Proceedings of the 12th
KACC, October 1997

Product-Sum 추론방식을 이용한 퍼지제어기의 FPGA구현

FPGA Implementation of Fuzzy Controller Using Product-Sum Inference Method

°김 재 회*, 박 준 열**

* 홍익대학교 전자공학과 (Tel : 02-334-4745; Fax : 02-320-1119; E-mail : gheekim@wow.hongik.ac.kr)
** 홍익대학교 전자공하과 (Tel : 02-320-1636; Fax : 02-320-1119; E-mail : parkjy@wow.hongik.ac.kr)

Abstracts This paper presents FPGA implementation of fuzzy controller using Product-Sum inference method. Product-Sum inference method has much better performance than other inference methods. This fuzzy controller is composed of several digital modules, e.g. fuzzifier, rule base, adder, multiplier, select center and divider, and is operated by error and error variation. We synthesised the fuzzy controller and performed wave simulation using Xilinx VHDL tool(ViewLogic, ViewSim).

Keywords Fuzzy Logic Controller, VHDL, FPGA

1. 서 론

일반적으로 퍼지제어기는 퍼지화 부분, 제어규칙 부분, 추론부분, 비퍼지화 부분으로 나눌 수 있다. 이 4가지 부분 중에서도 가장 많은 연구가 행해지는 부분이 추론부분이다.

기존의 퍼지 제어기들은 Mamdani가 제안한 Min-Max 방식의 추론방법을 이용하고 있다[3]. 이 추론 방식은 계산량이 적다는 장점을 가지고 있으나 정교한 제어가 힘들다는 단점을 또한 가지고 있다. 그래서 제안된 추론 방식이 Product-Sum 방식이다. Product-Sum 방식의 추론은 인간의 생각과 보다 유사하고 다른 추론 방식보다 좋은 성능을 보여주고 있다[1]. 그러나, 이 방식도 계산량이 많아지고 연산시간이 길어진다는 단점을 가지고 있다.

퍼지제어의 모든 추론 알고리즘은 소프트웨어 처리시 제어규칙이 많아지게 되면 데이터의 실시간 처리가 어렵게된다. 이 문제를 해결하기 위해 기존에는 부가적인 장비를 사용하였다. 한 예로써, 데이터의 처리속도가 빠른 DSP(Digital Signal Processor)인 TMS를 사용하여 실시간 처리를 하였다. 그러나 이 장비는 가격이 매우 비싸다. 그래서 퍼지 데이터의 처리에는 가격이 저렴한 전용 하드웨어가 꼭 필요한 실정이다.

본 논문에서는 Product-Sum 방식의 추론 알고리즘을 이용한 퍼지 제어 전용 하드웨어를 구현하고자 한다.

VHDL과 ASIC의 한 종류인 FPGA(Field Programmable Gate Array)를 이용하여 하나의 제어기를 구현하였다. FPGA를 이용하게 되면 레이아웃(layout)과정이 필요 없기 때문에 기존의 Full Custom ASIC나 Semi Custom ASIC을 이용할 때보다 제어기구현에 소요되는 시간과 비용을 줄일 수 있다. 이렇게 퍼지 제어 전용 하드웨어를 하나의 칩으로 구현하게 됨으로써 외부로부터의 왜란에 대한 영향 또한 크게 감소하게 되어 보다 좋은 성능을 얻을 수 있다. 또, 보다 소형화, 간소화된다면 적용범위의 확대를 기대할 수 있을 것이다. 본 제어기는 한 클럭에 데이터를 처리하기 위해서 모두 조합(Combinational)회로로 구현하였다.

본 논문은 Product-Sum 추론 방법, 퍼지 제어기의 구조, 구현된 퍼지 제어기, VHDL 파형 모의실험의 순서로 구성되어 있다.

2. Product-Sum 방식의 추론

Product-Sum 방식은 다음과 같이 정의된다.

