \cdot A$	$A+B = B+A$
	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	$(A+B)+C = A+(B+C)$
	$A \cdot A = A$	$A+A = A$
	$(A+B) \cdot A = A$	$(A \cdot B)+A = A$
	$A \cdot (B+C) = (A \cdot B)+(A \cdot C)$	
P2	$e_{n-1} \cdot A = A$	$e_0+A = A$
	$e_i \cdot e_{i+1} = e_i$ for $0<i<n-2$	
P3	$C_i(A) \cdot C_j(A) = e_0$ for $i \neq j$, $0 \leq i, j \leq n-1$	$C_0(A)+C_1(A)+\ldots+C_{n-2}(A)+C_{n-1}(A) = e_{n-1}$
P4	$C_i(A \cdot B) = C_i(A) \cdot [C_i(B)+C_{i+1}(B)+\ldots+C_{n-1}(B)] + C_i(B) \cdot [C_i(A)+C_{i+1}(A)+\ldots+C_{n-1}(A)]$ for $i = 0,1,\ldots,n-1$	$C_{n-1}(A+B) = C_{n-1}(A)+C_{n-1}(B)$
P5	$C_i(e_j) = e_0$ for $i \neq j$, $0 \leq i, j \leq n-1$	
	$C_{n-1}(e_0) = e_0$	
	$C_{n-1}(e_{n-2}) = e_0$	
P6	$e_1 \cdot C_1(A)+e_2 \cdot C_2(A)+\ldots+e_{n-1} \cdot C_{n-1}(A) = A$	

III. USING POST ALGEBRA AS FOUNDATION FOR MANY-VALUED CIRCUIT DESIGN

This section describes the mathematical foundation of many-valued logic that facilitates the design process of many-valued circuits. While Boolean algebra provides the mathematical foundation for designing two-valued digital circuits, Post algebra provides the mathematical foundation for designing many-valued circuits. We choose the disjoint system of Post algebras of order $n \geq 2$ for the reason that the disjoint system facilitates simple design processes (described in Section IV). The postulates for a disjoint system of Post algebras is provided in the following table (based on [36]).

Table 3 defines a disjoin system of Post algebras of order $n \geq 2$. Where, the A, B, and C are n-valued variables. The e_i for $0 \leq i \leq n-1$ are n constants. The $C_i(x)$ for $0 \leq i \leq n-1$ are n disjoint unary operations. The • is the binary operation that represents AND, while the + is the binary operation that represents OR.

Boolean algebras are Post algebras of order 2 as highlighted in Table 4. There are two constants: e_0 denoted by 0, and e_1 denoted by 1. The Boolean NOT(A)

Table 4. Boolean algebras are Post algebras of order 2

	Input	Output	
		NOT(A) =	A =
	A	$C_0(A)$	$C_1(A)$
$F = e_0$	0	1	0
$T = e_1$	1	0	1

Table 5. Post algebras of order 4
(Four-valued logic)

	Input	Output			
	A	$C_0(A)$	$C_1(A)$	$C_2(A)$	$C_3(A)$
$F=e_0$	0	3	0	0	0
e_1	1	0	3	0	0
e_2	2	0	0	3	0
$T=e_3$	3	0	0	0	3

= $C_0(A)$, while $C_1(A)$ = A. The • is equivalent to the Boolean AND operation, while the + is equivalent to the Boolean OR operation.

In the following, we choose, as an example, the disjoint system of Post algebras of order n = 4, and called the system a four-valued logic. As outlined in the following table, we use A as a 4-valued variable. The 4 constants are denoted by 0, 1, 2, 3, are the 4 disjoint unary operations $C_0(A)$, $C_1(A)$, $C_2(A)$, and $C_3(A)$ are defined as shown in Table 5.

The 4-valued AND, OR operations are defined as shown in Table 6, where the AND operation produce as output the minimum of (A, B), while the OR operation produce as output the Maximum of (A, B).

