Advancing from Two to Four Valued Logic Circuits

Ben Choi

Computer Science
Louisiana Tech University, USA
pro@BenChoi.org

Abstract

To further increase the speed of computation, this paper proposed to increase from binary to four-valued logic circuits. Four-valued logic circuits allow each wire to carry two bits at a time, each logic gate to operate two bits at a time, and each memory cell to record two bits at a time. The speed of communication between devices is also increased due to the increase in bandwidth of each wire. To make the base-four computations possible, this paper described the design and implementation of four-valued logic gates, memory cells, and flip-flops. To allow for future developments, the design of the memory cell and the flip-flop provided in this paper can be extended to be used for infinite-valued or Fuzzy logic circuits, for fully exploiting many-valued logics and fuzzy paradigms in hardware.

1. Introduction

The performances of current computers are reaching their limits. Almost all computers of today are built based on two-valued logic. Using two-valued logic each wire can have two states. The performance of current computers depends on most part 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 had doubled every 18 months. Now, it has become more and more difficult to increase the clock speed. The limit is approaching. Now, CPU manufacturers try to circumvent the limitation of speed by packing more and more "cores" into a chip, which resulted in dual-core or quad-core CPUs. However, this multi-core approach does not greatly improve the performances. 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.

We need an innovative approach to push the speed limits. The author proposed that now is the time to depart from the two-valued logic to venture into multi-valued logic and even into infinite-valued (Fuzzy) logic. With four-valued logic, for example, each wire or CPU pin can have four states, carrying two bits of data. This doubles 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 CPU pin now carries three bits of data, and thus triple the amount of data transferring between CPU and its connected devices. The extreme case will be the infinite-valued or Fuzzy logic. We are now pushing a different limit.

We started in small steps by increasing from two to four values. We need four symbols {0, 1, 2, 3} to distinguish the four values, as shown in Table 1. The four values might represent the four bases {A, C, G, T} found in DNA. They might represent probability {0, 1/3, 2/3, 1}. The four values can be converted to binary numbers {00, 01, 10, 11}. And, they can simply represent integer {0, 1, 2, 3}. We also started from the ground up by designing components needed for constructing four-valued logic circuits. Four-valued logic circuits allow each wire to carry two bits at a time, each logic gate to operate two bits at a time, and each memory cell to record two bits at a time. In this paper, we will focus on the design and implementation of four-valued logic gates, memory cells, and flip-flops.

To allow for future developments, the design of the memory cell and the flip-flop provided in this paper can be extended to be used for infinite-valued or Fuzzy logic circuits, for fully exploiting many-valued logics and fuzzy paradigms in hardware.

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

Table 1: Four-Valued Variables

The remaining of this paper is organized as follows. Section 2 outlined the related research and their limitations. Section 3 described our design of a four-valued NOT gate, including the implementation and testing results. Section 4 showed a modified four-valued AND gates. Section 5 showed a modified four-valued OR gates. Section 6 presented our design of a D-type many-valued and fuzzy flip-flop, including the simulation results. And, Section 7 gave the conclusion and outlined the future research.

2. Related Research

To exploit the base-four computations in hardware, we need the fundamental building blocks for four-valued logic circuits: four-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 [30, 31, 20, 14, 9, 18] 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 [28, 3] 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 [5, 2, 1]. We tried to reproduce the results reported in [5. 2] but found that we needed to modify the circuits slightly to make them work (as shown in Section 4 and 5). Other researchers used analog circuits to implement the many-valued and fuzzy logic gates [29, 13, 24, 6]. However, these analog circuits were more difficult to be fabricated.