추론결과인 C_i' 는 사실 「 x_0 , y_0 」 그리고 퍼지 룰 「 A_i and B_i ⇒ C_i 」에 의해서 구해진다.

$$\mu_{C_i'}(z) = \mu_{A_i}(x_0) \cdot \mu_{B_i}(y_0) \cdot \mu_{C_i}(z) \tag{1}$$

여기서, $\mu_{A_i}(x_0) \cdot \mu_{B_i}(y_0) = h_i$

그리고 C' 는 C_1' , C_2' , ⋯ , C_n' 의 sum(+)으로 구해진다.

$$C' = C_1' + C_2' + \cdots + C_n' \tag{2}$$

즉,

$$\mu_{C'}(z) = \mu_{C_1'}(z) + \cdots + \mu_{C_n'}(z) \tag{3}$$

또, C' 의 representative point z_0 은 다음과 같은 무게중심법에 의해 구해진다.

$$z_i = \frac{\int z \cdot \mu_{C_i'}(z)\,dz}{\int \mu_{C_i'}(z)\,dz} = \frac{\int z \cdot \mu_{C_i'}(z)\,dz}{S_i} \tag{4}$$

여기서, S_i 은 C_i' 의 면적이다.

$$\int z \cdot \mu_{C_i'}(z)\,dz = S_i z_i \tag{5}$$

그러므로,

$$z_0 = \frac{\int z \cdot \mu_{C'}(z)\,dz}{\int \mu_{C'}(z)\,dz} = \frac{\int z \cdot [\mu_{C_1'}(z) + \cdots + \mu_{C_n'}(z)]\,dz}{\int [\mu_{C_1'}(z) + \cdots + \mu_{C_n'}(z)]\,dz}$$

$$= \frac{\int z \cdot \mu_{C_1'}(z)\,dz + \cdots + \int z \cdot \mu_{C_n'}(z)\,dz}{\int \mu_{C_1'}(z)\,dz + \cdots + \int \mu_{C_n'}(z)\,dz} \tag{6}$$

$$= \frac{S_1 z_1 + \cdots + S_n z_n}{S_1 + \cdots + S_n}$$

따라서, 무게중심 z_0 은 S_i 에 z_i (C_i 의 무게중심)을 가중치한 것이다. 그래서 이 비퍼지화 방법을 면적법이라 한다.

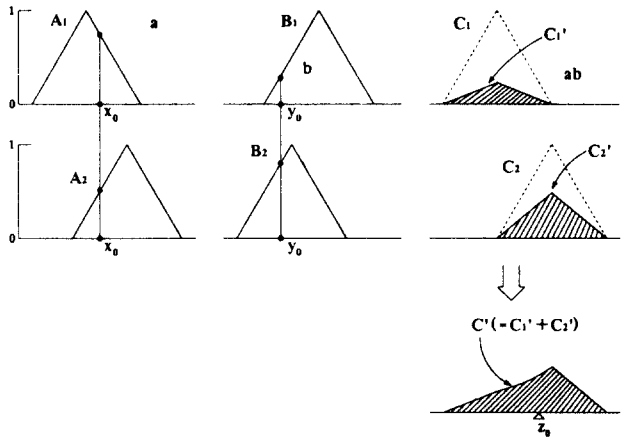

그림 1. Product-Sum 추론법.
Fig. 1. Product-Sum inference method.

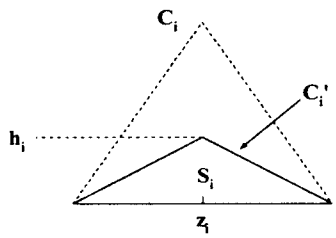

그림 2. 중심값 z_i 그리고 C_i' 의 면적 S_i.
Fig. 2. Center z_i and area S_i of C_i'.

3. 회로 구현

본 회로의 구조는 앞절에서 살펴본 product-sum 방식의 추론을 이용하였고 그 알고리즘을 디지탈 회로로 구현한 것이다.