IV. Simple Four-step Process for Designing Multi-valued Circuits

Based on the disjoint system of Post algebras of order $n \geq 2$ defined in Section III, we outline a simple four-step process for designing many-valued circuits to implement any many-valued functions [38]. The four steps are: (0) Creating a truth table to define the function; (1) Connecting each input x to n $C_i(x)$ gates; (2) Creating an AND gate for each output instance having a value > 0; and (3) Connecting the outputs of all the AND gates to an OR gate, which produces the outputs of the required function. These 4 steps are described in more details in the following sections. By following this simple four-step process, implementation of any many-valued function becomes feasible.

Table 6. Four-valued AND (min), OR (Max)

Input		Output	
		AND(A,B)	OR(A,B)
A	B	A·B	A+B
0	0	0	0
0	1	0	1
0	2	0	2
0	3	0	3
1	0	0	1
1	1	1	1
1	2	1	2
1	3	1	3
2	0	0	2
2	1	1	2
2	2	2	2
2	3	2	3
3	0	0	3
3	1	1	3
3	2	2	3
3	3	3	3

Step 0. Truth Table: Creating a truth table to define the many-valued functions

As an example, we choose to design an adder that adds two 4-valued numbers A, B. We create a truth table to define the required functions.

As shown in Table 7, all possible input combinations are shown in column A and B. The results of the addition is encoded by two outputs K and S, where K stands for carry and S stands for sum, and the total value is 4K+S. The column K defines the function required to produce K as output, and the column S defines the function required to produce S as output.

Step 1. $C_i(x)$ gates: Connecting each input x to n $C_i(x)$ gates for $0 \leq i \leq n-1$

Continuing the above example of designing an adder, the adder have two inputs, A and B. Now, we connect input A to 4 $C_i(A)$ gates:

$C_0(A)$, $C_1(A)$, $C_2(A)$, $C_3(A)$

Similarly, we connect input B to 4 $C_i(B)$ gates:

$C_0(B)$, $C_1(B)$, $C_2(B)$, $C_3(B)$

The results of these connection is shown in Figure 1.

Step 2. AND gates: Creating an AND gate for each output instance having a value > 0

For each input instance $A_0,A_1,\ldots A_{m-1} = x_0,x_1,\ldots x_{m-1}$ that produce an output e > 0, create an AND gate connecting:

Table 7. Truth table defining a
four-valued adder

Input		Output	
		4^1x	4^0x
A	B	K	S
0	0	0	0
0	1	0	1
0	2	0	2
0	3	0	3
1	0	0	1
1	1	0	2
1	2	0	3
1	3	1	0
2	0	0	2
2	1	0	3
2	2	1	0
2	3	1	1
3	0	0	3
3	1	1	0
3	2	1	1
3	3	1	2

$$C_{x0}(A_0)\cdot C_{x1}(A_1)\cdot \ldots \cdot C_{xm-1}(A_{m-1})\cdot e$$

For $e = e_{n-1}$, there is no need to connect the AND gate to e, which is the results of simplification based on the postulate P1 that is $e_{n-1}\cdot A = A$.

Continuing the example of designing an adder, for the function that produce S as output (in the S column of the truth table), there are 9 instances that produce output $e > 0$. For example, referring to the truth table, when inputs A=0, B=1, the output S=1, in this case we create an AND gate connecting: $C_0(A)\cdot C_1(B)\cdot 1$, in which since A=0 so the AND gate connects to the output of $C_0(A)$ gate (from Step 1), since B=1 so the AND gate connects to the output of $C_1(B)$ gate (from Step 1), and since S=1 so the AND gate connects to 1. When inputs A=0, B=2, the output S=2, in this case we create an AND gate connecting: $C_0(A)\cdot C_2(B)\cdot 2$, in which since A=0 so the AND gate connects to the output of $C_0(A)$ gate, since B=2 so the AND gate connects to the output of $C_2(B)$ gate, and since S=2 so the AND gate connects to 2. And, when inputs A=0, B=3, the output S=3, in this case we create an AND gate connecting: $C_0(A)\cdot C_3(B)\cdot 3$, which is simplified to $C_0(A)\cdot C_3(B)$. We create 9 AND gates for the 9 instances as shown in the below and the connections are shown in Figure 1.

