www.czasopisma.pan.pl PAN www.journals.pan.pl
INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2010, VOL. 56, NO. 4, PP. 405–410
Manuscript received October 21, 2010; revised November 2010. DOI: 10.2478/v10177-010-0054-8

Fuzzy Processing Implementation in Dedicated Digital Hardware

Przemysław M. Szecówka and Adam Musiał

***Abstract*—The paper presents a concept of digital circuit dedicated for fuzzy processing with numerical inputs and outputs. Partially concurrent and pipelined data flow provides high performance, with relatively low dependence on particular algorithm complexity. Sample design with triangular fuzzy sets, rule strength calculation ($minimum$ approach) and defuzzyfication by weighted sum of fuzzy sets centers was implemented in VHDL, verified and synthesized for FPGA. Floating point arithmetic was applied, including dvision performed by dedicated synchronous machine. All modules were prepared for easy reuse/redesign.**

***Keywords*—Fuzzy, hardware, floating point, VHDL, FPGA.**

I. Introduction

FUZZY processing, since its introduction in 60's [1], gained prestigious position in research and industry. Huge record of successful applications in data processing, identification, and control placed this technology in a position competitive to neural networks. Several scientific journals are devoted strictly to this technology. Computational complexity of classic fuzzy processor may be classified among moderate ones. It strongly depends on number of inputs, number of fuzzy sets, number of rules and number of outputs. For software implementation this feature means that higher complexity induces longer processing time. From computational point of view however, several parts of fuzzy processing may be performed concurrently. Thus it shall be expected that when implemented in dedicated digital hardware, fuzzy processing may remain very fast, regardless of complexity. These features invoked an interest in development of digital circuits performing various kinds of fuzzy processing. The first papers, usually theoretical, appeared in 90's [2]–[5]. Simultaneously mixed signal devices performing computation in analog way and programmable in digital way were used for research [6]. Growing complexity of programmable logic devices encouraged extensive practical use of digital architectures [7], [8], sometimes leading to modification of the algorithms towards easier implementation in hardware [9]. This paper presents another variant of digital architecture dedicated to fuzzy processor with numerical inputs and outputs. Specific features of the concept are full floating point arithmetic approach, data flow organization with optional pipelining and reuse/redesign capabilities. A sample design was implemented in VHDL, verified and synthesized for Xilinx FPGA.

This work was supported by the grant: Detectors and sensors for measuring factors hazardous to environment – modeling and monitoring of threats, POIG.01.03.01-02-002/08.

P. M. Szecówka is with the Faculty of Microsystem Electronics and Photonics, Wrocław Univeristy of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland (e-mail: przemyslaw.szecowka@pwr.wroc.pl).

A. Musiał is with Clarena, Kleczkowska 45, 50-227 Wrocław, Poland (e-mail: adam.musial@clarena.pl).

II. Fuzzy Processing Overview

Fuzzy processing is based on rules constructed for fuzzy sets. Fuzzy rules are formulated in a way quite similar to classic logic. For input vector $x = [x_1, x_2, .., x_n]^T$ the output y is described by the series of rules like:

$$IF\ x_1 \in F_1^{(l)},\ x_2 \in F_2^{(l)}, ...,\ x_n \in F_n^{(l)}\ THEN\ y \in G^{(l)}$$

where $l = 1...L$. The difference is in fuzzy sets $F_1...F_n$ and G applied instead of classic sets. Each fuzzy set is defined by a function determining to what extent particular element belongs to the set. These functions return values between 0 and 1, with boundary values compatible with classic sets theory and intermediate values pointing to growing connections between an argument and a set. For most of practical applications, with numeric input and numeric output, there are 3 steps of fuzzy processing: fuzzyfication, calculation of rules strength and defuzzyfication (Fig. 1).

Fuzzyfication means calculation of fuzzy set functions for each element of numeric input vector x. In this design triangular functions were applied for input (Fig. 2) and singletons for output.

