

# Soft-Core Implementation for Centre of Slice Area Average Defuzzifier

Antonio Hernández Zavala

Electronics Department

Unidad profesional interdisciplinaria en Ingeniería y  
Tecnologías Avanzadas – Instituto Politécnico Nacional  
Distrito Federal, México  
anhernandez@ipn.mx

Oscar Camacho Nieto, Cornelio Yáñez Márquez

Microelectronics Department

Centro de Investigación en Computación – Instituto  
Politécnico Nacional  
Distrito Federal, México  
{oscarc, cyanez}@cic.ipn.mx

**Abstract**—Fuzzy systems are broad used for the decision making task given its capability of extract knowledge from an expert and translate it into rules that define described system. Implementations are made according to speed requirements of the problem, for the case of critical real time systems hardware is the best choice. One important issue on fuzzy systems is the final step, defuzzifier, when computed fuzzy data is turned into crisp data. There are many different defuzzifier methods from which most used for hardware purposes are Center Of Gravity and Mean Of Maxima. This work presents hardware implementation for Center of Slice Area Average defuzzifier, for digital soft-core implementation; its main advantages are reduction on resource consumption and reduction of calculation time without loss of accuracy.

**Keywords:** *Soft Computing, Fuzzy Systems, Digital Systems, Discrete Numbers, Defuzzification*

## I. INTRODUCTION

Fuzzy systems are broad used for the decision making task given its capability of extract knowledge from an expert and translate it into rules that define described system providing better results than conventional controllers. Examples of successful implementations for fuzzy systems are: automatic parking car [1], nuclear reactors [2], robotics [3], function approximation [4], physics experiments [5], database Querying [6] among others.

A rule based fuzzy system is a special type of fuzzy systems that use fuzzy numbers as data for processing. It is constructed of four basic blocks known as: Fuzzifier, rule base, inference and defuzzifier. First and last blocks are used as interface between real world values and fuzzy values at input and output respectively. Rule base contains acquired knowledge from an expert that is managed by inference stage according to data retrieved from fuzzifier stage and delivers its result to defuzzifier stage.

Implementations for fuzzy systems are made according to speed requirements of the problem, for the case of critical real time systems where response time is crucial in order to keep a safe system operation, hardware is the best choice to provide high processing speed.

One important issue on fuzzy systems is the final step, defuzzifier, when computed fuzzy data is turned into crisp data. There are different forms for defuzzifiers satisfying different precision and speed requirements, even though, Center Of Gravity method COG and Mean Of Maxima MOM are the most used on hardware.

COG is proved to be more accurate than MOM, but it requires much more calculations [7], [10]. On this paper a defuzzifier method called Center of Slice Area Average Defuzzifier COSAA [8] is implemented in hardware, it is based on  $\alpha$ -levels for membership functions representation and uses discrete numbers which make it fit better into digital computers. As results, it reduces execution time of Center Of Gravity COG, without loss of accuracy as Mean Of Maxima MOM and even with hardware resources reduction.

Paper is organized as follows: in order to implement efficient digital circuit is necessary to make some digital considerations as presented in Section II. Generalities for defuzzifiers are covered on Section III. Soft Core implementation for Center of Slice Area Average Defuzzifier is presented on Section IV. Results are shown in section V. Finally conclusions and future work are considered in Section VI.

## II. CONSIDERATIONS FOR DIGITAL IMPLEMENTATION OF FUZZY SYSTEMS

Real world systems use precise units that can be grouped into a crisp universe where all elements are well defined. In contrast, fuzzy values are grouped into membership functions that belong to a linguistic variable where limits are defined, but not the elements.