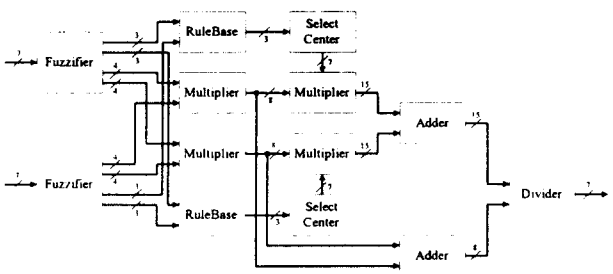

그림 3. Product-Sum 추론방식의 퍼지 제어기의 블럭도.
Fig. 3. Block diagram of fuzzy controller of Product-Sum inference method.

이 제어기는 2개의 퍼지화기(fuzzifier), 2개의 4×4 승산기(Multiplier), 2개의 8×8 승산기, 2개의 중심값 선택기, 1개의 8×8 가산기(Adder), 1개의 15×15 가산기, 1개의 제산기(Divider)로 구성되어 있다. 결과적으로 2개의 7-bit 입력을 받아서 1개의 7-bit 를 출력하게 된다.

3.1. 퍼지화기 (fuzzifier)

퍼지화기는 크리스프 입력 7-bit를 받아서 소속도값(membership vlaue) 인 4-bit 출력 2개와 퍼지집합을 나타내는 소속값(member value) 3-bit 출력 2개 도합 4개의 출력을 내보내고 있다. 여기서, 소속도 값은 4-bit 를 할당하였기 때문에 0000~1111 까지 16 단계로 나뉘어진다. 또, 입력에 대한 퍼지집합은 7개가 존재함으로 퍼지집합을 나타내기 위해 3-bit를 할당하였다. 그러므로, 크리스프 입력의 종류는 120개가 존재하게 된다. 그래서 크리스프 입력은 7-bit로 표현 가능하다.

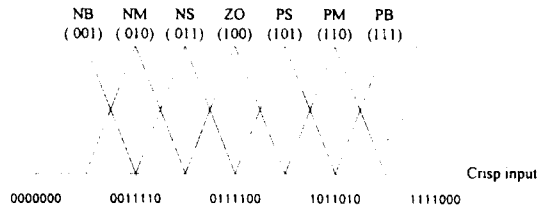

그림 4. 퍼지화기 구현을 위한 bit 할당.
Fig. 4. Bit assignment for fuzzifier implementation.

퍼지집합을 정의할 때 크리스프값의 범위가 중요하다. 이것은 전문가적 지식이나 실험 데이터 등을 통해 구해질 수 있다. 본 논문에서는 그 범위를 특별히 결정하지 않고 무조건 그 범위를 120개로만 구분하고 있다.

예) 만약 입력으로 0100111 이 인가되면
membership1 = 1001
membership2 = 0110
member1 = 011
member2 = 010
과 같은 출력을 내보낸다.

3.2. 제어규칙 (rule base)

표 1. 제어 규칙.
Table 1. Rule base.

Rule		Error Variation						
		NB	NM	NS	ZO	PS	PM	PB
Error	NB	PB	PB	PB	PM	PM	PM	PS
	NM	PB	PB	PM	PM	PM	ZO	NB
	NS	PB	PM	PM	PM	PS	NS	NB
	ZO	PB	PM	PS	ZO	NS	NM	NB
	PS	PB	PS	NS	NM	NM	NM	NB
	PM	PB	ZO	NM	NM	NM	NB	NB
	PB	NS	NM	NM	NM	NB	NB	NB

퍼지화기의 출력인 3-bit 의 소속값 2개를 입력으로 받아서 표 1.의 제어규칙에 의해 결과적으로 3-bit 의 출력 소속값을 출력한다. 이 제어규칙은 참고문헌[4]에서 사용한 유도전동기의 제어규칙을 인용하였다. 이 부분에서 숙련된 플랜트 조작자의 지식이 가장 필요하다.