Many-valued and fuzzy memory cells or fuzzy flip-flops were proposed in [23, 10, 12, 19, 22, 17, 11, 26, 16, 21]. Concept of fuzzy flip-flop was first mentioned by Hirota [23]. They used analog gates [15, 27, 8] for the design their JK-type flip-flop as discussed in [29]. 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. Consider two fuzzy sets x and y in universe of discourse U, a S-norm operation [4] is defined as,

$$\mu_{x\cup y}(u) = \max[\mu_x(u), \mu_y(u)\}],\ \ \forall\ u \in U$$

T-norm operation is defined as

Initial Q	J	K	Q_f		
			1	2	3
0	4	4	4	2	4
2	4	4	4	2	4
4	4	4	2	4	2
6	4	4	2	4	2

Table 2: One Unstable Condition for the Set-type or Reset-type JK Fuzzy Flip-flop

$$\mu_{x\cap y}(u) = \min[\mu_x(u), \mu_y(u)\}],\ \ \forall\ u \in U$$

Fuzzy negation is defined as follows:

$$\mu_{\bar{x}}(u) = 1 - \mu_x(u),\ \ \ \forall\ u \in U$$

Based on the fuzzy operations, Hirota [23] defined set-type and reset-type fuzzy flip-flop. Reset-type fuzzy JK flip-flop has the following characteristic equation:

$$Q_R(t+1) = \{J \wedge (1 - Q(t))\} \vee \{(1-K) \wedge Q(t)\}$$

Characteristic equation for set-type JK fuzzy flip-flop is as follows

$$Q_S(t+1) = \{J \vee Q(t)\} \wedge \{(1-K) \vee (1 - Q(t))\}$$

However, we found that the fuzzy memory cells or flip-flops reported previously, such as JK-type flip-flop [23, 10, 12] and T-type flip-flop [26], 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.

We found several unstable conditions for the set-type and the reset-type JK fuzzy flip-flop defined above. Some such examples are provided in Table 2. While the initial stored value of Q is 0v and when given 4v for inputs J and K, the resulting Q will continuously toggling between 4v and 2v. Similar unstable conditions appears when initial stored value is 2v and given 4v for inputs J and K. Other unstable conditions was also observed, but not shown in the table, for values J=K=6, J=4 K=6, and J=6 K=4.

Therefore, neither the set-type nor the reset-type alone can be used as a fuzzy flip-flop. Hirota [23] combined the characteristics of both set-type and reset-type fuzzy JK flip flop and introduced a fundamental equation for fuzzy JK flip flop. The characteristic equation for min-max type fuzzy JK flip flop is as follows:

$$Q(t+1) = \{J \vee \overline{K}\} \wedge \{J \vee Q(t)\} \wedge \{\overline{K} \vee \overline{Q(t)}\}$$

However, above equation also produced unstable conditions such as some shown in Table 2. The researchers tried to eliminate the above unstable conditions by introducing a pair of complicated sample and hold circuits. The sample and hold circuits latch the output during each clock pulse, thus emulating the behavior of a flip-flop. However, these circuits were difficult to design and cumbersome to modify. Such circuits cannot easily be combined with other fuzzy circuits.

Virant et al. [26] 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. They introduced the following two equations for T fuzzy flip-flop [26]:

$$Q(t+1) = \max\left(\min\left((1-T), Q(t)\right), \min\left(T, (1-Q(t))\right)\right)$$

$$Q(t+1) = \min\left(\max\left(T, Q(t)\right), \max\left((1-T), (1-Q(t))\right)\right)$$

However, the T fuzzy flip-flop has its own limitation. For example, it cannot be connected in such a way to produce a D-type fuzzy flip-flop.

In this paper, we proposed a fuzzy memory cell that can also function as a D-type fuzzy flip-flop. Our fuzzy memory cell can store any value ranging from zero to one, such as the four-valued case {0, 1/3, 2/3, 1}. Furthermore, it was built entirely based on fuzzy logic gates.

3. Our Design of a Four-valued NOT Gate

We began from the ground up by considering four-valued operations that take one input and produce one output. Let A be the four-valued input that takes the values {0, 1/3, 2/3, 1}, and let Y be the four-valued output. For any value of input A, there are four possible values of outputs. Thus, there are 4^4 possible operations. An important operation is the NOT operation, which could be defined as: NOT(A) = 1 – A.