Commonly in fuzzy systems, there is a range of values for result of operations that belongs to  $[0, 1]$  representing truth values, where 0 means no membership, and 1 is complete membership. Another form to represent membership values is using integer numbers, this case is best suited for digital implementations because floating point operations consume much more time and resources. For this case, results are considered on  $[0, I]$  interval, where  $I$  is an integer represented by equation 1, where  $m$  is the number of bits used for resolution [8].

$$I = 2^m - 1 \quad (1)$$

#### A. Linguistic Variables

Let us consider  $A$  be a linguistic variable, the  $k$  number of membership functions to define  $A$  is given by equation 2 and is graphically expressed on Figure 1.

$$A = \{A_0, A_1, A_2, \dots, A_K\} \quad (2)$$

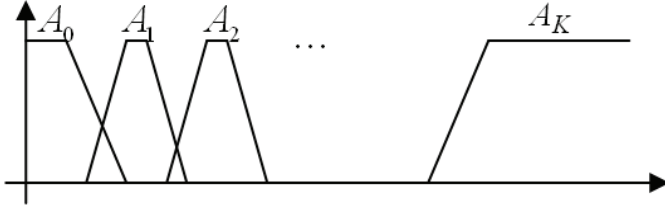


Figure 1. Membership functions for linguistic variable A

#### B. Membership Functions

To represent membership functions there are mainly two approaches, on the first, a membership function is represented through a membership value for each element on input universe, it means that for a better output resolution is necessary to use a higher bit number, so while more resolution is gained, more values must be computed independently of the membership value number that modify truth universe.

The second approach is to use  $\alpha$ -levels [9] or  $\alpha$ -cuts [10] for membership values representation, on this case, a membership function is decomposed on a set of vectors that contain for each membership value, the interval on which the function is true. This approach allows the use of any shape for membership functions and does not take care of input resolution.

Let  $M$  be a membership function, with  $x_i \in X; i = 0, 1, \dots, J$ ; where  $J$  is the number of points in  $x$  axis given by  $J=2^n-1$ ,  $n$  is the number of bits used for input space representation.

Each  $\alpha$ -level is defined by the following four parameters:

1.  $M$  is the name of the membership function;
2.  $i$  is the weight index, is at most equal to  $I$  and represents membership value in  $M$ ;
3.  $\alpha_0^i$  is the initial point for the interval with  $i$  as membership value.
4.  $\alpha_f^i$  is the final point for the interval with  $i$  as membership value.

Figure 2 shows the representation of a membership function  $M$  represented by  $I$  number of  $\alpha$ -levels. The whole  $M$  membership function is now defined by the set of all its  $\alpha$ -levels over input universe  $X$ , values for  $m$  and  $n$  depend on the number of bits used for each universe and can be different, better if  $m < n$ , usually 5 or 6 bits are suitable for membership

universe  $m$ , and while more bits are used for crisp universe is a better resolution.

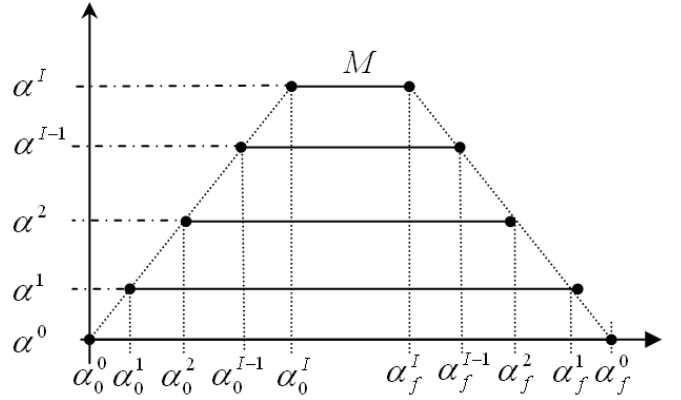


Figure 2. Discretization of a trapezoidal membership function into  $I$  number of  $\alpha$ -levels

#### C. Rule-base and Inference

The fuzzy system is dedicated to infer from fuzzy input data which is the most suitable fuzzy output it must deliver. Rule-base that contains knowledge acquired from an expert is expressed with rules in the form of Mamdani as on equation 3.

$$R_s: \text{IF } x \text{ is } A_i \text{ AND } y \text{ is } B_j \text{ THEN } z \text{ is } C_k \quad (3)$$

where  $x$  and  $y$  are input variables,  $z$  is output variable,  $A_i, B_j$  and  $C_k$  are linguistic values for variables  $A$  and  $B$  in input and  $C$  at output.

