VHDL Modelling of a Fuzzy Co-processor Architecture

Radoslav Raychev°, Abdellatif Mtibaa**, Mohamed Abid***

Abstract *- The main benefit of hardware fuzzy processors over software implementations is their higher performance. This advantage is greatly appreciated in real-time and embedded applications. The paper presents a VHDL modelling approach for synthesis of a possible architecture for fuzzy co-processors. The target structural design is predefined for fuzzy calculus acceleration. An ordinary external RAM has to be attached to the fuzzy co-processor and to the host as a dedicated fuzzy inference rules and fuzzy database medium. So, the overall system architecture is quite simple and cost saving. This architecture has been implemented on a FPGA design plat-form. It makes confident the using of the fuzzy co-processor in real-time and/or embedded applications.*

Keywords: *fuzzy co-processor, architecture, VHDL, FPGA*

I. INTRODUCTION

Fuzzy co-processor (FzCoP) hardware implementations are a preferred option in embedded control systems [7]. Figure 1 presents data flow graph in a fuzzy-controlled system.

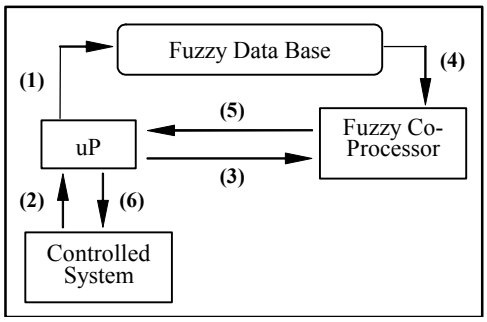

Figure 1. Data flow for embedded fuzzy co-processor

(1) The host uP makes an off-line download of fuzzy inference rules into fuzzy database, taking into account it's well defined structure [3, 9]; then it enter the in-line command loop, as follows:
(2) Reading input data from the controlled system;
(3) Passing this data from the host to the FzCoP;
(4) Finding out the result by the FzCoP taking into account the application specific fuzzy inference rules;
(5) Returning the result to the host;
(6) Using the result for outputting commands to the controlled system.

The following sections present a VHDL [10] description of such architecture.

II. ARCHITECTURE OF FUZZY CO-PROCESSOR

The target design aims a hardware accelerator for fuzzy calculations (see Figure 2). The FzCoP is connected, via the External Interface, to the uP, on the one hand and to the dedicated RAM, on the other. This RAM contains the application-specific fuzzy inference rules database (FzIfr-DB RAM). The rules data structure is well defined and the FzCoP takes it into account during fuzzy calculations. The VHDL modelling keeps trace of it too.

The behavioral description deals primarily with the FzCoP comportment. According to the Figure 3, two functional units compose the FzCoP: the Fuzzy Treatments Unit (FzTrtmU) and the RAM bus arbiter (BArb). The latter is dedicated for host to RAM and FzCoP to RAM bus time multiplexing. The former is a fuzzy calculus processor. The whole architecture is modeled as two VHDL processes, rather than in sequentially manner.

The FzCoP to FzIfr-DB RAM and the host to FzIfr-DB RAM communication modelling are based on the data flow intensity. Taking in mind the asymmetric data access stream between the FzIfr-DB RAM and two processors, the higher priority of the RAM access is assigned to the FzCoP in the design. The host to RAM interface is based on the "wait-for-bus-release"concept, while the FzCoP to RAM interface is serviced on time.