$C_0(A)\cdot C_1(B)\cdot 1,\ C_0(A)\cdot C_2(B)\cdot 2,\ C_0(A)\cdot C_3(B),$
$C_1(A)\cdot C_0(B)\cdot 1,\ C_1(A)\cdot C_1(B)\cdot 2,\ C_1(A)\cdot C_2(B),$
$C_2(A)\cdot C_0(B)\cdot 2,\ C_2(A)\cdot C_1(B),\ C_2(A)\cdot C_3(B)\cdot 1,$
$C_3(A)\cdot C_0(B),\ C_3(A)\cdot C_2(B)\cdot 1,\ C_3(A)\cdot C_3(B)\cdot 2$

Similarly, for the function that produce K as output (in the K column of the truth table), there are 6 instances that produce output $e > 0$. We create 6 AND gates as shown in the below and the connections are shown in Figure 1.

$C_1(A)\cdot C_3(B)\cdot 1,\ C_2(A)\cdot C_2(B)\cdot 1,\ C_2(A)\cdot C_3(B)\cdot 1,$
$C_3(A)\cdot C_1(B)\cdot 1,\ C_3(A)\cdot C_2(B)\cdot 1,\ C_3(A)\cdot C_3(B)\cdot 1$

Step 3: OR gate: Connecting the outputs of all the AND gates to an OR gate, which produces the outputs of the required function.

Finishing the example of designing an adder, for the function that produce S as output (in the S column of the truth table), we connect the outputs of all the 9 AND gates (from Step 2) to an OR gate, as defined below:

$$\begin{aligned} S = {} & C_0(A)\cdot C_1(B)\cdot 1 + C_0(A)\cdot C_2(B)\cdot 2 + C_0(A)\cdot C_3(B) + \\ & C_1(A)\cdot C_0(B)\cdot 1 + C_1(A)\cdot C_1(B)\cdot 2 + C_1(A)\cdot C_2(B) + \\ & C_2(A)\cdot C_0(B)\cdot 2 + C_2(A)\cdot C_1(B) + C_2(A)\cdot C_3(B)\cdot 1 + \\ & C_3(A)\cdot C_0(B) + C_3(A)\cdot C_2(B)\cdot 1 + C_3(A)\cdot C_3(B)\cdot 2 \end{aligned}$$

Similarly, for the function that produce K as output (in the K column of the truth table), we connect the outputs of all the 6 AND gates (from Step 2) to an OR gate, as defined below:

$$\begin{aligned} K = {} & C_1(A)\cdot C_3(B)\cdot 1 + C_2(A)\cdot C_2(B)\cdot 1 + C_2(A)\cdot C_3(B)\cdot 1 + \\ & C_3(A)\cdot C_1(B)\cdot 1 + C_3(A)\cdot C_2(B)\cdot 1 + C_3(A)\cdot C_3(B)\cdot 1 \end{aligned}$$