Membership degree values for  $x$  and  $y$  inputs are used to calculate antecedent part of rule, realizing an intersection with both input membership values, commonly this is computed by some t-norm operation as on equation 4.

$$\mu_{C_k}(z) = T(\mu_{A_i}(x), \mu_{B_j}(y)) \quad (4)$$

When all rules are evaluated, only those membership functions that contribute to final result are joined by the use of an or t-conorm as on equation 5, obtaining a unified membership function made of those output membership functions whose maximum membership value is greater than zero, this resultant function is called aggregated membership function

$$\mu_C(z) = S((\mu_{C_0}(z)), \dots, (\mu_{C_k}(z))) \quad (5)$$

An example for aggregated function is the one from Figure 3. There is a case for an  $\alpha$ -level based fuzzy system with two inputs with four membership functions each, and one output with five membership functions, all trapezoidal. When two inputs  $x=119$  and  $y=374$  are evaluated, there are four activated rules, when they are aggregated, they turn into a unified shape represented with 512 values for universe of discourse and 32 membership values. Functions 0 and 5 do not contribute to final result, on behalf, functions 1, 2 and 3 do, with 13, 25 and 9 as membership values respectively.

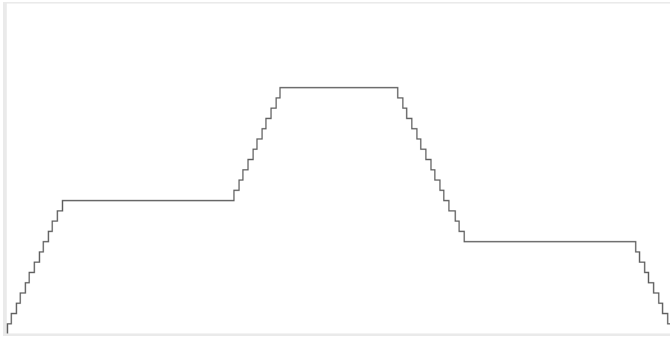


Figure 3. Aggregated membership function for  $x=119$  and  $y=374$ .

### III. DIGITAL DEFUZZIFIERS

Final step on a fuzzy system is defuzzifier; it calculates the final result out of aggregated membership function, by computing only one crisp value that best represent desired output. There are many defuzzifier methods satisfying different precision and speed requirements, traditional methods are Center of Gravity COG and Mean of Maxima MOM, others are presented as Plateau Average PA [11], and Trapezoid Median Average [12].

As it has been proved on literature that COG is the best defuzzifier, many authors use it as ideal for testing, obtaining deviations from their methods with respect to COG [7], [10]. The intention then is to obtain a defuzzifier as good as COG but that is more suitable for digital implementations avoiding excessive calculation time elapsed by COG.

#### A. Mean of Maxima Defuzzifier

For calculation of Mean of Maxima defuzzifier MOM (equation 6), it is only necessary to know which was the highest membership value on aggregated function and obtain the middle point of that level as result, it is very simple, that is why it is fast, but does not consider rest of function so changes here are like big increases or decreases for final output, being unstable, but is well suited for many applications.

$$z_{MOM} = \sum_{j=1}^l \frac{w_j}{l} \quad (6)$$

#### B. Center of Gravity Defuzzifier

Center Of Gravity defuzzifier COG realization requires calculating the whole area and determining where its weighted midpoint is. As it uses all elements from input universe, it requires  $r=2^n-1$  iterations according to number of bits used for input universe as on equation 7. This defuzzifier is preferred by most authors because of its accuracy and easiness for implementation.

$$z_{COG} = \frac{\sum_{i=1}^k x_i \times \mu(x_i)}{\sum_{i=1}^k \mu(x_i)} \quad (7)$$

#### C. Centre of Slice Area Average Defuzzifier

To work efficiently with a  $\alpha$ -level based aggregated membership function Centre of Slice Area Average defuzzifier COSAA is used. On this case it requires at most  $I$  iterations according to membership values resolution as on (1).