Rules strength calculation provides a number $F^{(l)}$, which says how far particular input vector matches each rule. In this design the $minimum$ approach (1) was applied. This one is not the best solution. It may be shown that e.g. product of fuzzy sets functions calculated for each element of input vector, provides more accurate output. The $minimum$, however is very easy to implement and does not consume much resources.

$$F^{(l)} = min(F_1^{(l)}(x_1), (F_2^{(l)}(x_2), ..., F_n^{(l)}(x_n)) \quad (1)$$

$$y = \frac{\sum_{l=1}^{L} F^{(l)} c_l}{\sum_{l=1}^{L} F^{(l)}} \quad (2)$$

The final stage is defuzzyfication, where the numeric value of output is calculated, according to what the rules say. Each rule points to some fuzzy set $G^{(l)}$, with some central value c_l. Weighted sum of fuzzy sets centers (2) was applied. This one provides very accurate results. It is rather expensive in the context of hardware resources and contains division which is hard to implement, but it is also hard to find anything more simple providing reasonable accuracy. Thus the authors decided to invest some effort in this solution.

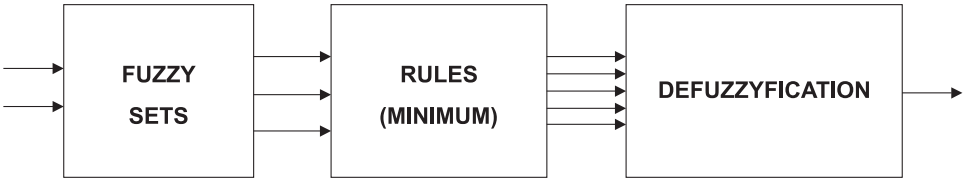

Fig. 1. Three stages of fuzzy processing

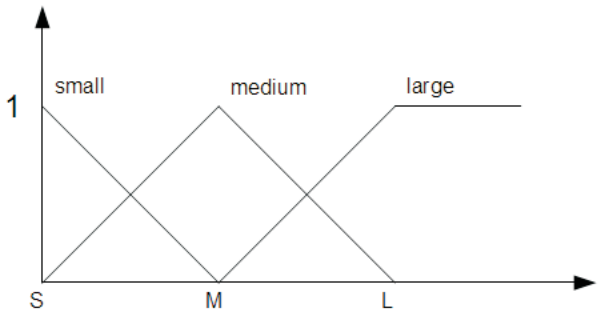

Fig. 2. Fuzzy sets definition

III. Architecture

The architecture is full-synchronous with single clock. The main components roughly match elements of computation described in previous section. For the sample design it was presumed that there are 2 inputs, 1 output and 3 fuzzy sets for each input/output, what makes 6 rules. All numbers are floating point. The other inputs are global clock and reset signals, and a strobe – external pulse starting calculation. Data flow is pipelined and some operations are performed concurrently.

A. Fuzzy Sets

The architecture of fuzzy sets block is presented in Fig. 3. This block provides calculation of 3 functions describing 3 sets – SMALL, MEDIMUM and LARGE. The shape of MEDIUM set function is triangular, with 2 symmetric slopes. The SMALL and LARGE functions have single slopes, falling and rising respectively. Any classic configuration of 3 sets may be defined by 3 characteristic values S, M, L defined in the input domain and the slope tg. General architecture remains same. Eventually the design may be easily reused and replicated for various configurations of 3 fuzzy sets. Calculation of each fuzzy set function involves muxes with comparators, subtractors, multipliers. Chains of combinatorial logic are relatively long. Series of registers may be inserted between consecutive stages to provide pipelined operation and higher clock speed.

B. Rules Strength Calculation

After calculation of fuzzy sets for all inputs, the rules strength is estimated. For the selected method ($minimum$), it means that the strength of each rule is equal to the lowest value of fuzzy set appearing in it (1). The search for minimum