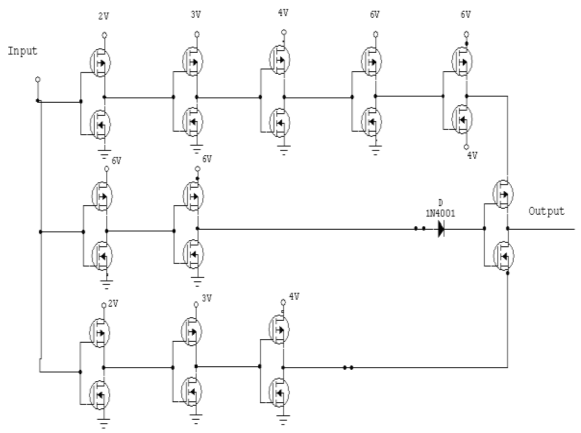
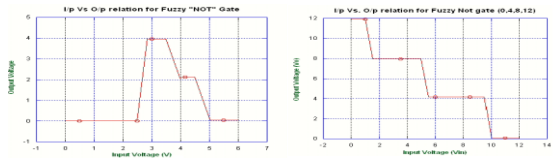


Fig. 1. Our Design of Four-valued NOT gate



(a) max voltage 6v (b) max voltage 12v
Fig. 2. Test results of Four-valued NOT gate

We attempted to design a four-valued NOT gate using digital approach [7]. The design is shown in Fig. 1. This design uses only digital NOT gate as building block. As shown in the figure, the digital NOT gates are supplied by different value of voltages. In this design, we use (0v, 2v, 4v, 6v) to represent the logic values (0, 1/3, 2/3, 1). During our design, we tried many different combinations and so far, the combination of voltage in the figure produced the best result in the case that we limited the maximum voltage to be 6v. During our testing, we found that a diode (as shown in the Fig. 1) was needed to prevent reverse current.

Fig. 2(a) shows the result of one of our tests. The gate begins to behave like a four-valued NOT gate after the input voltage rising to 2.5v. This is due to the threshold voltage required to switch the transistors. If we allowed the supply voltage to reach maximum 12v and scaled the supply voltages to each digital NOT gate accordingly, we were able to get better results as shown in Fig. 2 (b).

Although this digital approach is easier to be implemented in microchip technology, it requires several supply voltages.

4. A Modified Four-valued AND gate

We now consider four-valued operations that take two inputs and produce one input. Let the inputs be A, B, each of which takes four values, and let Y be the four-valued output. With two four-valued inputs, there are 16 possible combinations of input values, and for each of these combinations there are 4 possible outputs. Thus, there are 4^{16} possible operations. Two

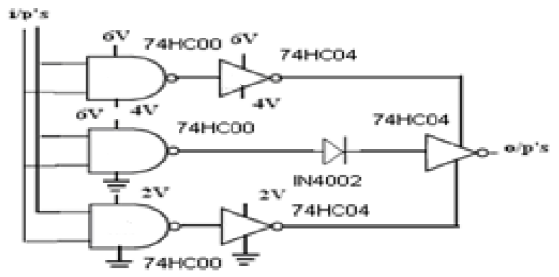
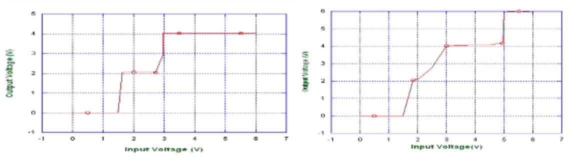


Fig. 3. A Four-valued AND gate


(a) One input held at 4v (b) One input held at 6v
Fig. 4. Test Results of the Four-valued AND gate

important operations are the AND operation and the OR operation. In this section, we show a design of a four-valued AND gates while the design of the OR gate is shown in the next section.