3.3. 가산기 (adder)

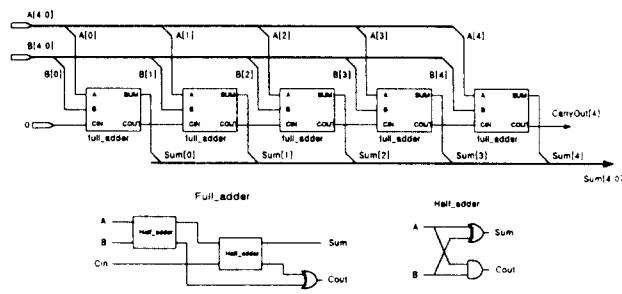

그림 5. 5×5 ripple carry 가산기의 블록도.
Fig. 5. Block diagram of 5×5 ripple carry adder.

가장 일반적으로 사용되는 가산기인 ripple carry 가산기의 구

조는 그림 5. 와 같다. bit의 확장은 전가산기(full-adder)를 더 사용하면 되고 구조는 동일하다. 그림 5.와 같은 구조의 가산기는 본 논문의 제어기에서 두 개가 사용된다. 하나는 8×8 가산기, 다른 하나는 15×15 가산기이다.

3.4. 승산기 (multiplier)

Shift-add 알고리즘은 그림 6.과 같다.

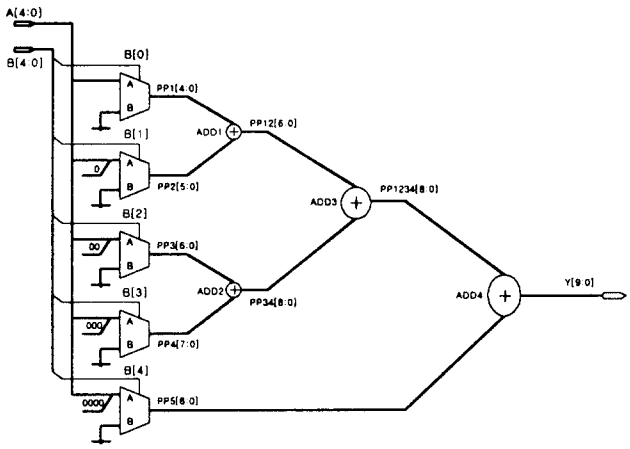

그림 6. 5×5 Shift-Add 승산알고리즘의 블록도.
Fig. 6. Block diagram of 5×5 Shift-Add Multiple algorithm.

Shift-add 알고리즘을 이용한 4×4 승산기와 8×8 승산기를 구현하였다.

3.5. 중심값 선택 (Select Center)

이 모듈(module)은 각 출력 소속값이 결정되면 그 소속값의 중심값을 결정하는 것이다. 이 모듈은 식 (6)에 의해 사용된다.
출력 퍼지집합 각각의 소속값에 대한 중심값은 표 2. 와 같다.

표 2. 각 출력 퍼지집합에 대한 중심값.
Table 2. Center value of each output fuzzy set.

member 값	중심값
NB (001)	0001111
NM (010)	0011110
NS (011)	0101101
ZO (100)	0111100
PS (101)	1001011
PM (110)	1011010
PB (111)	1101001

3.6. 제산기 (divider)

Shift-substract 알고리즘을 이용하여 15-bit를 8-bit 로 나누게 되고 7-bit의 결과값을 출력한다. Shift는 곱셈기와 같은 구조를 가지며 감산은 2's complement 를 이용하여 구현하였다.

4. 모의실험

앞 절에서 설명된 회로들은 Xilinx 사의 ViewLogic VHDL을 이용하여 작성되었고 Synthesis 되었다. 또, 파형 모의실험은 Xilinx 사의 ViewSim을 이용하였다.

