VLSI Implementation of Basic Fuzzy *t* -norms

Antonio Hernandez, Oscar Camacho,
Ildar Batyrshin, Luis Villa and
Oswaldo Espinosa Sosa, CIC-IPN/IMP

Abstract

Fuzzy theory applications have been explored and analyzed on different fields such as pattern recognition control, data classification, signal processing, expert systems, among others. To accomplish this, more complex calculations and faster processing speed are required. Fuzzy hardware implementation has turned to be the choice to implement it. Fuzzy operations t-norms and t-conorms are used in fuzzy systems as disjunction and conjunction operations respectively. Commonly used t-norms for hardware implementation are minimum and algebraic product, first one is cheaper to implement; second consumes lots of hardware. On this work FPGA technology is used to implement basic fuzzy T-norms as minimum, bounded product and drastic product which are analyzed and compared each other. Results show hardware resources consumption and processing time required for each operation. These blocks can be used to construct more complex t-norms with sufficiently simple implementation.

1. Introduction

Fuzzy logic has been widely accepted as a reasoning method for process control, automated systems [1], pattern recognition [2], pattern classification [3] and decision making [4], among others. This is because of its capabilities of modeling non-linear systems by using knowledge rules. There are researchers that have shown all advantages that fuzzy logic controllers offer, remarking them as universal approximators [5-8].

As applications of fuzzy theory have grown to different fields, more complex calculations and faster processing speed are required in order to satisfy application demands. Because of this, fuzzy hardware has turned to be the choice to implement faster applications that meet previously stated requirements.

There have been many implementations of fuzzy processors [9-13], that can be divided in two main categories: Analog and Digital. First category uses discrete components to evaluate current or voltage levels to operate with them; Second category uses basic digital gates such as AND, OR and NOT, to create structures that realizes operations required using binary values.

Digital hardware provides better noise immunity than analog counterpart does, besides, usage of FPGA technology for implementing reconfigurable designs allow us to provide great versatility to implemented systems.

Inference procedure is realized with basic operations known as *t*-norms for conjunction, and *t*-conorms for disjunction [14-16]. For the case of *t*-norms, we can mention simple structures such as minimum and algebraic product, which are most commonly used on hardware implementations because of its simplicity. Other known *t*-norms are also candidates for digital implementations, e.g. drastic product and bounded product, the latter is also known as Lukasciewicz *t*-norm. From these basic operations, more complex structures can be constructed to adapt to a wider variety of fuzzy applications.

For doing this, blocks of fuzzy *t*-norm operations are required, so that we can interconnect some of them on different configurations obtaining more options to choose the appropriated for a specific purpose, increasing versatility of fuzzy hardware.

The paper is organized as follows. In Section 2 basic *t*-norms are considered and in Section 3 digital implementation of membership values used in *t*-norms is discussed. Disadvantages of algebraic product is discussed in Section 4. Proposed hardware implementation of basic *t*-norms is discussed in

Sections 5 and 6. In Section 7-8 results and conclusions are commented, including comparisons between operations implemented.

2. Basic *t*-norms

Intersection operation for fuzzy sets is usually defied in fuzzy set theory by some *t*-norm (triangular norm) [16][21] $T:[0,1]\times[0,1]\rightarrow[0,1]$, defined as associative, commutative monotonically increasing function satisfying boundary condition *T(x,1)=1.* For a given pair of input membership functions *X, Y* intersection operation defined for all elements *x, y* of input domains as following:

$$\mu_{X\cap Y}(x)=T(\mu_X(x),\mu_Y(x)) \quad (1)$$

The following *t*-norm operators are candidates for hardware realization: Minimum (2), Algebraic Product (3), Bounded Product or Lukaasiewicz (4) and Drastic Product (5) [16][21].

$$T_M(x,y)=\min\{x,y\} \quad (2)$$

$$T_P(x,y)=xy \quad (3)$$

$$T_L(x,y)=\max\{0,x+y-1\} \quad (4)$$

$$T_D(x,y)=\begin{cases} x, & if\ y=1 \\ y, & if\ x=1 \\ 0, & if\ x,y<1 \end{cases} \quad (5)$$

