Jurnal Kejuruteraan 11(1) (1999) 53-60

A Top-Down Approach in Fuzzy Controller Design with VHDL

Khandaker Abul Kalam Azad
Zahari Mohamed Darus
Mohd Alauddin Mohd Ali

ABSTRACT

With the rapid increase in size and complexity of digital systems, hardware description languages, such as VHDL, are quickly becoming the main integral part of the tools used for advanced digital system design. Although many of the HDL-based design practices follow bottom-up fashion, the real strength of VHDL is that it facilitates the top-down design process where high level design concepts can be described and verified without delving into implementation details. The top-down design approach is described in this paper for designing a multi-level fuzzy controller.

ABSTRAK

Dengan peningkatan saiz dan kekompleksan sistem digit, bahasa perihalan hardwer, seperti VHDL, telah semakin cepat menjadi sebahagian daripada perkakasan utama yang digunakan untuk reka bentuk sistem digit lanjutan. Walaupun kebanyakan amalan reka bentuk yang berasaskan HDL mengikut fesyen bawah ke atas, kekuatan sebenar VHDL adalah ia memudahkan proses reka bentuk atas ke bawah yang mana konsep reka bentuk aras tinggi boleh diperihalkan dan disahkan tanpa menyelidiki perincian perlaksanaan. Pendekatan reka bentuk atas ke bawah diperihalkan dalam kertas kerja ini untuk mereka bentuk sebuah pengawal kabur berbilang aras.

INTRODUCTION

In the last twenty years, a change took place in the methodology of digital circuits design. In the past, integrated circuits were manually composed with graphical CAD-tools. For that purpose basic elements (logic gates from a library, or rather their symbols) had to be selected, placed on a schematic and connected with each other. In this way simple modules could be created which were then used to assemble complex circuits. This methodology is called bottom-up. It could take a long time to generate large circuits and the result was difficult to change because this meant laborious redrawing of the schematics. Today, designing electrical systems deals with more and more complex systems, which can be integrated in single chips due to the increasing packing density. A short development cycle is another decisive factor designers have to consider in order to stay on top of the competition and to satisfy the requirements of the customers. Therefore, the re-use of once generated functional blocks and module in new systems is important. This requires a technology independent description of the circuits. As far as digital circuits are concerned, the above considerations lead to the adoption of a top-down design flow. Using hardware description languages, modeling of systems at various levels of abstraction is possible. Due to the stepwise refinement in the top-down design flow, such a description language has to support all levels of abstraction: system specification, algorithmic description, functional blocks, and gate-level netlists. An important aspect in today's design flows is the use of synthesis tools, which automatically create gate-level netlists from behavioral descriptions. This requires a standardized language, which would allow the simulation of the modeled system at different levels of abstraction.

In recent years, the popularity of VHDL (IEEE 1988) as a hardware description language and a design tool has been increasing significantly. VHDL language has powerful capabilities that have several possible uses depending on its application. The language can be used to describe and specify a wide variety of electronic systems, at many levels of abstraction ranging from pure behavioral down to gate and switch level details, with the provision for specifying its timing explicitly. In addition to the description capability, systems modeled in VHDL can also be simulated at any of those same levels in order to verify their functional operation and performance parameters. The language provides support for modeling the system hierarchically and also supports top-down and bottom-up design methodologies. Precise simulation semantics are associated with all the language constructs, and therefore, models written in VHDL can be verified using a VHDL simulator. While it is often found that the traditional bottom-up design methodology is a more natural approach, as the size and complexity of digital systems increase, there is a growing need to explore ways to take advantage of layered characteristics of VHDL (Jain et al. 1989).