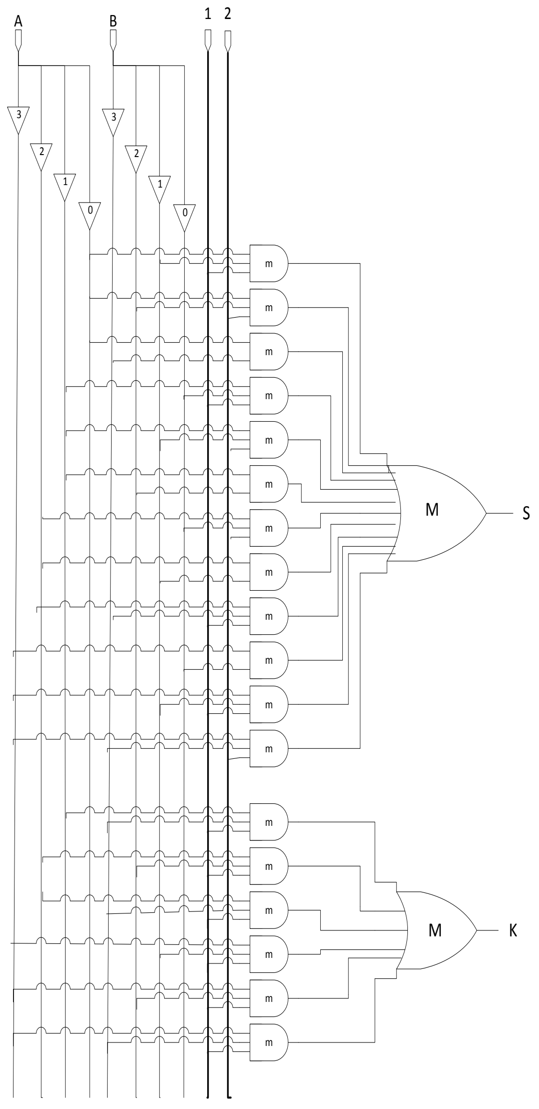


Figure. 1. Four-valued Adder Circuit

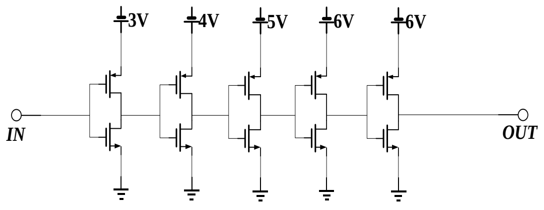
The results all the connections are shown in Figure 1, which is the four-valued circuit that implements the four-valued addition of two four-valued numbers.

V. Designing Multi-valued Logic Gates

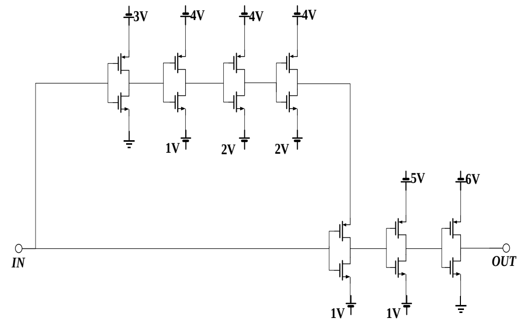
In order to realize the design of many-valued logic circuits in hardware, we need to design and implement the necessary components. Based on the design process described in the last section, we need six four-valued logic gates to implement any four-valued logic circuits. These gates are C_0, C_1, C_2, and C_3 gates (as defined in Table 5) and the AND gate and the OR gate (as defined in Table 6). Our design and implementation of these necessary and sufficient gates are provided in the following.

Our designs of the six necessary four-valued logic gates use CMOS technologies for ready implementation in IC chips. Four logic values 0, 1, 2, and 3 are represented physically by 0V, 2V, 4V, and 6V respectively. There are only 2V between two values and thus the circuits are more subjective to noise. These voltages may be increased if noise causes problems. Figure 2 shows our designs of C_0, C_1, C_2, and C_3 gates and Figure 3 shows the test results, which verified that the designs meet the functions defined in Table 5. Figure 4 shows our design of n-input four-valued AND gate and Figure 5 shows the test results of a 2-input AND gate, which verified that the design meets the function defined in Table 6. Figure 6 shows our design of n-input four-valued OR gate and Figure 7 shows the test results a 2-input OR gate, which verified that the design meets the function defined in Table 6, as well.

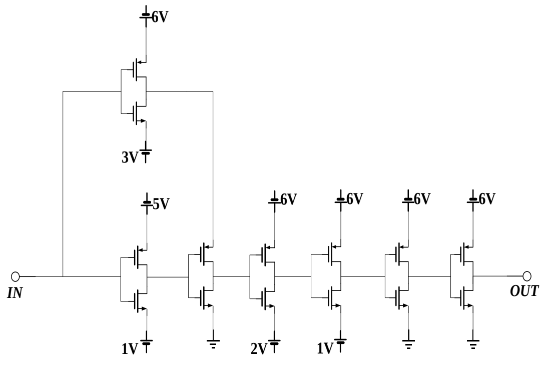
After each of the logic gates has been designed and tested, we then use the logic gates for building circuits. One of our test circuits is the four-valued adder circuit (Figure 1). We implemented the adder circuit using our gates and tested the function. The test results (Figure 8) verified that the adder circuit functions as defined in Table 7. The glitches in the output S (SUM) and the output K (CARRY_OUT) are results of delays in the circuits. After it stables down, the results matched the required output and the circuit function as required. To further test the circuits, we then designed and built a circuit to add three four-valued numbers by using two of the adder circuits. The test results verified that we can combine circuits to build larger many-valued circuits.