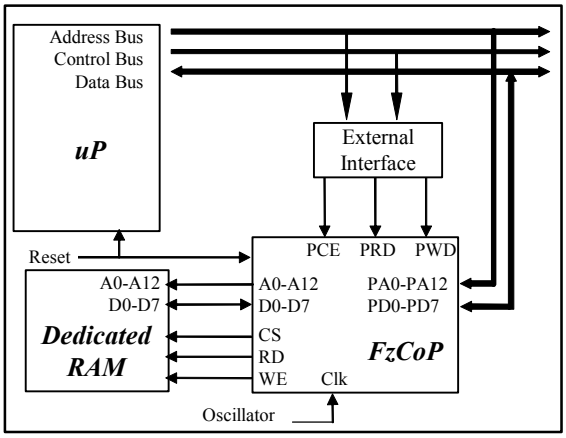

Figure 2. Overall system architecture

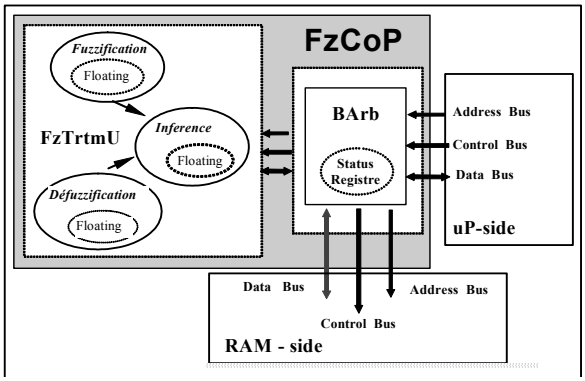

Figure 3. Architecture of FzCoP

The FzTrtmU performs three chained tasks on a host demand [5]: fuzzification, inference and defuzzification. All three carry out on the floating-point calculations. For this reason, a floating-point unit (FPU) is also modeled as a fourth task [12].

III. SPECIFICATION REFINEMENT OF FUZZY TREATMENTS UNIT

A. Fuzzification function specification

The fuzzification function converts any physical entry variable passed from the host into corresponding fuzzy variable in the FzCoP [4, 6, 11]. The value of the measured parameter is used as entry point of the function and the value of the corresponding linguistic (fuzzy) variable is calculated for each membership function associated to it. The fuzzification algorithm is based essentially on the table retrieving and linear interpolation calculus.

B. Inference function specification

The inference function deals with inference rules memorized in FzIfr-DB RAM. It is configured according to one of three user-selectable inference methods: max-min, sum-product and max-product [3, 9]. Figure 4 illustrate the inference algorithm specification for the case of two fuzzy variables passed trough multiple fuzzy rules, each of the following type:

$$R_k: \textit{if}\ (FV1\ \textit{is}\ MSF1_i)\ \textit{and}\ (FV2\ \textit{is}\ MSF2_j)\ \textit{then}\ MS\_SHAPE,$$

where

- R_k is the **k**-th inference rule for treated fuzzy variables stored in FzIfr-DB RAM,
- FV1 and FV2 are the fuzzy variable values at inference function entry,
- $MSF1_i$ and $MSF2_j$ are two particular membership functions out of all ones in the corresponding sets MSF1, respectively MSF2 for the variables, implicated in the test conditions,
- (FV1 *is* $MSF1_i$) respectively (FV2 *is* $MSF2_j$) are the values of membership functions for the given values of FV1 and FV2,
- MS_SHAPE is the deducted shape of the membership function for the resulting fuzzy variable at rule exit.

C. Defuzzification function specification

The defuzzification function transforms the area's shape of the membership function obtained at inference function leave into a single non-fuzzy corresponding value. The

FzCoP outputs it to the host so concluding the latest request. The algorithm specification gives the user a choice between one of two optional defuzzification methods:

- shape's center of gravity method and
- max-average method.

Although the center of gravity method is performance consuming in cases of using the max-min and the max-product inference methods it is reasonably appropriate for sum-product one.

In all cases the FzTrtmU has to conduct intense floating-point numeric computing. The VHDL description makes easy the modelling of FPU of FzCoP.

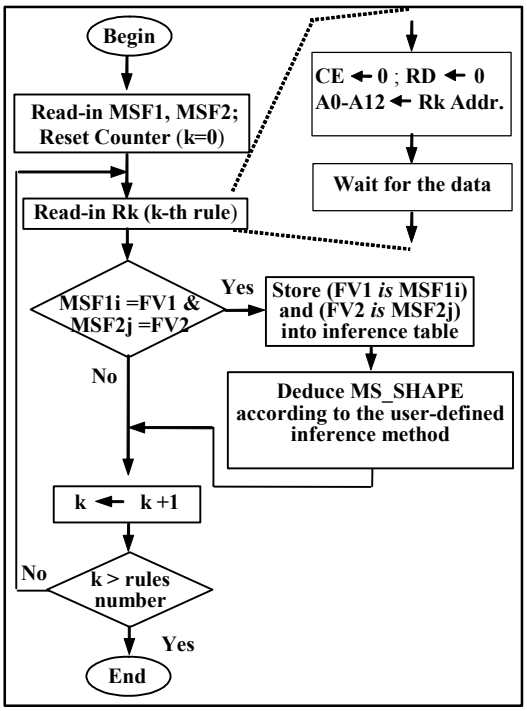

Figure 4. Two-variables fuzzy rule inference

D. Floating-point unit specification

The FPU has been performance optimized for fuzzy operations by means of a 16 bits proprietary version of floating-point format for real numbers representation [12]. Using 1 bit for the sign, 5 bits for the exponent and 10 bits for the mantissa, it reduces the circuit complexity without important loss of precision in many applications.