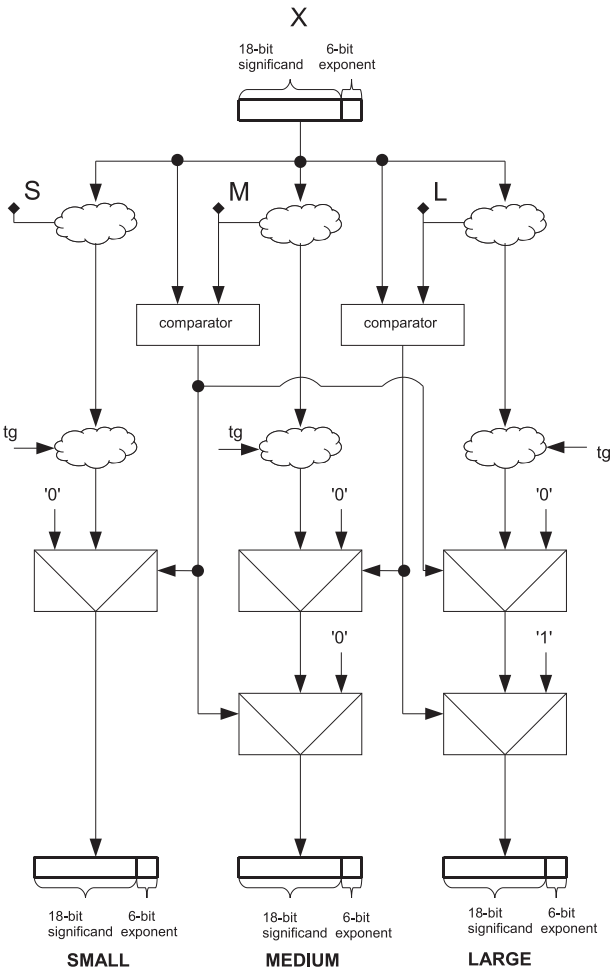

Fig. 3. Fuzzy sets calculation block.

was implemented in tournament type architecture with a single comparator and register preserving the actual master. In consecutive clock cycles the consecutive candidates are delivered by a mux, controlled by a counter. In a comparator the candidate fights with the current master. If it the new candidate is lower, it is registered and becomes the new master. If the new candidate is higher, the previous master remains in register. After checking all the fuzzy set values, the winner is sent to another register and then to defuzzyfication block. The whole process takes the number of clock cycles equal to the dimension of input vector. This hardware however is replicated for each rule, as it is shown in central part of schematic in Fig. 4. Thus all rules are processed concurrently.