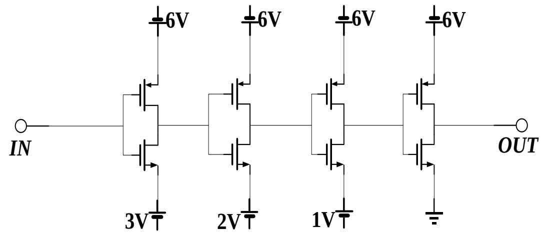
C_0 gate



C_1 gate



C_2 gate



C_3 gate

Figure. 2. Designs of C_0, C_1, C_2, and C_3 gates

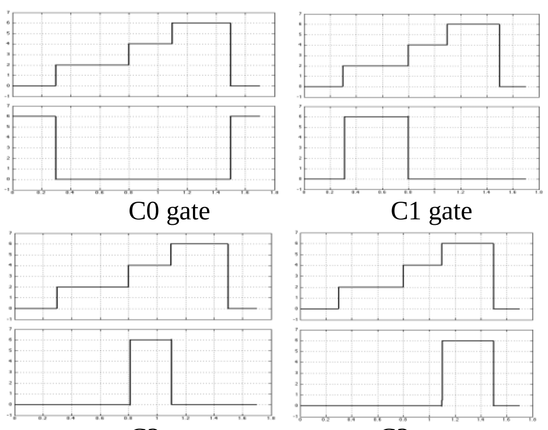


Figure. 3. Test results of C_0, C_1, C_2, and C_3 gates (the top graphs are inputs and the bottom ones are the outputs)

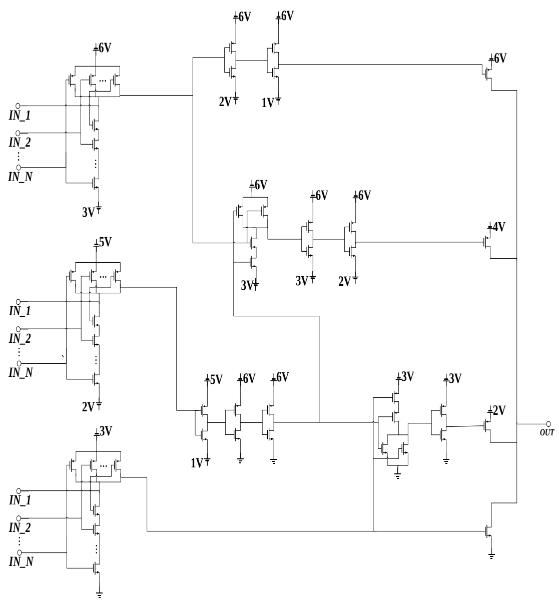


Figure. 4. Design of n-input four-valued AND gate

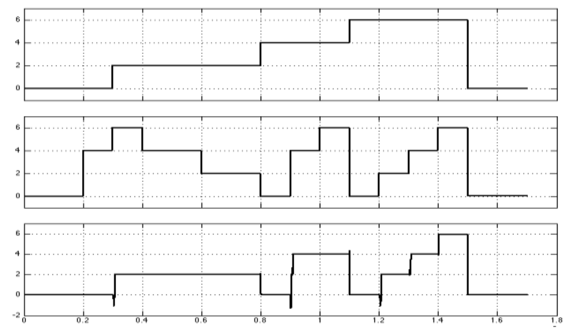


Figure. 5. Test results of a 2-input four-valued AND gate (the output, bottom graph, is the minimum of the 2 inputs, top 2 graphs)

VI. Conclusion and Future Research

Now is the time to depart from the two-valued logic to venture into many-valued logic and even into infinite-valued or Fuzzy logic. To make many-valued computation possible, this paper provides the necessary tools for designing many-valued systems entirely within the domain of many-valued logic. We describe a simple four-step process for feasible design of many-valued circuits to implement any many-valued function.

We also provided CMOS designs of many-valued logic gates, including the Disjoin ($C_i(x)$) gates, n-input AND gates, and n-input OR gates. Thus, it would be feasible to implement the designed many-valued circuits in integrated circuits (IC chips).