The VLSI CAD technology has been used for development of fuzzy logic based hardware. The digital hardware development approach deploys the state of the art VHDL based design methodologies for developing fuzzy logic integrated circuits. The suitability of VHDL has been viewed as a common hardware description language for specifications, simulation and synthesis of complex VLSI designs. VHDL has helped in exploring various alternatives for fuzzy logic hardware chip design (Chorafas 1990 and Costa et al. 1995). The rich descriptive capabilities of VHDL (Lipsett et al. 1989) and the algorithmic power of fuzzy logic (Zadeh 1965) complement each other. The reasons for this are (Zamfirescu 1992):

1 The robust descriptive nature of VHDL maps well into the computational demands of fuzzy logic.
2 VHDL supports the exact level of abstraction and information hiding required making fuzzy logic implementations user-friendly.
3 VHDL user-specifiable resolution functions and overloading capabilities map directly into a fuzzy logic paradigm, and
4 Modern fuzzy logic engineering requires support for hardware/software co-development, incremental modification of systems, reusability of modules, etc., all of which are enabled by VHDL.

In this paper, we discuss a multi-level, top-down hierarchical design approach we have adopted in a recent design practice. In this approach, a high-level VHDL model of a fuzzy logic controller (FLC) is designed and its characteristics are simulated by using Mentor Graphics QuickHDL. The important reasons for using VHDL instead of traditional schematic are shorter development times for electronic design and simpler maintenance design (Eichfeld et al. 1992, Togai 1986, Yamakawa et al. 1986 and Yamakawa 1987). The use of VHDL here emphasizes system verification rather than synthesis. More detailed information about the designed system, the design methodology, and further discussion are included in the rest of the sections.

DATA MODEL OF THE SYSTEM

The designed system is a fuzzy controller consisting of 4 main blocks as shown in Figure 1 with four input variables (A, B, C and D) and one output (Z) variable. Each variable has a precision of 8 bits.

Fuzzification transforms a crisp data value in a linguistic variable to membership degree. The membership degree depends on the shape of the membership function used. As the overlapping of the fuzzy sets is limited in the current implementation of 2, each crisp variable has to be transformed to a fuzzy variable X with at most 2 non-zero membership values μX_i and μX_{i+1} (Figure 2), resulting in a total of 8 non-zero membership values to be further processed.

In the middle part of Figure 2 the data model of the inference machine is illustrated. The processing of the membership values μX_i in the inference

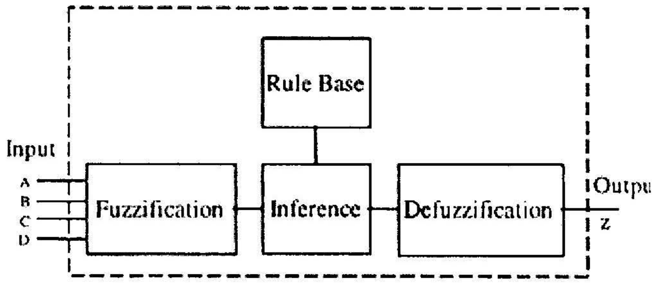

FIGURE 1. Model of a Fuzzy Logic System

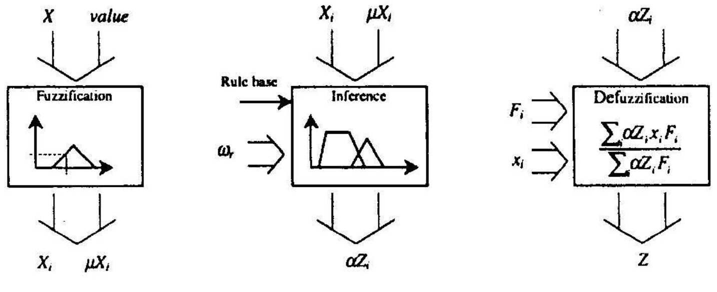

FIGURE 2. Data model of fuzzification (left), inference (middle) anddefuzzification (right). X represents one of the 4 input variables A, B, C or D, X_i is the identifier of the fuzzy sets of these variables, μXi is the membership value after fuzzification, Z is the output variable, Z_i represents the fuzzy set of the output variable, and finally αZ_i is the α value oftheoutput fuzzy sets (adopted from Jamshidi et al. 1993).

machine is controlled by the rule base. The contributions of each rule are individually weighted by the rule-weight factor ω_r. The output result of the inference machine are αZ_i values of the 8 output variable fuzzy sets Z_i. The individual rule weight factor is used to optimize fuzzy systems manually or automatically (Jacomet et al. 1995).