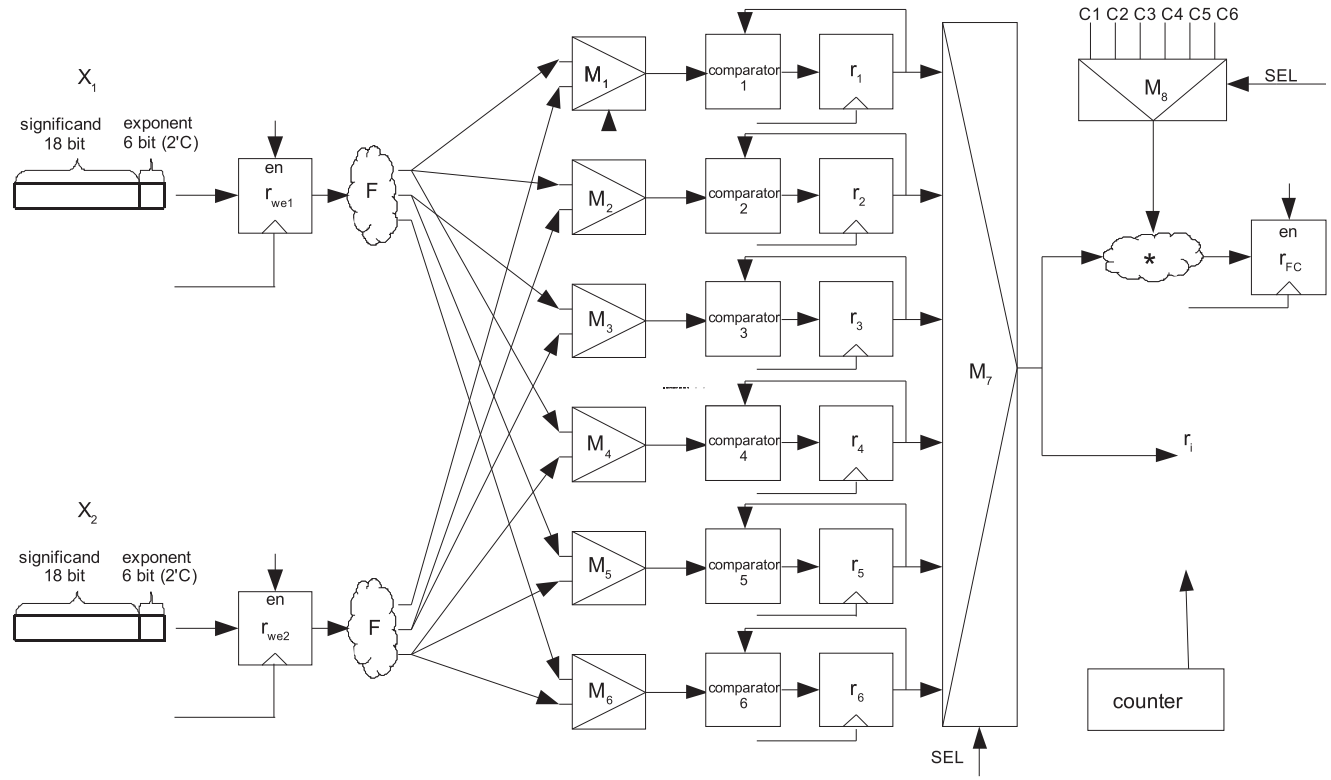

Fig. 4. Fuzzy sets and rules strength calculation.

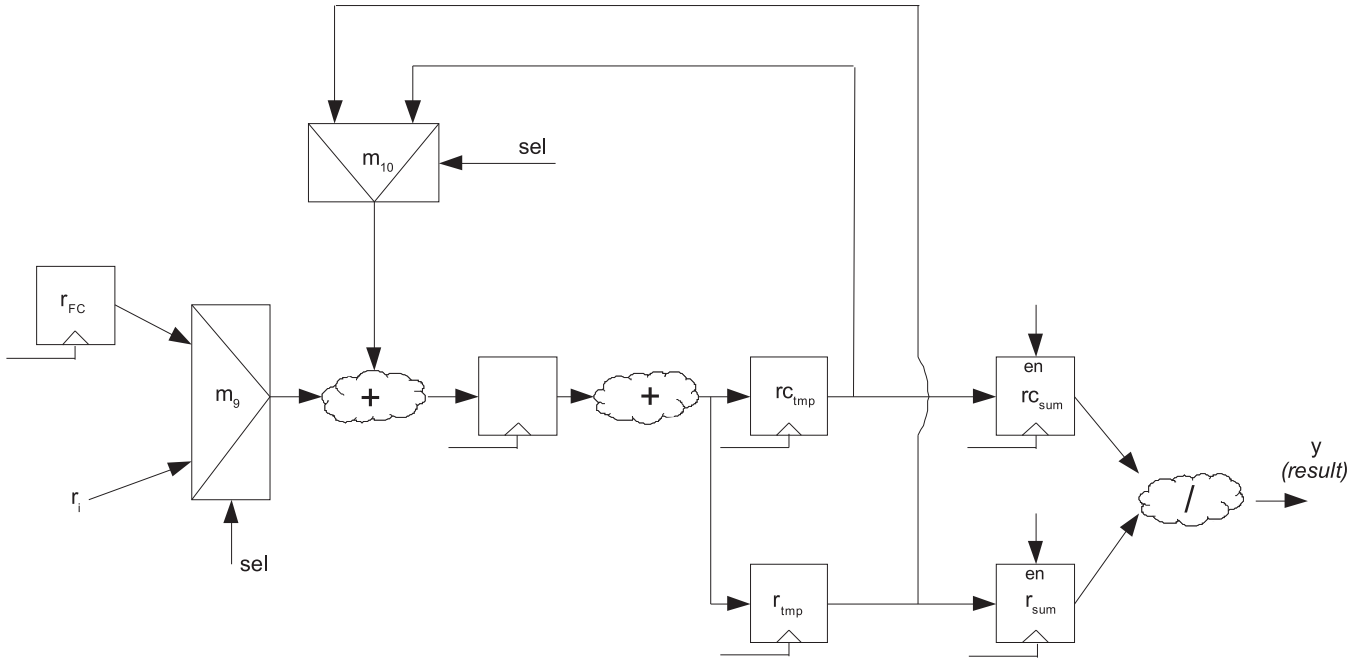

Fig. 5. Defuzzyfication block.