3. Digital Considerations of *t*-norms

Values of truth space for fuzzy numbers are commonly expressed as floating point number whose truth value can be infinite between [0, 1], where 0 means non-membership and 1 means complete membership. Unfortunately it is not so easy for a computer to do calculations by using this representation, instead of this we can work with integers between [0, 2^n-1], where *n* is the number of bits used to represent truth space and gives resolution, 0 means non-membership, and 2^n-1 is the complete membership value. Taking into account these considerations, equations (4) and (5) on section 2 can be rewritten using *x, y* as integers, and $d=2^n-1$ instead of 1 as maximum value [17].

$$T_L(x,y)=\max\{0,x+y-d\} \quad (6)$$

$$T_D(x,y)=\begin{cases} x, & if\ y=d \\ y, & if\ x=d \\ 0, & if\ x,y<d \end{cases} \quad (7)$$

4. Disadvantages of Algebraic Product

This operation is much more complex because of its nature, if real numbers less than one are used to represent truth values, every product realized will be less than 1, but using integers, overflow will occur and more bits are used at output as twice input bits reason. There are some other reasons listed next:

1. It requires at least n-1 iterations on a sequential multiplier [18].
2. For the case of a combinatorial array circuit *n*-1 levels of adders are required for each pair of *n* bits used for input length [19].
3. Shifting operation can be realized but only for multiplying a number by a power of two [18].

5. Hardware implementation of *t*-norms

5.1. Minimum

For the case of *min* operation (2), a comparator circuit makes a comparison between two input values *A* and *B* and activate flags that let us know if A is greater than *B* (GT) or *A* is less than B (LT). This flags are used as input to a second block where selection of bus output is realized through SEL according to: If *A* greater than *B* is true (GT=1), it selects data on bus *B* as output; if it is false (GT=0), data bus *A* is selected as output. This way the minimum 8 bit value passes to the output as on Fig. 1.

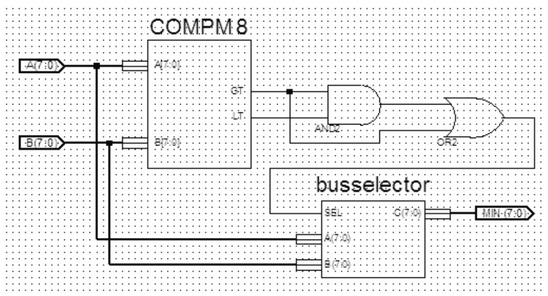


Figure 1. Harware Realized for Minimum Operator

5.2. Bounded Product

Bounded product, also known as Lukasiewicz *t*-norm, is presented on (6) where *d* is the maximum point over the truth space, this implementation has already been presented on [20] using a complex structure. Circuit presented in Fig. 2, corresponds to bounded product operation with 8-bit resolution, according to considerations on section 3. On first block from left down (ADD8), there is an 8 bit adder to implement $x+y$, if the sum of this values is greater than 8 bit maximum count, there is a carry output (CO) flag to indicate that data is not valid. Next block left top (ADSU8) is an 8 bit subtractor, it subtracts *d* from result of previous block; there is also a carry output flag to identify when data is not valid.

In order to obtain a valid data up to this part it is necessary to have two valid conditions on two previous blocks, this corresponds to CO=0 on the addition block and CO=1 on the subtraction block, this is realized by an OR gate between carry flags, managing any undesired condition. Next two blocks COMPM8 and BusSelector correspond to the maximum operation between 0 and valid result from previous stage, AND gate between both blocks is on charge to decide if there is no valid data, let 0 be the output value.

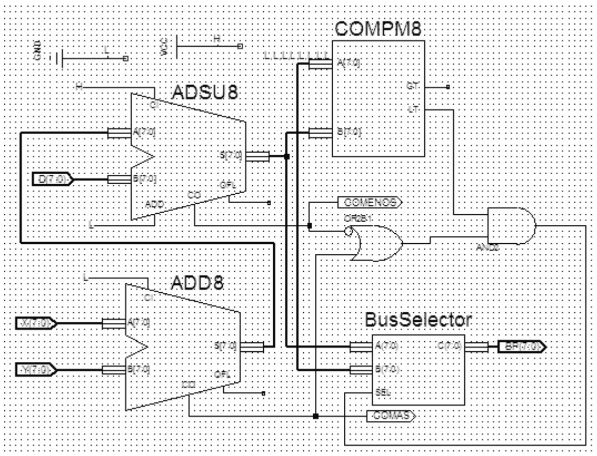


Figure 2. Digital hardware for bounded product operation

5.3. Drastic Product

Drastic product presented on (7) where *d* is the maximum point over the truth space, is a set of conditions that determine the output. Referring to Fig. 3 where circuit implementation is shown, and starting from left, two comparisons are required to satisfy conditions given in equation: first condition is among *x* and *d* (COMPM8 up); second condition is with *y* and *d* (COMPM8 down), result of comparisons is evaluated through AND gates, which give the necessary signals to enable one of three 8-bit outputs, from top down, $x = d$; $x, y < d$, and $y = d$, respectively.