COSAA is defined by 8, where upper part calculates summation of midpoints for all  $\alpha$ -levels, to calculate midpoint of each level do not require complex operations, it uses only addition, subtraction and right shift to realize division by 2. the only required division is made at endpoint with  $\alpha_{MAX}$ . Operating description is graphically expressed on Figure 4 where  $I$  is equal to 4, i.e. two bits are used. Midpoints for each level are calculated and accumulated to finally obtain the average result indicated as COSAA.

$$z_{COSAA} = \frac{\sum_{i=0}^{\alpha_{max}} \left( \frac{(x_f^{\alpha_i} - x_0^{\alpha_i})}{2} + x_0^{\alpha_i} \right)}{\alpha_{max}} \quad (8)$$

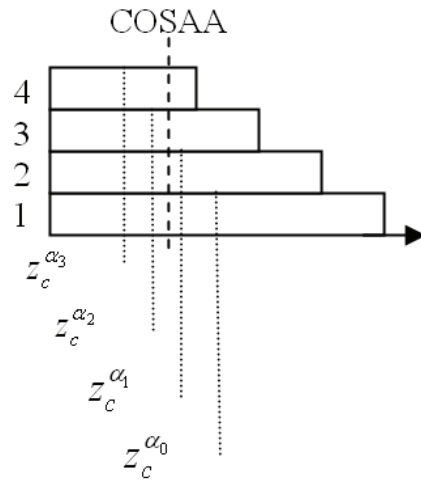


Figure 4. Center of Slice Area Average Defuzzifier operation.

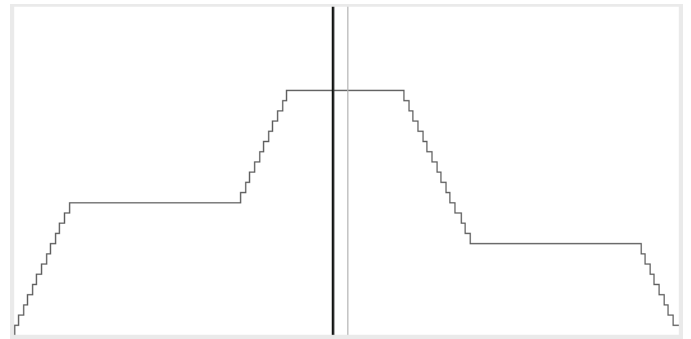


Figure 5. Defuzzifier results for inputs  $x=119$  and  $y=374$ , outputs are COG=243, MOM=255 and COSAA=244.

As example of defuzzifiers presented, there is a case for a  $\alpha$ -level based fuzzy system with two inputs with four

membership functions each, and one output with five membership functions, all trapezoidal, represented on universes of 512 values (9-bit) for universe of discourse and 32 membership values (5-bit). Inputs are  $x=119$  and  $y=374$ , output results are COG=243, MOM=255 and COSAA=244 as on Figure 5.

#### IV. HARDWARE IMPLEMENTATION OF COSAA

Centre of Slice Area Average Defuzzifier Hardware implementation is based on equation (8) and is presented on

Figure 8. Aggregated membership function resultant from previous stage is loaded into two ROM memories that contain the complete description of the function in terms of its  $\alpha$ -levels. Height index  $k$  for each  $\alpha$ -level is given by memory address, and the contents are the initial point  $\alpha_0^k$  on MEM\_32x8 block, and final point  $\alpha_f^k$  on MEM\_32x8\_FINAL. Five bits are used for membership resolution obtaining 32 levels, for that reason, memories are 32 deep, 8-bit wide.