For defuzzification, two methods are implemented in our fuzzy controller, the maximum and the center of area (COA) method. In the center of area method it is easier and thus quite common to count overlapping regions twice. An overlapping region signifies that there are rules that came to a certain degree of same calculation. Therefore it is justified to weight this conformability twice. With this simplification we find the data model as illustrated in right of Figure 2.

DESIGN METHODOLOGY

Our design methodology is illustrated by the flowchart as shown in Figure 3. The design and verification process has three levels. The design started at

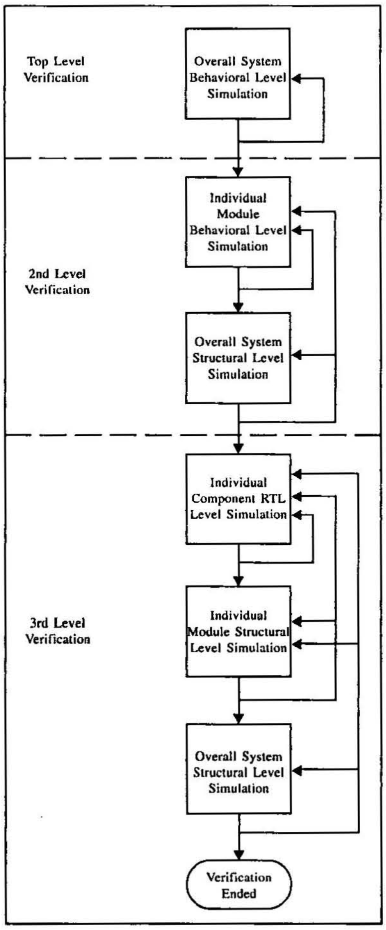
Top Level Verification
Overall System Behavioral Level Simulation
Individual Module Behavioral Level Simulation
2nd Level Verification
Overall System Structural Level Simulation
Individual Component RTL Level Simulation
3rd Level Verification
Individual Module Structural Level Simulation
Overall System Structural Level Simulation
Verification Ended

FIGURE 3. VHDL design and verification flow

the top level where the overall system was specified by behavioral level VHDL description. A test bench in VHDL code was used to verify the correctness of the design concept. This was done by comparing the systems output data to an expected data computed by a C++ program, for all possible input vectors. The design and verification process then proceeded to the second level where the individual modules of the datapath and the controller of the system were specified in the behavioral level VHDL code and then simulated by using VHDL test benches. When this step succeeded, the overall system was constructed by wiring all the modules and the controller together through structural level VHDL description. This two layer version of the overall system was then tested using the same VHDL test bench used for the top level verification and the results were compared to that obtained from the top level simulation. The third level of the design and verification process went further down to another layer where the components of each module in the datapath were coded in VHDL, mostly at RTL level. In other words, design at this level is technology dependent. After the components were simulated, the two layer versions of the modules were constructed by wiring components together by using structural level VHDL description, then verified by using the same test benches used for their one-layered versions in the second level verification, and compared to the previous test results. Finally, a three-layer version of the overall system was constructed based on the two-layer version of modules and again for the verification, the same test bench created before was used and the results were compared to the previous ones.

VHDL MODEL OF THE SYSTEM

Because of the feature that behavioral and structural modeling can be mixed in the design, VHDL is suitable for the modeling of Fuzzy Logic Controls (FLC). The main step in the VHDL implementation of the FLC involves a rule base building. Note that, one of the most important advantages of the presented architecture is the overall rule decomposition into sub-rules, each of which is stored in the local memory. The FLC system distribution into separate/parallel Rule Base Units (RBU) is the key to achieve the high latency of the data inferencing. This technique reduces the number of operations required for inferencing and achieves highest parallelism.