AND2_8B are blocks that use SEL signal to let an 8-bit data number go to the output, when SEL = 1, or make output equal to zero, when SEL = 0. OR3_8B is an OR gate for three 8-bit data buses with bit to bit evaluation, output is the result of drastic product operation with *x*, *y* and *d* as input data.

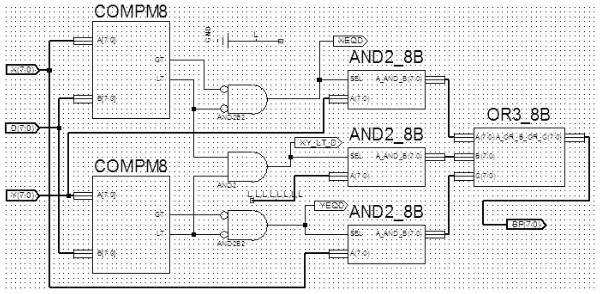


Figure 3. Digital circuitry for drastic product

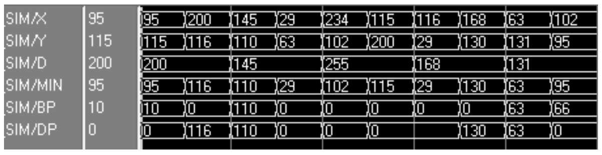
6. Evaluation Tools

For this project realization a Xilinx Spartan3 XC3S50-5PQ208 FPGA was used to implement circuits. For design purpose, XilinxISE Webpack tools were used on schematic mode to allow direct gate level designs. Simulation is performed on ModelSim XE which is designed to work with files generated by XilinxISE.

7. Results

7.1. Simulation Results

Numerical data were introduced as inputs to the circuits for *x, y* and *d*, for different cases as on Fig. 4, where last three rows correspond to the result of operations minimum, bounded product and drastic product on that order from top.



```
SIM/X    95   95   200  145  29   234  115  116  168  63   102
SIM/Y    115  115  116  110  63   102  200  29   130  131  95
SIM/D    200  200            145            255            168            131
SIM/MIN  95   95   116  110  29   102  115  29   130  63   95
SIM/BP   10   10   0    110  0    0    0    0    0    63   66
SIM/DP   0    0    116  110  0    0    0         130  63   0
```

Figure 4. Simulation of operations with input data

7.1. Numerical Results

After file compilation with XST tool we have numerical results that reflect the hardware resources consumed by each circuit, this is shown in Table 1, where drastic product has the lower number of logic levels, but minimum is the one who uses lower cell number, this is because drastic product uses more parallel elements. IO's are less on minimum which has 2 8-bit inputs and one 8-bit output, for the case of bounded and drastic products numbers are alike this is because they have three 8-bit inputs and one 8-bit output.

Table 2 shows timing results for each implementation, where total delay is the time required to obtain a result on the output and is expressed on nanoseconds; middle column corresponds to the time consumed on logical gates, while last column correspond to the time required to connect logical elements. As can be seen drastic product is the one which consumes less processing time, analog to bounded product, which is the most time consuming operation. This result is graphically expressed in Fig. 5, where can be easily seen that drastic product is the faster application.

Table 1. Hardware resources consumed

Operatio n	Logic Levels	Used cells	IO Used
Minimum	13	89	24
Bounded Product	25	177	34
Drastic Product	12	149	35

Table 2. Circuits Delay

Operation	Total Delay ns	Delay Logic ns	Delay Route ns
Minimum	18.73	10.41	8.324
Bounded Product	22.76	13.82	8.940
Drastic Product	17.55	9.93	7.620

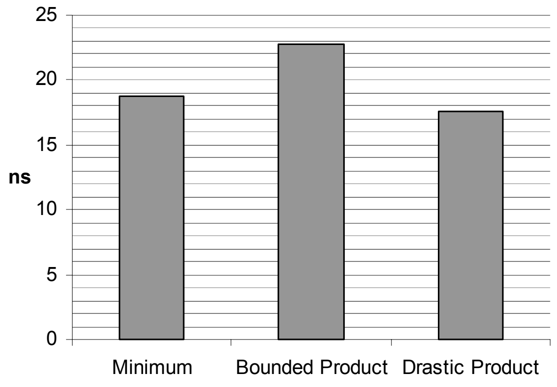


Figure 5. Time required to give a result on output

8. Conclusions

From this work we can conclude that drastic product is the fastest application given its parallel elements, because of this it requires the major quantity of logical gates. If application demands few hardware elements, minimum is the best choice. If there is no restrictions on resources or speed, bounded product is a good choice.