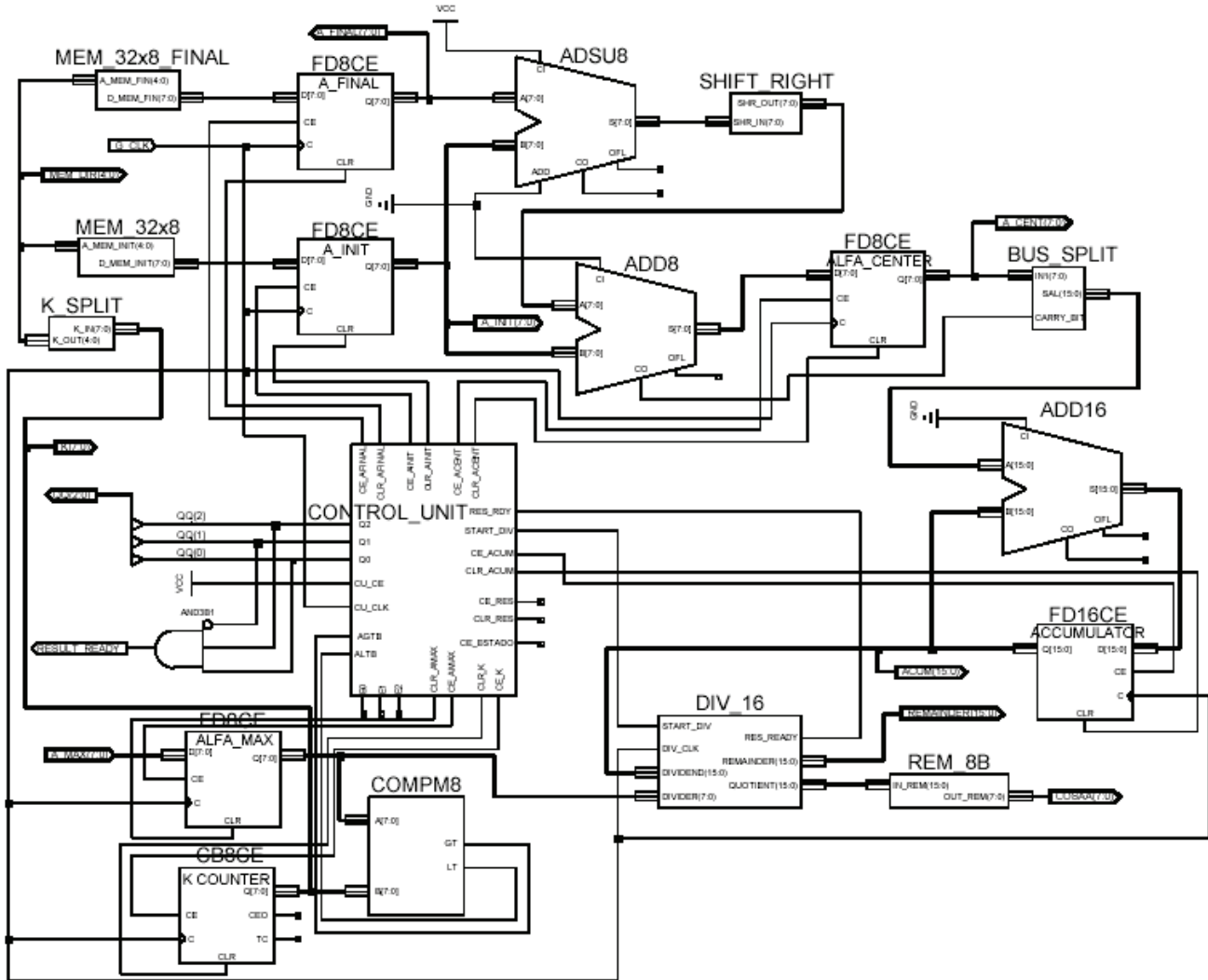


Figure 6. Hardware realization for Centre of Slice Area Average defuzzifier. There can be identified five 8-bit registers FD8CE that are D-type flip-flops and are used to store values of  $\alpha_0^k$  (A\_INIT),  $\alpha_f^k$  (A\_FINAL),  $\alpha_C^k$  (ALFA\_CENTER),  $\alpha_{MAX}$  (ALFA\_MAX) and Accumulator respectively labelled.

There can be identified four 8-bit registers (FD8CE) that are D-type flip-flops used to store values for  $\alpha_0^k$  (A\_INIT),  $\alpha_f^k$  (A\_FINAL),  $\alpha_C^k$  (ALFA\_CENTER),  $\alpha_{MAX}$  (ALFA\_MAX) and a 16 bit register for accumulator.

To calculate the center of a level as on upper part from Equation 8, ADSU8 is used as an 8-bit subtractor to realize

the  $\alpha_f^k - \alpha_0^k$  part, division by two is performed by SHIFT\_RIGHT unit; ADD8 realizes the summation of offset to obtain the center of evaluated level that will be now stored on its correspondent ALFA\_CENTER register.

BUS\_SPLIT is an 8 to 16 bit conversion block that drive  $\alpha_C^k$  calculated value as input for ADD16 that is a 16 bit

adder used to realize summation among calculated value and previous values stored on accumulator register.