The four-valued AND operation could be defined as: AND(A, B) = min(A, B). For example, given A = 0, and B = 2/3, AND(A, B) = 0. Fig. 3 shows the circuit of a modified four-valued AND gate, which was based on the design provided in [5, 2]. When we tried to implement the design reported in [5] using discrete components, such as microchip 74HC04 and 74HC00, we encountered small problem and needed to introduce a diode as shown in Fig. 3. The diode blocks reverse current from the output, without which wrong results are produced. This modification may reflect the special needs in the use of discrete components and does not necessary imply any fault in the design.

One test result of the AND gate is shown in Fig. 4(a). In this test, one of the input to the AND gate is held at 4v, while the other input allows to vary from 0 to 6v. The output is shown in the figure. Fig. 4(b) shows another test care where one of the inputs is held at 6v and the other input is allowed to vary from 0 to 6v. These test results does not fully matched the ideal behavior. For example, the ideal case for the output in Fig. 3 should be prefect steps. Nevertheless, these results showed the AND gate behaves like a four-valued logic gate having min function.

5. A Modified Four-valued OR gate

The four-valued OR operation could be defined as: OR(A, B) = Max(A, B). For example, given A = 0, and

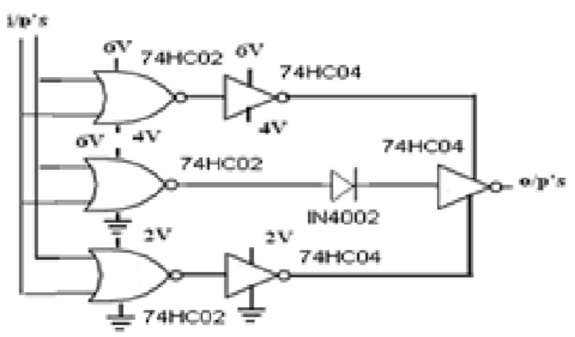
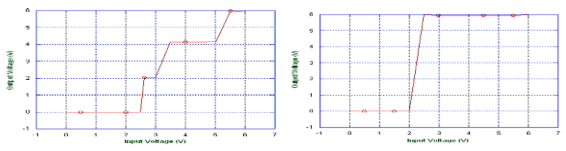


Fig. 5. A Four-valued OR gate


(a) One input held at 2v (b) One input held at 6v
Fig. 6. Test results of the modified OR gate

B = 2/3, OR(A, B) = 2/3. Fig. 5 shows the circuit of a modified four-valued OR gate, which is based on the design provided in [5, 2]. The modification is similar to that of the AND gate by introducing one diode to prevent the reverse current. Fig. 6 show some test results. Fig. 6(a) shows the results of a test care where one input is held at 2v and the other input is allowed to vary from 0 to 6v. Fig. 6(b) shows the results of another test care where one input is held at 6v and the other input is allowed to vary from 0 to 6v. Despite some imperfections, the results show the OR gate behaves like many-valued OR gate having max function.

6. Our Design of a D-type Fuzzy Flip-flop

In this section, we present our design of a D-type fuzzy flip-flop or fuzzy memory cell [7]. Our design is based on an extension of the idea of binary D flip-flop. Excitation table for binary D flip flop is shown in Table 3. The next state $Q(t+1)$ of a D fuzzy flip-flop is characterized as a function of both the present state $Q(t)$ and the input state D. Min term expression for $Q(t+1)$ is

$$Q(t+1) = DQ(t) + D\overline{Q(t)}$$

Above equation is also referred to as the characteristic equation of the D Flip-flop. A mutually equivalent equation can be derived from Table 3 consisting of max terms

$$Q(t+1) = \left(D + Q(t)\right)\cdot\left(D + \overline{Q(t)}\right)$$

Above two equations can be transformed to fuzzy domain by replacing the binary operators by fuzzy operators. They can be redefined using min-max type operation and fuzzy negation as follows:

$$Q(t+1) = \{(1-Q(t)) \wedge D\} \vee \{Q(t) \wedge D\}$$
$$Q(t+1) = \{(1-Q(t)) \vee D\} \wedge \{Q(t) \vee D\}$$

In which the $\wedge$ represents min operation and $\vee$ represents max operation. These two equations, however, do not completely transform D flip-flop to

Q(t)	D(t)	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Table 3. Excitation Table for Binary D flip-flop.