C. *Defuzzyfication*

The final stage of fuzzy processing is defuzzyfication. For the weighted sum of centers of fuzzy sets defined for the output (2), it consists of a multiplier, two accumulations and division block. The first accumulator adds together fuzzy set centers multiplied by the appropriate rule strength factors (numerator in (2)). Multiplication blocks are shown in the right part of Fig. 4. In the consecutive clock cycles the rule strength factors are selected from a mux and directed to multiplication. Simultaneously another mux selects the appropriate fuzzy set center for multiplication. The result of multiplication is stored in register and then sent to accumulator shown in Fig. 5. The

key functionality of this block is accumulation of incoming numbers. However due to the specific properties of floating point arithmetic blocks, somewhat unusual architecture was proposed. Floating point multiplication block is almost twice faster than floating point adder. Thus the adder may be divided to two stages, separated by a register. In such case ca. twice higher clock frequency may be applied. This additional register however, destroys the natural synchronicity of traditional accumulator. When it takes two clock cycles to add a new element, then the consecutive elements shall be delivered every two clock cycles too. The idle clock cycle would have to be inserted making this idea questionable. This idle cycle however may be used to perform another computation. In this case there is a need to calculate a simple sum of rule strength factors (denominator in (2)). Thus the single adder may be shared by the two accumulators. A common blinking signal controls two muxes switching the channels connected to the adder. Same blink line enables and disables the synchronous latches, which store the cumulated values. The final stage of defuzzyfication is division. This block is definitely the most complicated part of the design. The key part is synchronous machine providing fixed point division, presented in Fig. 6. In consecutive stages it compares series of bits cut from the numerator with denominator bits. Depending on comparison result, logic 0 or 1 is sent to a shift register containing the result of division. Simultaneously either the original bits of numerator or result of subtraction are sent back to the appropriate part of numerator register. In the next clock cycle the modified (or not) numerator is shifted left and the new comparison/subtrac. For proper operation the most significant bit of denominator must not be 0. Thus in the very begin the denominator shall be shifted accordingly. Special counter memorizes the number of shifted bits and at the final stage the result is shifted too. Various other exceptions may occur and must be handled by additional logic. The whole process takes a number of clock cycles slightly exceeding the number of bits in the output value, i.e. it depends on precision required. Floating point division block relies on this synchronous machine to calculate the significand. Another part of logic delivers exponent and provides the standardized shape of output.

D. Floating Point Arithmetic

All calculations are performed for the floating-point numbers with 25-bit representation. The vector consists of sign bit, 18-bit significand and 6-bit exponent. Octal basis was applied, i.e. there are 6 digits in significand. Sign-module coding of significand was found the most effective for arithmetic and logic operations, and in several points of fuzzy processing the result is never negative, thus the sign bit may be removed. On the other hand the exponent is represented by 2-complement notation. Arithmetic modules were reused/redesigned from other in-house developed architectures [10]–[12]. Floating point division block was based on existing, fully verified, sequential fixed point divider [13], combined with specific logic added. The basic fixed-point arithmetic operations were taken from the IEEE std_logic_signed and std_logic_unsigned packages.