An 8-bit counter CB8CE labeled K\_COUNTER, is used as pointer for each  $\alpha$ -level address on both memories. COMPM8 is a magnitude comparator used for iteration control procedure between  $\alpha_{MAX}$  and K. The procedure of calculate midpoints for all levels is repeated until *ALTB* signal from comparator goes high that is the moment when  $k$  is equal to  $\alpha_{MAX}$  and is the appropriated moment to start division. The completion of division algorithm requires about 50 clock cycles to obtain a 16-bit result that is converted to 8-bit value trough REM\_8B block. Result is available when RESULT\_READY signal goes high and remains there until next COSAA calculation overwrite final value.

## V. RESULTS

### A. Simulation Results

This methodology was first implemented on a C++ Builder simulator on which every stage was constructed according to methodology presented. Simulation is realized for a fuzzy system with two inputs with four membership functions each, one output with five membership functions and 16 rules at rule base, for an easy visualization of universe spaces they are discretized at 32  $\alpha$ -levels i.e. 5 bits for  $y$  axis (membership universe), and 256 values, i.e. 8 bits for  $x$  axis (input universe).

Table I presents some test cases realized for three defuzzifiers, there is shown the result and elapsed time.

### B. Soft-Core Implementation Results

Hardware for COSAA was constructed from logical gate design using Xilinx tools for FPGA design over a Spartan 3 XC3S400-5FG320 device that has 400,000 logic gates. Execution results for COSAA hardware are shown on Figure 7 and Figure 8 using the set of  $\alpha$ -levels that describes aggregated function.

$G\_CLK$  is the global clock signal for the system;  $QQ$  is the 3-bit state counter, it iterates from states 1 to 3 until all levels are evaluated, this is controlled by  $K$  counter until its value reaches  $\alpha_{MAX}$  ( $A\_MAX$ ) value as last level to evaluate.  $MEM\_DIR$  is the current address index of the memory that contains  $\alpha_0^k$  ( $A\_INIT$ ) and  $\alpha_f^k$  ( $A\_FINAL$ ) values, the calculation of  $\alpha_c^k$  is presented on  $A\_CENT$  and its summation is stored on  $ACUM$ .

TABLE I. SOME TESTS REALIZED FOR THE SYSTEM.

| x   | y   | Defuzz | Result | Time us |
|-----|-----|--------|--------|---------|
| 10  | 63  | COSAA  | 23     | 2.23    |
| 10  | 63  | COG    | 33     | 13.40   |
| 10  | 63  | MOM    | 9      | 1.95    |
| 10  | 157 | COSAA  | 65     | 2.23    |
| 10  | 157 | COG    | 70     | 22.346  |
| 10  | 157 | MOM    | 63     | 1.95    |
| 43  | 215 | COSAA  | 128    | 2.23    |
| 43  | 215 | COG    | 128    | 32.68   |
| 43  | 215 | MOM    | 127    | 1.95    |
| 75  | 229 | COSAA  | 176    | 2.79    |
| 75  | 229 | COG    | 160    | 49.34   |
| 75  | 229 | MOM    | 191    | 2.51    |
| 109 | 109 | COSAA  | 86     | 3.35    |
| 109 | 109 | COG    | 107    | 70.67   |
| 109 | 109 | MOM    | 63     | 2.69    |
| 146 | 117 | COSAA  | 132    | 3.07    |
| 146 | 117 | COG    | 132    | 71.79   |
| 146 | 117 | MOM    | 127    | 3.07    |
| 216 | 216 | COSAA  | 214    | 2.23    |
| 216 | 216 | COG    | 204    | 18.15   |
| 216 | 216 | MOM    | 245    | 2.23    |

When all levels are calculated system goes to state 4 where it starts the final division between  $ACUM$  and  $A\_MAX$  and wait there until division completion that is the moment to go to state 5 where result ready signal (*RESULT\_READY*) is activated indicating that final result can be read from *COSAA* as integer and it will be available until next division is started on state 4.

On Table II there are resources consumed by final implementation of COSAA defuzzifier, it can be noticed that it uses few circuitry and delay times are considered for a combinatory array. Calculation of COSAA from an aggregated membership function requires 129 clock cycles to provide final result, considering that division uses almost 50 clock cycles, then it takes about 80 clock cycles to calculate the summation of all levels. To make this function more efficient it maybe must consider another type of division that is faster, but maybe hardware complexity increases.