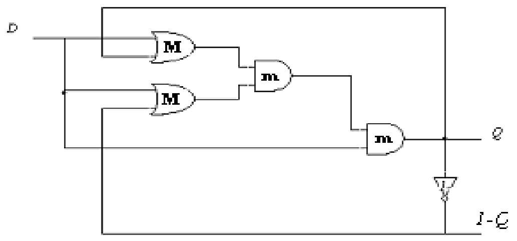


Fig. 7. A New D-type Fuzzy Flip-Flop

the fuzzy domain. Hence, we proposed an equation that has the characteristics of both the equations and also exhibits an analogy with the binary counterpart, as follows:

$$Q(t+1) = \{D\} \wedge \{D \vee Q(t)\} \wedge \{(1-Q(t)) \vee D\}$$

This equation has led to realization of the circuit of D-type fuzzy flip-flop. The design of the new D fuzzy flip-flop is shown in Fig. 7, in which the gates are fuzzy logic AND, OR, and NOT gates. This D-type fuzzy flip-flop can be used as a fuzzy memory cell.

Working of the D-type fuzzy flip-flop (shown in Fig. 7) can be understood by initially considering binary values 0 or 1. If the value of the input D is set at either 0 or 1 regardless the initial value of Q at time t, Q will be set to the value of D. Any value ranging from 0 to 1 also produce the required results. To get an initial idea of the behavior of the D fuzzy flip-flop, we simulated our design using MATLAB and Simulink [25]. Fig. 8 shows the setup of the simulation and the results are shown in Fig. 9. The results show that the D fuzzy flip-flop is simply storing whatever value provided on the input D. It is simply a fuzzy memory cell.

We extended the fuzzy memory cell to clocked D fuzzy flip-flop. Fig. 10 shows our design of the fuzzy flip-flop. This clocked D fuzzy flip-flop can be used in the design of sequential fuzzy circuits.

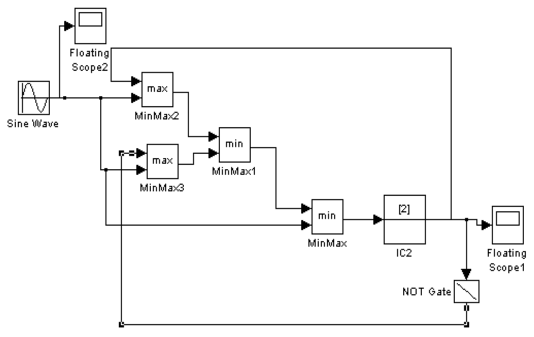
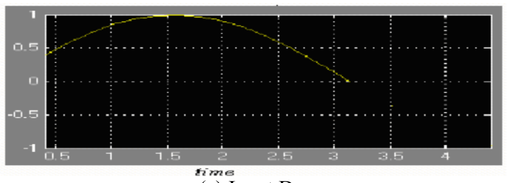
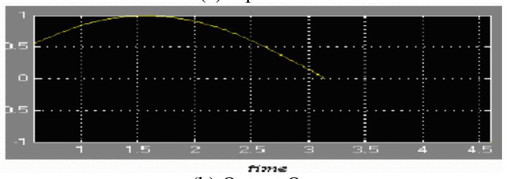


Fig. 8. Simulation Setup of D fuzzy flip-flop using Simulink



(a) Input D



(b) Output Q

Fig. 9. Simulated Result of D fuzzy flip-flop

7. Conclusion and Future Research

The author proposed that now is the time to depart from the two-valued logic to venture into multi-valued logic and even into infinite-valued or Fuzzy logic. However, even the essential logic gates and memory cells for many-valued and Fuzzy logic circuits are not yet fully developed. This paper focused on the development of four-valued logic gates, memory cells, and flip-flops for exploiting the base-four computations in hardware.