E. Timing and Synchronization

The circuit starts new calculation with external strobe signal. The input signals are registered. Calculation of fuzzy sets functions takes single clock cycle and is performed concurrently for all inputs and sets. Finding a minimum for each rule takes the number of clock cycles equal to the number of input vector elements. For the sample design it is 2. Simultaneous calculation of numerator and delimiter of the defuzzyfication fraction takes a number of clock cycles twice higher than the number of rules plus 1 more cycle to register the final value. In this case it makes 13. The final division is performed in specific synchronous machine. It consumes 18 clock cycles for calculation of 18-bit significand and 3 cycles for other logic operations. This makes 21 clock cycles. Thus there are 3 steps of calculation, which take 37 clock cycles. It is possible however to organize a pipelined data flow. For the presented example the slice was set to 21 clock cycles, determined by the division block. The first two stages may be combined to a single step of pipeline, with 16 active and 5 idle cycles. In such case the whole calculation takes 42 clock cycles and every 21 clock cycles the new output is delivered. For the other configurations of fuzzy system this setup may be modified. For higher complexity (number of inputs and rules) superiority of pipelined solution for massive throughput is more visible.

IV. Implementation

The design was implemented in VHDL [14], with all logic signals and operations based on the std_logic_1164 package from the IEEE library. A little code partitioning was applied. All floating-point arithmetic operations were kept in separate entities. Another specific entity was fuzzy sets block, replicated for each input variable, with specific constants. All the other functionality – data flow control with counters and muxes was placed in a single, top-level module. VHDL code was processed, simulated and synthesized for FPGA using Xilinx ISE tools [15]. Verification plan covered individual tests of each module and comprehensive simulation of the whole fuzzy processor. The most exhaustive tests were applied for arithmetic modules. In particular the fixed-point division block was verified for all possible combinations of input bit vectors. As a target device for synthesis process, Xilinx XC3S1000 circuit was selected – Spartan 3 series FPGA with complexity estimated to 1 million of equivalent gates. Synthesis allocated 60% of available logic, 743 registers (4%) and seven 18-bit multiplication blocks (29%). Shall be stated that floating point division module consumes significant part of resources, i.e. there is enough space for extended functionality. Maximum clock frequency, according to static timing analysis is above 30 MHz. Eventually for the sample design the processing speed may be estimated to 1.4 million of samples per second.

V. Discussion

Main advantage of dedicated digital circuit, when compared with the software implementation of typical computational algorithm is the higher processing speed obtained for similar or smaller complexity of electronics involved. Digital hardware may be organized in several ways, depending on features of