Using the simple four-step process to design many-valued circuits do not necessary provide the most simplified circuits. In most case, the circuits can further be simplified, which can be done by algebraic manipulation based on the postulates for the disjoint system of Post algebras provided in Section III.

The next stage for future research will be to use the many-valued circuit design methodology and memory cells [38] to design large-scale circuits for fully exploiting many-valued logics and fuzzy paradigms in hardware.

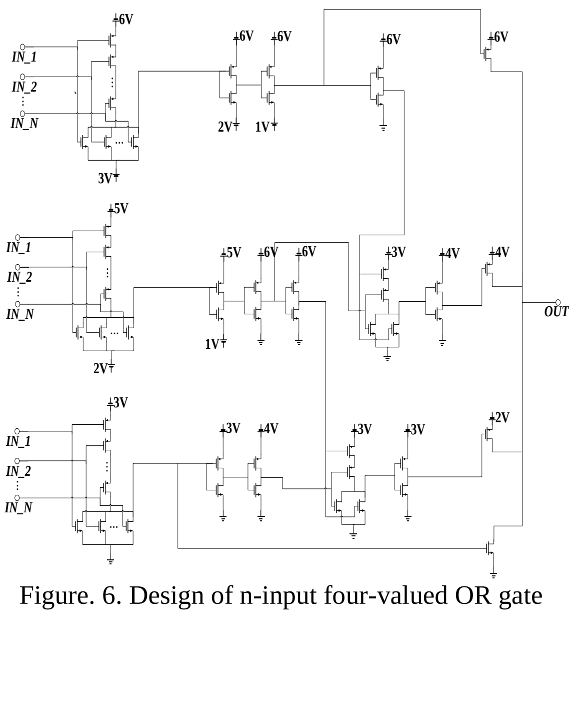


Figure. 6. Design of n-input four-valued OR gate

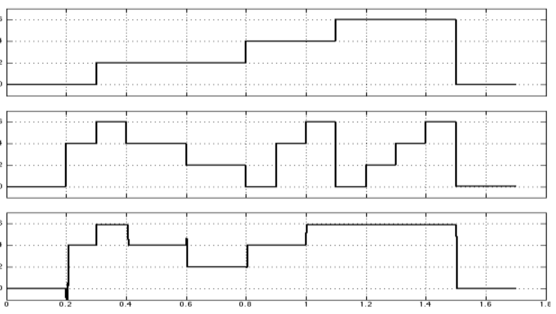


Figure. 7. Test results of a 2-input four-valued OR gate (the output, bottom graph, is the maximum of the 2 inputs, top 2 graphs)

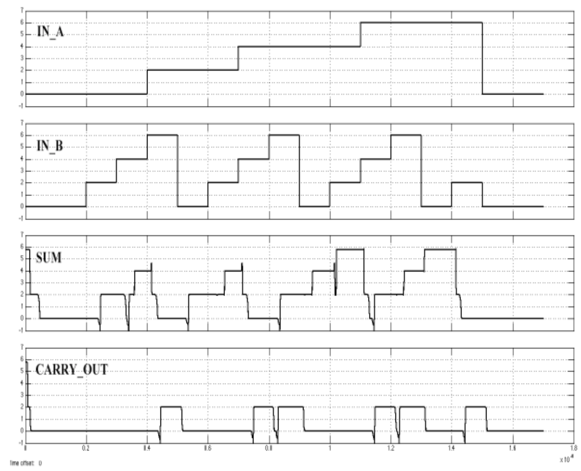


Figure. 8. Test results for the four-valued adder circuit (Figure 1).

References

[1] Choi, B.; "Advancing from Two to Four Valued Logic Circuits," IEEE International Conference on Industrial Technology (ICIT 2013), February 2013.

[2] Kimura K and Kobayashi T, "Trends in high-density flash memory technologies", 2003 IEEE Conference on Electron Devices and Solid-State Circuits, pp. 45-50, Dec. 2003.

[3] Jigour R., "A tour of the basic of embedded NAND flash options", EE Times, Aug. 2013.

[4] Choi Y., "NAND flash – The new era of 4 bit per cell and beyond", EE Times, May, 2009.

[5] Marinos, P; "Fuzzy logic and its application to switching systems" *IEEE Transactions on Computing*, vol. C-18, no.4, p 343-348, Apr 1969.