This paper presented the design of the first D-type fuzzy flip-flop that can also be used as a fuzzy memory cell. It also presented the circuit of a clocked D-type fuzzy flip-flop that can be used in the design of sequential fuzzy circuits. The D-type fuzzy flip-flop is a truly fuzzy component that allows any logical value arranging from 0 to 1 to be stored. The fuzzy flip-flop is designed entire in the fuzzy domain using fuzzy AND gate, fuzzy OR gate, and fuzzy NOT gate. Thus, the realization of the flip-flop depends on the realization of the fuzzy logic gates. We then investigated the hardware realization of the fuzzy logic gates. In here, we concentrated on digital approach, which resulted in many-valued logic gates. We slightly modified existent designs of the four-valued AND gate and OR gate in an attempt to test them using discrete components. We then proposed a design of four-valued NOT gate. However, using off-the-shelf

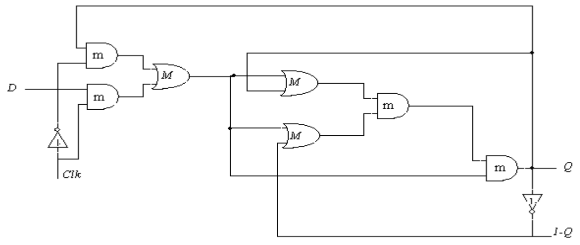


Fig. 10. Clocked D-type Fuzzy Flip-Flop

discrete components, the test results do not produce ideal outputs.

Future research may focus on the improvement on the realization of many-valued and fuzzy logic gates. The next stage for future research will be to use these many-valued and fuzzy logic gates and flip-flops to design large-scale circuits for fully exploiting many-valued logics and fuzzy paradigms in hardware.

REFERENCES

1. 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.
2. 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.
3. Baker, R. Jacob; "CMOS circuit design, layout, and simulation", 2nd *ed.* Baker, R. Jacob, 1964.
4. 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.
5. 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.
6. Catania, V.; Russo, M.; "Analog gates for a VLSI fuzzy processor" *8th International Conference of VLSI Design*, Jan 1995.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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
15. 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.
16. 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.
17. 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.
18. Marinos, P; "Fuzzy logic and its application to switching systems" *IEEE Transactions on Computing*, vol. C-18, no.4, p 343-348, Apr 1969.
19. 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.
20. Mendel, J.M.; "Fuzzy Logic Systems for Engineering: A Tutorial" Proceedings of *IEEE*, vol. 83, No.3, March 1995.
21. Miki, T; Yamakawa, T; "Fuzzy Inference on an Analog Fuzzy Chip," IEEE Micro, pp. 8-18, 1995.
22. 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.
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. 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.
25. Simulink; "Simulink: Simulation and Model Based Design" version 6, The Mathworks, 2005.
26. 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.
27. 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.
28. Weste, N.H.E; Eshraghian, K; "Principles of CMOS VLSI Design", Addison- Wesley Publishing Company, ISBN 0-201-53376-6, 1994, p 61-69.
29. Yamakawa, T; Inoue, T; Ueno, F; and Shirai, Y; "Implementation of Fuzzy Logic hardware systems- Three fundamental arithmetic circuits." *Trans Inst. Electron Common. Eng. Japan*, vol. J63-C, no.10, Oct 1980. p 720-721.
30. Zadeh, L.A.; "Fuzzy Logic = Computing with words" *IEEE Transactions on Fuzzy Systems*, vol.4, p 103-11, 1996.
31. Zadeh, L.A.; "The Concept of Linguistic Variables and its Application Approximate Reasoning," *Information Sciences*, p 43-80, 1975.