IV. DESCRIPTION AND VHDL VERIFICATION OF FZCOP

The FzCoP description is based on the adopted architecture presented at Figure 5. The bus arbiter BArb in the FzCoP gets the charge of sharing the RAM bus between two processors.

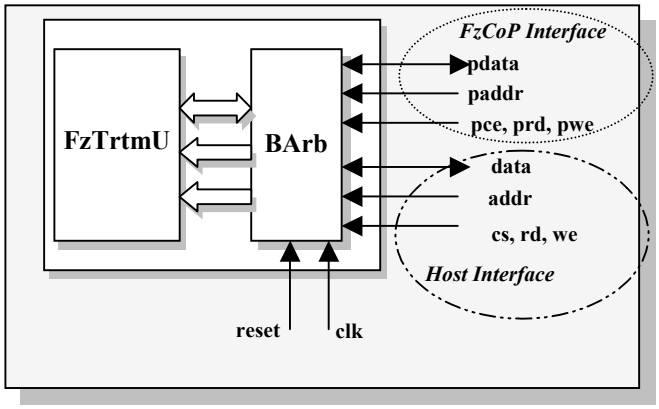

Figure 5. Entity description

A. Behavioral description

The modelling VHDL program is a high level behavioral description [2, 8, 12]. A code extract example is done in Table 1. The description is specified by means of sequentially instructions (i.e. if-then-else, wait, case, repeat, while …) as well as of concurrent instructions (process).

TABLE 1. CODE EXTRACT EXAMPLE FOR FzCoP DESCRIPTION

```
architecture arch_PF of En_Pf is
        « signal declarations »
begin
     arbitre : process
         « variables declarations »
         begin
         wait until clk='1';
          « sequentially instructions »
     end process arbiter
     fuzzy_calcul : process
         « variables declarations »
         begin
         wait until clk='1';
         fuzzification
         inference
         défuzzification
     end process fuzzy_calcul
end arch_Pf;
```

B. Test environment

The consistency of FzCoP design is verified on the environment connected to the tested FzCoP implementation [1, 13, 14] (see Figure 6). The association of VHDL components modelling external components, such a RAM on the Figure 6 makes the test procedures effortless.

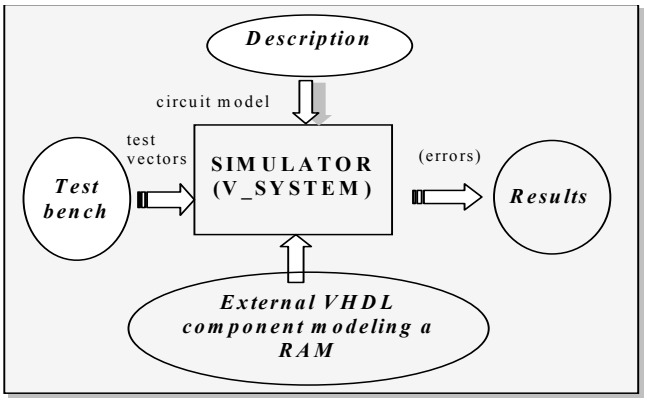

Figure 6. Test environment

The behavior is observed on different time diagrams. It is mandatory to create particular test vectors for FPU as well as for whole FzCoP. Figure 7 shows the result of FzCoP test using bit-vector and real number concepts.

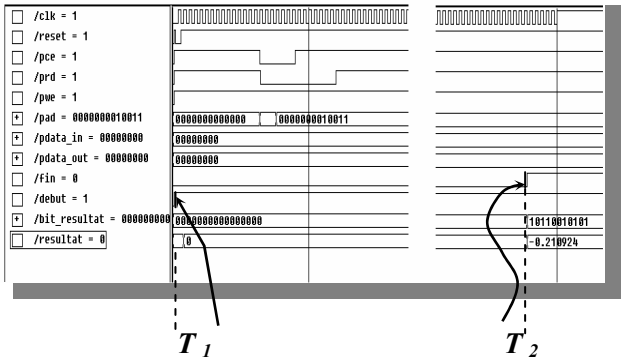

Figure 7. An FzCoP simulation time diagram

The FzCoP starts an operation at T1 instant, commanded by the signal ***debut*** that passes to 1. At T2 instant, it finishes requested operation acknowledging it by means of the signal ***fin*** that passes to 1. The simulated variable *bit_resultat* represents the binary value contrary to the variable *resultat* that is a real number result (see the numeric representations of these variables at T2 instant on the Figure 7).