[6] Zadeh, L.A.; "Fuzzy Logic = Computing with words" *IEEE Transactions on Fuzzy Systems*, vol.4, p 103-11, 1996.

[7] Zadeh, L.A.; "The Concept of Linguistic Variables and its Application Approximate Reasoning," *Information Sciences*, p 43-80, 1975.

[8] Mendel, J.M.; "Fuzzy Logic Systems for Engineering: A Tutorial" Proceedings of *IEEE*, vol. 83, No.3, March 1995.

[9] Isik, C.; "Fuzzy logic: principles, applications and perspectives" *SAE (Society of Automotive Engineers) Transactions*, vol. 100, n Sect 1 pt 1, 1991, 911148, p 393-396

[10] Fattaruso, J.W.; Mahant Shetti, S.S.; Barton, J.B.; " A Fuzzy logic inference processor," *IEEE Journal of solid state circuits*, vol. 29, issue 4, April 1994, p 397-402.

[11] Leslaw, G.; Kluska, J.; "Family of fuzzy J-K flip-flops based on bounded product, bounded sum and complementation" *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, n 6, Dec, 1998, p 861-868.

[12] Weste, N.H.E; Eshraghian, K; "Principles of CMOS VLSI Design", Addison- Wesley Publishing Company, ISBN 0-201-53376-6, 1994, p 61-69.

[13] Baker, R. Jacob; "CMOS circuit design, layout, and simulation", 2nd *ed.* Baker, R. Jacob, 1964.

[14] Catania, V.; Puliafito, A.; Russo, M.; Vita, L.; "A VLSI fuzzy inference processor based on a discrete analog approach," *IEEE Transactions on Fuzzy Systems*, vol. 2, Issue 2, May 1994, p 93-106.

[15] Ascia, G.; Catania, V.; Russo, M.; "VLSI hardware architecture for complex fuzzy systems" IEEE Transaction on Fuzzy systems", vol. 7, issue 5, Oct 1999, p 553-570.

[16] Ascia, G.; Catania, V.; "A high performance processor for application based on fuzzy logic" Fuzzy systems conference proceedings, 1999. FUZZ-IEEE '99. 1999 IEEE international vol. 3, 22-25 Aug. 1999, p1685-1690.

[17] Raychowdhury A. and Roy K., "Carbon-Nanotube-Based Voltage-Mode Multiple-Valued Logic Design," *IEEE Trans. Nanotechnol.*, vol. 4, no. 2, pp. 168–179, Mar. 2005.

[18] Sakhare A. N. and Keote M. L., "Application of Galois Field in VLSI Using Multi-Valued Logic," *Comput. Sci.*, vol. 2, no. 1, 2013.

[19] Wu, G., Cai, L., and Li, Q.; "Ternary logic circuit design based on single electron transistors," *Journal of Semiconductors*, Vol.30, No.2, February 2009.

[20] Hirota, K; "Fuzzy logic and its Hardware implementation" 2nd New Zealand Two-stream international conference on Artificial Neural Networks and Expert systems (ANNES '95), annes, p 102, 1995.

[21] Perez, J.L.; Banuloes, M.A.; "Electronic model on fuzzy gates" *Journal of the Mexican society of instrumentation*, vol. 3, NR 5, 1995, p 43-46.

[22] Catania, V.; Russo, M.; "Analog gates for a VLSI fuzzy processor" *8th International Conference of VLSI Design*, Jan 1995.

[23] Ozawa, K.; Hirota, K.; Koczy, L.T.; Pedrycz, W.; Ikoma, N.; "Summary of fuzzy flip-flop" IEEE International Conference on Fuzzy Systems, vol. 3, International Joint Conference of the 4th IEEE International Conference on Fuzzy Systems and the 2nd International Fuzzy Engineering Symposium, 1995, p 1641-1648

[24] Hirota, K.; Ozawa, K.; "The concept of fuzzy flip-flop" *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, n 5, Sep-Oct, 1989, p 980-997.

[25] Hirota, K.; Pedrycz, W.; "Designing sequential systems with fuzzy J-K flip-flops" *Fuzzy Sets and Systems*, vol. 39, n 3, Feb 15, 1991, p 261.