The fuzzification unit has been realized by an arithmetic calculation or by a lookup table procedure. As an arithmetic calculation can be quite time consuming, we implemented the fast lookup table method. A second advantage of the lookup table is its very high flexibility in defining membership functions. Our restriction of maximal 2 overlapping membership function leads to a small memory. Inputs to the fuzzifier are input variables and outputs are vectors containing membership degrees for all input fuzzy sets. To each input fuzzy variable corresponds one output vector, whose range depends on the number of fuzzy. Once the ranges of the input sets are determined, the input sets are removed from the entity declaration and are declared as constants.

The defuzzifier is modeled in a similar way. Inputs to the defuzzifier are vectors, one for each controlling variable. The range of the vector is determined by the number of the solution fuzzy sets - it contains a membership degree for each output fuzzy set. The defuzzifier determines the gravitational center of the solution fuzzy set and delivers the results as an output control signal. In implementing the defuzzifier, we have used the method proposed by Yager (1991). This approach requires 2N addition, N multiplication and one division, where N is the number of output fuzzy sets of a single output variable.

The high level description of the rule base were described into synthesizable VHDL code. From the hardware point of view, inferencing corresponds to the execution of a set of minimum operations for each rule. Implementations with active rule-driven architectures for the inference machine are reported in Ikeda et al. (1992) and Lemaitre et al. (1995). A part of the code of the rule base is given in Figure 4. This code evaluates rule 21 from the rule base. In process Rule_21, the calculation of the minimal membership degrees of the premises of rule 21 is performed. While coding the rules into VHDL, case statements have been used instead of if - then statements. Nested if – then statements are synthesized into a priority encoding circuit. On the other hand, a case statement synthesizes into hardware that does all the comparisons in parallel, faster than a priority encoding approach.

```
Rule_21 : PROCESS (memb_cof, memb_rs, state_low)
BEGIN
      CASE state_low IS
            WHEN s21b => min (memb_cof(LOW), memb_rs (LOW2));
            WHEN others => NULL;
      END CASE;
END PROCESS;

Out_calc : PROCESS (...., min21, min22, out_low ...)
BEGIN
      CASE out_low IS
            WHEN out2 => memb_prob(1) <= max (min21, min22);
            WHEN others => NULL;
      END CASE;
END PROCESS;
```

FIGURE 4. A part of VHDL code for rule base

DISCUSSION

Our experience of VHDL is described in this section to encourage its use in complex system design:

A substantial amount of storage modules (RAMs and ROMs) were included in designed system. Several parameterized procedures were written and included in a user-defined package to initialize and modify these memory modules.

We found that the efficient way to use the simulation timing information by summing up the delays on the same path within a module (second level) and assigning the total delay time to the output. At the behavioral level description of the module, signals within a process make transitions only when an event occurs on signals (inputs) in the sensitivity list of the process.

Therefore detailed delay assignments on intermediate signals often lead to wrong results. Referring to the following example:

```
ENTITY timing IS
PORT      (a : IN BIT;
           b  : OUT BIT);
END timing;
ARCHITECTURE timing_arch OF timing IS
  SIGNAL x,y : BIT := '0';
BEGIN
  PROCESS (a)
  BEGIN
          x <= NOT a AFTER 5 ns;
          y <= NOT x AFTER 6 ns;
          b <= y AFTER 7 ns;
  END PROCESS;
END timing_arch;
```

The above code intends to emulate a combinational circuitry where a is the input, y is the output, x and y are intermediate signals. Delays on each segment of the path are assigned to the correspondent signal explicitly. Since all assignments are based on the time when there is an event on a, simulation will bring misleading results on signals y and b. To remedy this problem, the code can be changed to:

```
ARCHITECTURE timing_arch OF timing IS
BEGIN
    PROCESS (a)
VARIABLE x,y : BIT := '0';
    BEGIN
      x := NOT a ;
      y := NOT x ;
      b <= y AFTER 18 ns;
    END PROCESS;
    END timing_arch;
```