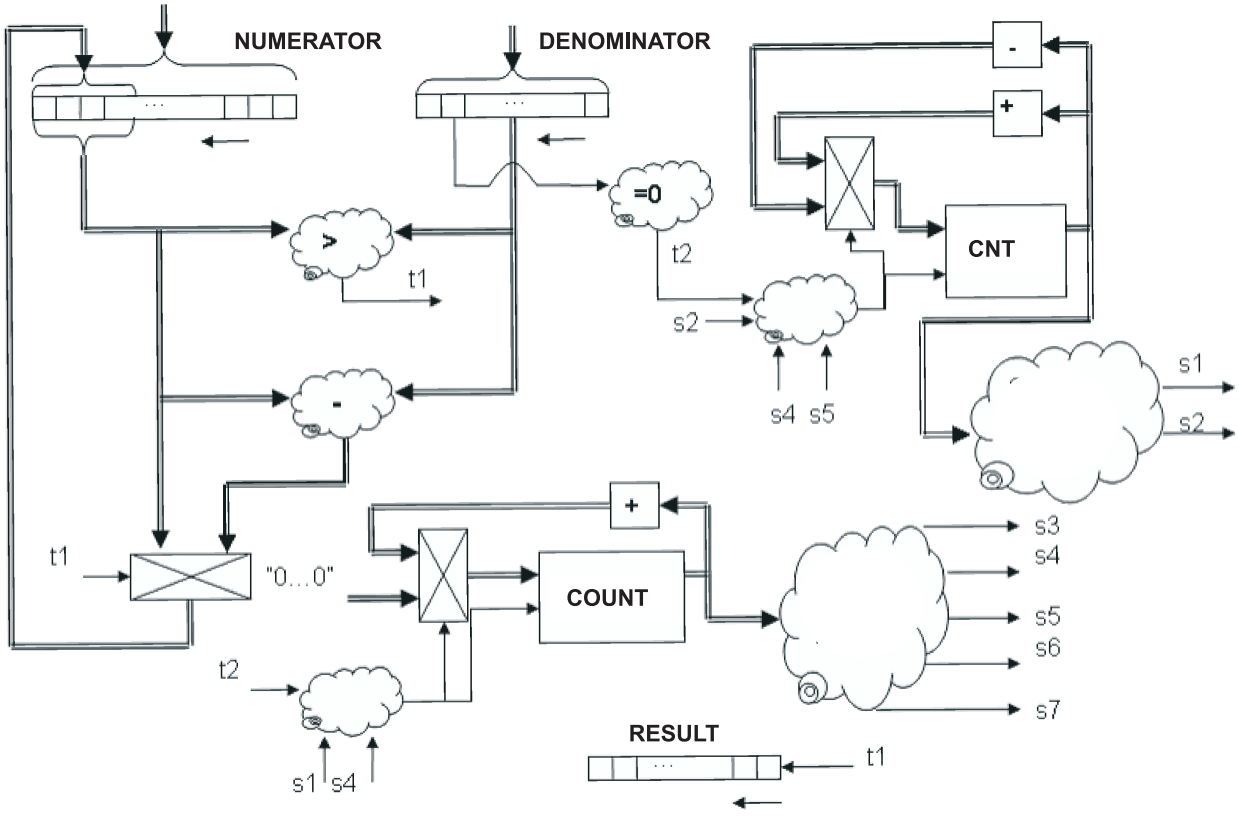

Fig. 6. Synchronous machine for division.

the algorithm and the actual need for processing speed. If the problem may be decomposed to a series of independent calculations, which may be executed in parallel, the dedicated pieces of hardware may be designed and allotted to each of them. This approach provides the ultimate speed of processing, sometimes independent from problem dimension or computational complexity. Other solution is required when operations are not independent, i.e. some of them must be executed one after another. Parallel processing is not possible, however the appropriate set of processing units may be designed and connected serially, to form a pipeline. For the massive stream of data to process, the speed is equal to parallel approach. Common disadvantage of these two concurrent architectures is a cost of implementation, growing with the complexity of calculations. Sometimes the high performance is not affordable and/or not necessary, encouraging selection of some low-cost variant. If the same operation shall be performed on all elements of the input vector or the intermediate results shall be processed again in similar manner, a single processing unit may be reused for a set of operations. This kind of architecture is somewhat similar to microprocessor approach. But the processing unit, optimized for the required operations, together with simultaneous access to data stored in unlimited number of registers allows such low-cost solution to outperform the software approach.

The three kinds of a hardware approach presented above shall be perceived as simplified, canonic solutions. The real world applications, including this design, are usually some combinations of all of them. Calculation of fuzzy sets is a good example of parallel solution. These operations are independent and very short. It is possible however to redesign this part to a single unit with two queued inputs – one for consecutive elements of input vector and another for parameters of fuzzy sets defined for particular inputs. Another example of parallel operation is rule strength calculation block, replicated for each rule. Simultaneously the internal part of this block itself operates in slow, sequential manner, handling elements of input vector one after another. It is possible to redesign this part to a tree of comparators, working substantially faster. The last stage of computation is division module, which is a good example of architecture, which is naturally sequential and hard to redesign for parallel operation. Pipelined approach is used for the whole design. It was divided to two parts performing two steps of pipeline. These two steps must be executed one after another and simultaneously they have reasonably balanced time of execution. The bottleneck for processing speed is division module. For the presented design and for other similar configurations of fuzzy system to be implemented, the division consumes more than 50% of overall time required for processing of single data. Thus the division block alone shall occupy its own stage of pipelining and there is absolutely no need to speed-up e.g. the rule strength calculation blocks. It may be considered however to design a single block handling all the rules one afer another and/or a single fuzzy set block for

all inputs. For some combinations of fuzzy system complexity these changes would bring some savings on logic resources without any loss on processing speed.