[26] McLeod, D.; Pedrycz, W.; Diamond, J.; "Fuzzy JK flip-flops as computational structures: design and implementation" *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 41, n 3, Mar, 1994, p 215-226.

[27] Ozawa, K.; Hirota, K.; Koczy, L.T.; Omori, K.; "Algebraic fuzzy flip-flop circuits" *Fuzzy Sets and Systems*, vol. 39, n 2, Jan 25, 1991, p 215.

[28] Hirota, K.; Pedrycz, W.; "Design of fuzzy systems with fuzzy flip-flops" *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, n 1, Jan, 1995, p 169-176.

[29] Virant, J.; Zimic, N.; Mraz, M.; "T-type fuzzy memory cells" *Fuzzy Sets and Systems*, vol. 102, n 2, Mar 1, 1999, p 175-183.

[30] Kia, S.M.; Parmeswaran, S.; "Designs for self checking flip-flops" *IEE Proceedings: Computers and Digital Techniques*, vol. 145, n 2, Mar, 1998, p 81-88.

[31] Miki, T; Yamakawa, T; "Fuzzy Inference on an Analog Fuzzy Chip," IEEE Micro, pp. 8-18, 1995.

[32] Kettner, T,; Heite, C.; Schumacher, K.; " Analog CMOS realization of fuzzy logic membership functions," *IEEE Journal of solid state circuits*, vol. 28, Issue 7, July 1993, p 857-86.

[33] Watanabe, H; W. D. Dettloff, and K.E. Yount, "A VLSI fuzzy logic controller with reconfigurable, cascadable architecture," *IEEE Journal Of Solid-State Circuits*., vol**.** 25, p 376-382, Apr. 1990.

[34] De Venuto, D.; Ohletz, M.J.; Ricco, B.; "Testing of analogue circuits via (standard) digital gates"; *Proceedings. International Symposium on Quality Electronic Design*, 2002.18-21 March 2002, p 112 – 119.

[35] Baturone, I; Barriga, A; Sanchez-Solano, S; Lopez, D.R; "Microelectronic Design of Fuzzy Logic-Based Systems", CRC Press, ISBN 0-8493-0091-6, 2000.

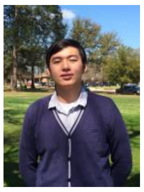
[36] Epstein, G.; *Multiple-Valued Logic Design: An Introduction*, Institute of Physics Publishing, 1993.

[37] Choi, B.; Tipnis, K.; "New Components for Building Fuzzy Logic Circuits," *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, Vol.2, pp. 586-590, 2007.

[38] Choi B. and Shukla K., "Multi-Valued Logic Circuit Design and Implementation," *International Journal of Electronics and Electrical Engineering*, Vol. 3, No. 4, pp. 256-262, August 2015.

Dr. Ben Choi has a Ph.D. degree in Electrical and Computer Engineering and also has a Pilot certificate for flying airplanes and helicopters. He is an Associate Professor in Computer Science at Louisiana Tech University. He received his Ph.D., M.S., and B.S. degrees from The Ohio State University, studied Computer Science, Computer Engineering, and Electrical Engineering. His areas of research include Humanoid Robots, Artificial Intelligence, Machine Learning, Intelligent Agents, Semantic Web, Data Mining, Fuzzy Systems, and Parallel Computing. His future research includes developing advanced software and hardware methods for building intelligent machines and theorizing the Universe as a Computer.

Kankana Shukla works as a Data Scientist at FedEx Services. She holds two Master degrees in Computer Science and in Biomedical Engineering from Louisiana Tech University. She completed her Bachelors in Electronics and Instrumentation. Her research interest includes Data Mining, Big Data Analysis, Machine Learning, Wireless Sensor Networks, Bioinformatics, Robotics and Artificial Intelligence.



Rong Zheng is a Master student in Computer Science at Louisiana Tech University. He completed his Bachelors in Electronic & Information Engineering in Shenzhen University in China. His research interest includes Fuzzy Computing, Machine Learning, Artificial Intelligence, Data Mining and Network Security. His future work includes pursuing a Ph.D. degree in Computer Science.