Implementations presented here can be used to expand application fields of fuzzy logic systems, because they can be chained to form more complex structures as parametric forms.

9. References

[1] R. Isermann, "On fuzzy logic applications for automatic control, supervision, and fault diagnosis", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 28, Issue 2, 1998, pp. 221-235.

[2] J. C. Bezdek, "Fuzzy models & algorithms for pattern recognition & image processing", *Handbook of fuzzy sets*, Vol. 4, Lavoisier, 2005.

[3] N.B. Karayiannis, G. Purushothaman, "Fuzzy pattern classification using feedforward neural networks with multilevel hidden neurons", *IEEE International Conference on Neural Networks and Computational Intelligence*, Vol. 3, 1994, pp. 1577-1582.

[4] Chen, S.-M. "A new approach to handling fuzzy decision-making problems", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 18, Issue: 6, Nov/Dec 1988, pp. 1012-1016.

[5] B. Kosko, "Fuzzy systems as universal aproximators", *IEEE International Conference on Fuzzy systems*, Vol. 43 No. 11, 1992, pp. 1329-1333.
[6] L. X. Wang, "Fuzzy systems are universal approximators", *IEEE Transactions on Systems Man and Cybernetics*, 1992, pp. 1163-1170.

[7] J. J. Buckley, "Sugeno type controllers are universal controllers", *Fuzzy sets and systems*, Vol. 53 No. 3, 1993, pp. 299-303.

[8] J. L. Castro, "Fuzzy logic controllers are universal aproximators", *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 25 No. 4, 1995, pp. 629-635.

[9] M. Togai. "Expert system on a chip: an engine for real-time approximate reasoning", *Proceedings of the ACM SIGART international symposium on Methodologies for intelligent systems*, 1986, pp. 147-154.

[10] H. Watanabe, "RISC approach to design of fuzzy processor architecture", *Proceedings First IEEE International Conference on Fuzzy Systems*, San Diego, CA. 1992. pp. 431–441.

[11] G.C. Cardarilli, M. Re, R. Lojacono, "VLSI Implementation of a Real Time Fuzzy Processor", *Journal of Intelligent & Fuzzy Systems*, Vol. 6 No. 3, 1998, pp. 389-401.

[12] A. Gabrielli, E. Gandolfi, "A Fast Digital Fuzzy Processor", *IEEE Micro*, Vol. 19, Issue 1, 1999, pp. 68-79.

[13] G. Ascia, V. Catania, and M. Russo, "VLSI Hardware Architecture for Complex Fuzzy Systems", *IEEE Transactions on Fuzzy Systems*, Vol.7 No.5, 1999, pp. 553-570.

[14] L. A. Zadeh, "Fuzzy logic, Neural Networks, and Soft Computing", *Communications of the ACM*, Vol. 37, Issue 3, March 1994, pp. 77-84.

[15] C.C. Lee, "Fuzzy logic in control systems: fuzzy logic controller. Part II", *IEEE Transactions on Systems, Man and Cybernetics*, Volume 20, Issue 2, March-April 1990, pp. 419-435.

[16] Jang, J.S. R., C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1996.

[17] Tocci, R. J., N. S. Widmer, G. L. Moss, *Digital Systems: Principles and Applications*, 9th edition Prentice Hall, 2003.

[18] Patterson, D. A., and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface,* Second edition, Morgan Kaufmann Publishers, 1998.

[19] Zargham, Mehdi R., *Computer architecture: Single and Parallel Systems,* Prentice Hall, 1996.

[20] L. de Salvador Carrasco, J. Gutiérrez Ríos, "Serial Architecture for Efficient Digital Fuzzy Hardware Processing", On, *Fuzzy Hardware Architectures and applications,* A. Kandel and G. Langholz, eds., Kluwer Academic Publishers, 1998, Chap. 6, pp. 117-157.

[21] Klement E.P., Mesiar R., Pap E. *Triangular Norms.* Dordrecht, Kluwer, 2000.