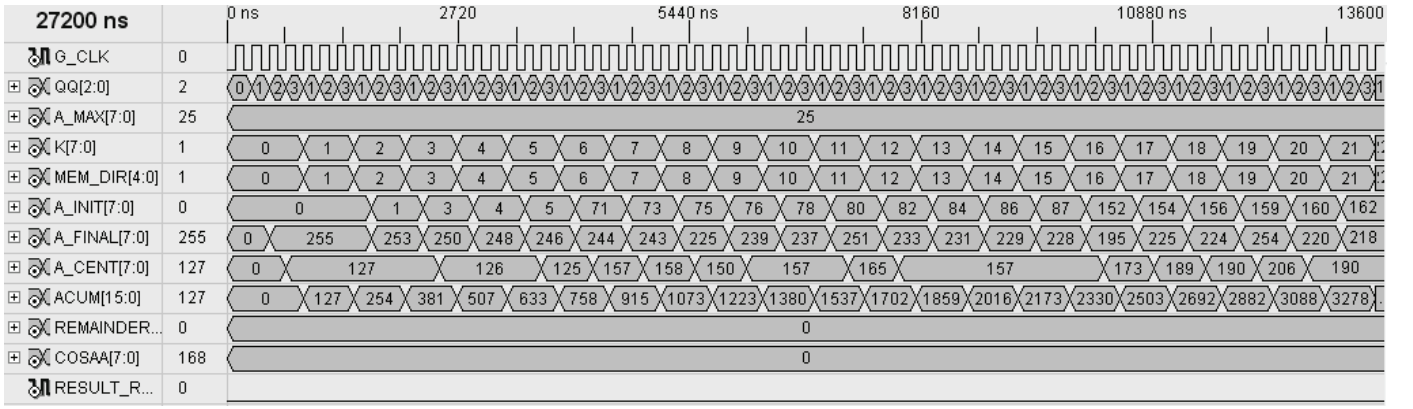


Figure 7. Simulation results for COSAA hardware implementation, the calculation of each level stored on memory is realized using  $K$  as index pointer until it is equal to  $A\_MAX$  on state 4 where it starts to calculate final division for average. Final result is presented on state 5 where  $RESULT\_READY$  signal is activated.

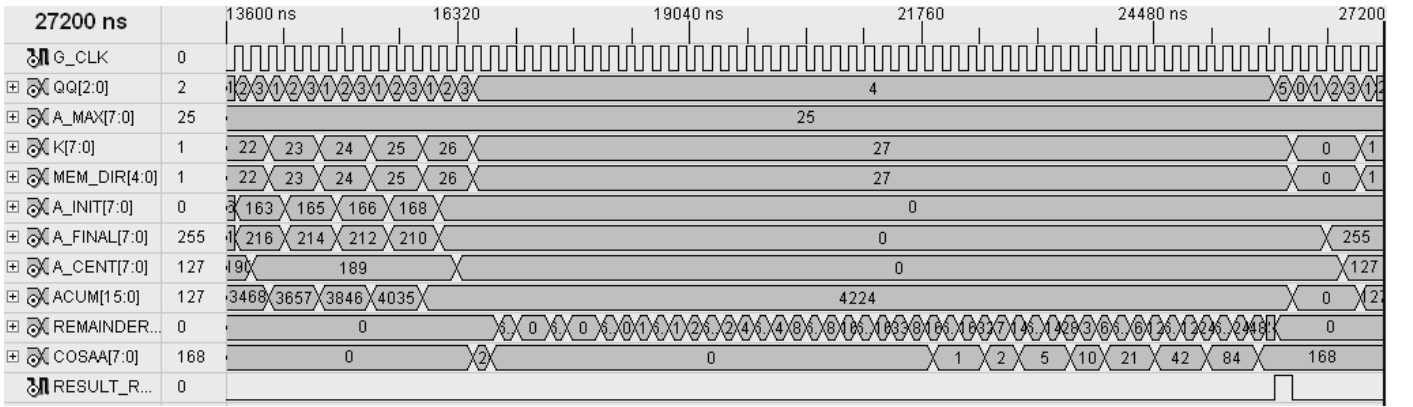


Figure 8. Simulation results for COSAA hardware implementation (Continued from Figure 17)