VI. Conclusions

An architecture of logic circuit dedicated to fuzzy processing was proposed. Sample design was implemented in VHDL, verified and synthesized for FPGA. The code was prepared in a way providing easy and fast redesign for any variant of fuzzy processor with numeric input and output. High allocation of FPGA resources was observed. Shall be stated however that most of them were allotted for division module working on floating point numbers. This module is singular for single output of fuzzy system, regardless of its complexity. In-house developed floating point arithmetic was applied. This approach remains questionable as usually. Perhaps it is not necessary, but it simplifies migration of typical fuzzy algorithms from software to hardware, regardless of particular details, and eliminates problems with range of intermediate values. Thus it was shown that in spite of floating point arithmetic applied fuzzy processing may be physically implemented in contemporary 1M gates FPGA, leaving some space for extension or other tasks. Regardless of arithmetic applied – floating or fixed point, the synchronous machine for division remains notable element of this design.

The clock speed of 30 MHz, revealed by static timing analysis is not very high but quite reasonable for this kind of design and for very low number of clock cycles consumed for processing a single input. Processing speed – 1.4 M samples per second is competitive to personal computers and DSPs. At the current stage, clock frequency may be increased by redesign of modules calculating fuzzy set functions – applying more registers leading to elimination of the idle cycles in pipelined mode. Another option for increasing speed is redesign of arithmetic modules to fixed point.

References

[1] L. A. Zadeh, "Fuzzy sets," *Information and Control*, no. 8, 1965.
[2] G. Ascia, V. Catania, and M. Russo, "Vlsi hardware architecture for complex fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 553–570, 1999.
[3] D. L. Hung, "Dedicated digital fuzzy hardware," *IEEE Micro*, vol. 4, pp. 31–39, 1959.
[4] A. Jaramillo-Botero and Y. Miyake, "A high speed parallel architecture for fuzzy inference and fuzzy control of multiple processes," in *Proc. Third IEEE Conference on Fuzzy Systems*, vol. 3, Orlando FL, 1994, pp. 1765–1770.
[5] J. I. M. Torre, J. P. Deschamps, and M. F. Centeno, "Synthesis tools for dedicated fuzzy hardware," in *Proc. Fifth IEEE International Conference on Fuzzy Systems*, vol. 1, 1996, pp. 570–574.
[6] G. Louverdis and I. Andreadis, "Design and implementation of a fuzzy hardware structure for morphological color image processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 277–288, 2003.
[7] N. Masmoudi, M. Hachicha, and L. Kamoun, "Hardware design of programmable fuzzy controller on fpga," in *Proc. Fuzzy Systems Conference (FUZZ-IEEE '99)*, vol. 3, 1999, pp. 1675–1679.
[8] S. Sanchez-Solano, A. J. Cabrera, I. Baturone, F. J. Moreno-Velo, and M. Brox, "Fpga implementation of embedded fuzzy controllers for robotic applications," *IEEE Trans. Ind. Electron.*, vol. 54, pp. 1937–1945, 2007.
[9] S. Cai, X. Chen, Q. Wang, and M. Yin, "Fpga implementation of generalized fuzzy operations," in *Proc. Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, Shandong, 2008, pp. 560–564.
[10] J. Góra, P. M. Szecówka, and A. R. Wołczowski, "Fft based emg signals analysis on fpgas for dexterous hand prosthesis control," in *Man-machine interactions*, K. A. Cyran *et al.*, Ed. Springer, 2009, pp. 655–662.
[11] P. M. Szecówka, G. Sługocki, and T. P. Gotszalk, "Fast and high resolution frequency meter based on fpga," *Elektronika*, vol. 10, pp. 16–19, 2009.
[12] A. R. Wołczowski, P. M. Szecówka, K. Krysztoforski, and M. Kowalski, "Hardware approach to the artificial hand control algorithm realization," *Lecture Notes in Computer Science*, vol. 3745, pp. 149–160, 2005.
[13] P. M. Szecówka, A. Charytoniuk, and R. Najbert, "Automated implementation of feedforward neural network in digital integrated circuit," in *Proc. 10th international conference MIXDES*, 2003, pp. 164–167.
[14] "VHDL Language Reference Manual," IEEE Std No. 1076, 2000.
[15] "ISE Web Pack," Xlinx, 2009.