V. CONCLUSIONS

The overall system architecture is simple, including three hardware units: the host uP, the FzCoP core described here and an external RAM.

Taking the FzIfr-DB RAM out of the FzCoP makes its implementation simpler and cost saving. First, the entire FzCoP design is surface efficient and can be logged on inexpensive, commercially available FPGA chips. Furthermore, it is not problematical to tune up the RAM capacity in function of the fuzzy rules and database complexity the user application has to match. And the last, the host to RAM interface allows the using of an external single-port RAM instead of more complex double-port RAM.

Modern FPGAs with incorporated blocks of RAM offer other option making this approach even practical.

Adopted solution makes constructive the FzCoP use in real-time applications with embedded processor in place of software implementations of the fuzzy logic.

VI. ACKNOWLEDGMENTS

Special thanks to S. Mannoubi for the help in preparing and testing a set of VHDL code examples.

VII. REFERENCES

[1] Abid, M., Mtibaa, A., and Tourki, R. « Conception d'un circuit de distribution de recommandation et de séquences d'un ETCD en émission : Modélisation et Simulation de haut niveau », *16-èmes Journées Tunisiennes d'Electrotechnique et d'Automatique, JTEA 96, 8-9 Novembre 1996,* Hammamet - Nabeul – Tunisie, pp. 123-128, Nov. 1996.

[2] Airiau, R., Berge, J.-M., Olive, V., and Rouillard, J. *VHDL : du langage à la modélisation*, Collection informatique, Presses Polytechniques et Universitaires Romandes. 1990.

[3] ARGO. *La logique floue,* Observatoire Français des Technologies Avancées, Paris : Masson, 1994.

[4] Barbat, J. P., Barbat, M., and Le Cluse, Y. « Applications de la logique floue », *Technique de l'ingénieur, Vol.* A, # A120-R70428, 1995.

[5] Bouchon, B. M. *La logique floue*. Paris : Presse Universitaire de France, 1993.

[6] Brule, J. F. « Fuzzy systems », *Electronique radio plan*, No 541, 1992.

[7] Buhler, H. *Réglage par la logique floue.* Lausanne : Presses Polytechniques et Universitaires Romandes, 1994.

[8] Calvez, J.-P. *Spécification et conception des ASICs.* Paris : MASSON, 1993.

[9] Hirota, K. *Industrial Applications of Fuzzy Technology.* New York: Springer-Verlag, 1993.

[10] IEEE Std. 1076.B-1987. VHDL Language Reference Manual.

[11] Kaufman, A. « La logique floue ». *Technique de l'ingénieur, Vol.* A, # A120-R7032, 1994.

[12] Mannoubi, S., Raytchev, R., and Mtibaa, A. « Conception d'un processeur flou », Master's thesis. Ecole Nationale d'Ingénieurs de Monastir, Monastir : Tunisie, 1997.

[13] Model Technology. *V_System/ Windows, User Manual: VHDL, Simulation for PCs Running Windows & Windows NT,* Ver. 4.3, June 1995.

[14] Mtibaa, A., Abid, M., and Tourki, R. « La méthodologie de conception de haut niveau des circuits intégrés fonctionnant en temps réels ». *17-èmes Journées*

Tunisiennes d'Electrotechnique et d'Automatique, JTEA 97, 5-6 Novembre 1997, Nabeul - Tunisie, pp. 239-244, Nov. 1997.

VIII. ABOUT AUTHORS

° R. Raychev, PhD, Assoc. Prof., Computer Systems and Technology Department, Technical University of Gabrovo, Bulgaria, Tel. +359 66 223 501, E-mail: raychev@tugab.bg

** Abdellatif Mtibaa, PhD, Assoc. Prof., EE Department, Engineering School of Monastir, Tunisia, Tel. +216 500 244, E-mail: abdellatif.mtibaa@enim.rnu.tn

*** Mohamed Abid, PhD, Assoc. Prof., EE Department, Engineering School of Sfax, Tunisia, E-mail: mohamed.abid@enis.rnu.tn