TABLE II. HARDWARE RESOURCES CONSUMED BY IMPLEMENTATION OF COSAA DEFUZZIFIER. TOTAL DELAY IS CONSIDERED JUST FOR A COMBINATORY CIRCUIT THAT DO NOT ITERATE.

| Operation | Logic Gates | Logic Levels | I/O used | Total Delay us | Delay Logic | Delay Route |
|-----------|-------------|--------------|----------|----------------|-------------|-------------|
| COSAA     | 972         | 14           | 124      | 26             | 41%         | 59%         |

## VI. CONCLUSIONS

Center of Slice Area Average (COSAA) Defuzzification reduces considerably calculation time consumed by Center of Gravity (COG) that uses at least  $2^n - 1$  iterations. The new method proposed has a maximum iteration number equal to the  $I$  number of  $\alpha$ -levels and is proportional to the number of bits used to quantify membership space, being COSAA as fast as MOM.

About precision, COSAA competes with COG and is much better than MOM, versatility can be added by use of different aggregation methods.

The  $\alpha$ -level method used for representing fuzzy systems with integer numbers, can be implemented easily over any binary computer system using simple operations such as

additions and shifting, avoiding time consuming operations such as multiplications and divisions obtaining a reduction on calculation time, and a more flexible and efficient approach for fuzzy systems realization.

## ACKNOWLEDGMENT

Authors would like to thank ICYTDF project 243/2010 and SIP-IPN Project 200110465 for their support.

## REFERENCES

- [1] M. Sugeno, K. Murakami. "An experimental study of fuzzy parking control using a model car". In Industrial Applications of Fuzzy Control, North-Holland, Amsterdam. 125-138, 1985.

- [2] J. A. Bernard. "Use of rule-based system for process control". IEEE Control Systems Magazine, Vol. 8, No. 5. 3-13, 1988.
- [3] E. H. Ruspini. "Reactive Fuzzy Control of Autonomous Robots". Industrial Applications of Fuzzy technology in the world, Hirota, Sugeno Eds., World Scientific. 1995.
- [4] R. Lovassy, L. Koczy, L. Gal, "Function Approximation Capability of a Novel Fuzzy Flip-Flop Based Neural Network," Proc. of the International Joint Conference on Neural Networks, pp. 1900-1907, 2009.
- [5] D. Falchieri, A. Gabrielli, E. Gandolfi, M. Masetti. "Design of very high speed cmos fuzzy processors for applications in high energy physics experiments". Seventh International Conference on Microelectronics for Neural, Fuzzy, and Bio-Inspired Systems. 284-291, 1999
- [6] C. Tudore, S. Bumbaru, L. Dimitriu, "Relative Aggregation Operator in Database Fuzzy Querying,"The Annals of "Dunarea de Jos" Unniversity of Galati, Fascicle III, pp.39-44, 2005.
- [7] T. Runkler, "Selection of Appropriate Defuzzification Methods Using Application Specific Properties". IEEE Transactions on Fuzzy Systems, Vol. 5, Issue 1. 72-79, 1997.
- [8] A. Hernández Zavala, O. Camacho Nieto, I. Batyrshin, "Center of Slice Area Average Defuzzifier for Digital Implementations of Fuzzy Systems," Proc. of International Conference on Artificial Intelligence, pp. 97-102, 2010.
- [9] K. Uehara, M. Fujise. "Fuzzy inference based on families of  $\alpha$ -level Sets". IEEE Transactions on Fuzzy Systems, Vol. 1, No. 2. 111-124, 1993.
- [10] G. Bojadziev, M. Bojadziev. "Fuzzy Sets, Fuzzy Logic, Applications", Advances in Fuzzy Systems – Applications and Theory Vol. 5, World Scientific Publishers. 1995.
- [11] S.S. Lancaster, M.J. Wierman, "Empirical study of defuzzification", Conference of the North American Fuzzy Information Processing Society. 121- 126, 2003.
- [12] A. Banaiyan, S. M. Fakhraie, H. R. Mahdiani. "Cost-performance co-analysis in VLSI implementation of existing and new defuzzification methods". International Conference on Computational Intelligence for Modelling, Control and Automation Vol. 1. 828 – 833. 2005.