CONCLUSION

It has been shown that with careful planning, VHDL can be a very efficient design tool and is especially suitable for designing large and complex systems. In addition to providing an organized, hierarchical approach toward complex systems design tasks, VHDL with a top-down design style allows a designer to verify the design concept starting from the topmost level before wasting too much time in lower level details. Furthermore the nature of VHDL makes it easy to change bit size and timing of any signal in a system. This flexibility makes it possible for a designer to experiment at a high level on systems with different sizes and timing characteristics. In our design, we kept our VHDL codes highly parameterized so that the size and timing characteristics of the system could be easily changed. During the verification process, the system was scaled down, which saved simulation runtime, reduced the requirements on hardware platforms and tremendously eased the debugging effort. This flexibility is hard to achieve when using schematic based design tools.

ACKNOWLEDGEMENT

We thank the Ministry of Science, Technology and the Environment of Malaysia for sponsoring this research under the Intensification of Research in Priority Areas (IRPA) grants (IRPA 04-02-02-0011 and 03-02-02-0033).

REFERENCES

Chorafas, D.N. & Steinmann, H. 1990. *Super computers*. McGraw Hill, Inc.

Costa, A., Gloria, A.D., Faraboschi, P., Pagni, A. & Rizzotta, G. 1995. Hardware solutions for fuzzy control. *Proc. Of the IEEE* 83: 422-34.

Eichfeld, H., Lohener, M. & Mueller, M. 1992. Architecture of a CMOS fuzzy logic controller with optimized memory organization and operator design. *Proceedings of the 1st IEEE International Conference of Fuzzy Systems*, pp. 1317-23.

IEEE. 1988. IEEE standard VHDL language reference manual. *IEEE Standard* 1076-1987.

Ikeda, Kisu, Hiramato & Nakamura. 1992. A fuzzy inference co-processor using flexible active-rule-driven architecture. *Information and Knowledge Engineering*, pp. 537-544.

Jacomet, M., Stahel, A. & Waelti, R. 1995. On-line optimization of fuzzy systems. *2nd Annual Joint Conference on Information Sciences*, pp. 490-493. Wrightsville Beach.

Jain, P.P., Dhingra, S., & Browne, J.C. 1989. Bring Top Down Synthesis. *High Performance Systems*, pp. 86-94.

Jamshidi, M., Vadiee, N. & Ross, T. 1993. *Fuzzy logic and control: Software and hardware applications.* New Jersey: Prentice Hall.

Lemaitre, L., Patyra & M.J. 1995. FIT arithmetic: toward a high performance Implementation of Digital Fuzzy Logic Circuits. *Fourth IEEE International Conference on Fuzzy Systems*, pp. 1627-1632.

Lipsett, R., Schafer, C. & Ussery, C. 1989. VHDL : *Hardware description and design.* New York: Kluwer Academic Publishers.

QuickHDL User's Manual. 1992. *Mentor Graphics Corporation.*

Togai, M. & Watanaba, H. 1986. A VLSI implementation of a fuzzy inference engine: Toward an expert system on a chip. *Information Science* 38: 147-163.

Yager, R. 1991. Connectiveness and quantifiers in fuzzy sets. *Fuzzy Sets and Systems* 40: 39-75.

Yamakawa, T. & Miki, T. 1986. The current mode fuzzy logic integrated circuits fabricated by the standard CMOS process. *IEEE Transactions on Computers* C-35(2): pp. 161-167.

Yamakawa, T. 1987. A Simple fuzzy computer hardware system employing MIN & MAX Operations - A Challenge to 6th Generation Computer. *2nd IFSA Congress* 2: 823-830.

Zadeh, L.A. 1965. Fuzzy sets. *Information and Control.* 8: 338-353.

Zamfirescu, A. and Ussery, C. 1992. VHDL and fuzzy logic if-then rules. *European Design Automation Conference, EURODAC'92*. pp. 636-41.

Khandaker Abdul Kalam, Zahari Mohamed Darus and Mohd Alauddin Mohd Ali
Department of Electrical, Electronic and Systems Engineering
Faculty of Engineering
Universiti Kebangsaan Malaysia
43600 UKM Bangi
Selangor D.E., Malaysia