그림 8.은 Synthesis 된 퍼지제어기이다. 마지막 단의 플립-플롭(Flip-Flop)은 게이트 지연(Gate Delay)에 의해 발생하는 글리치를 없애기 위한 것이다. 그러므로 입력이 들어간 뒤 한 클럭뒤에 출력을 얻을 수 있게 되어있다. 그림 9.는 제어기의 입력과 출력에 대한 파형 모의실험 결과이다.

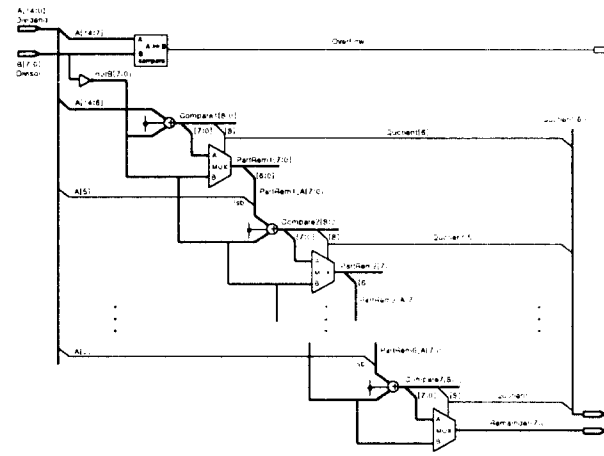

그림 7. 15/8 Shift-Substract 제산알고리즘의 블럭도.
Fig. 7. Block diagram of 15/8 Shift-Substract division algorithm.

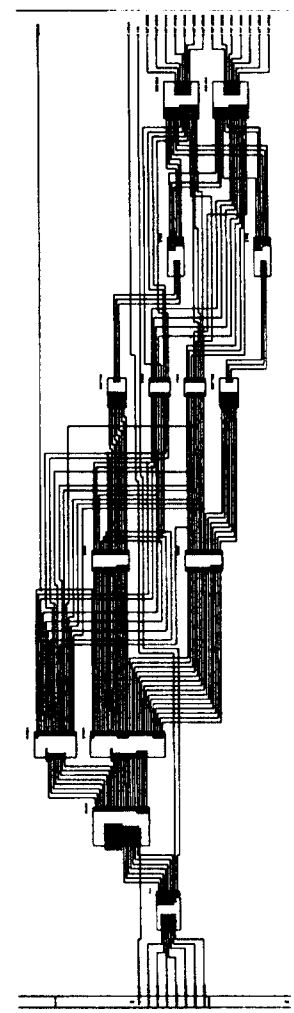

그림 8. Synthesis 된 퍼지 제어기.
Fig. 8. Synthesized fuzzy controller.

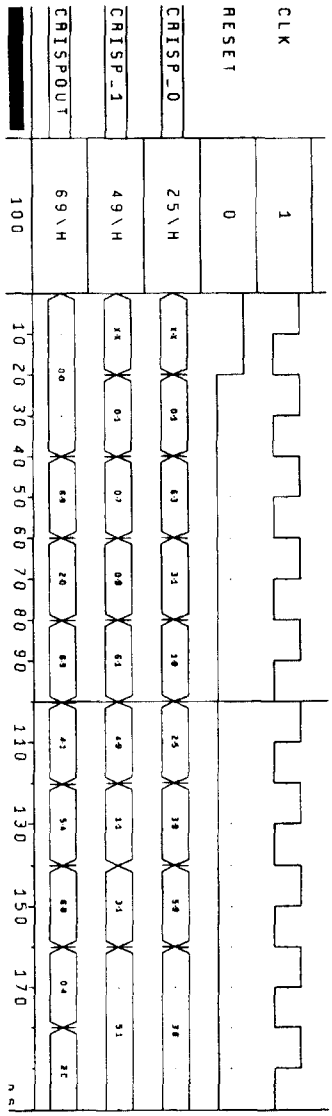

그림 9. 파형 모의실험 결과.
Fig. 9. Wave Simulation result.

표 3. 두 입력에 대해 계산된 출력값.
Table 3. Calculated output value for two inputs.

Input		Output
Error	Error Variation	
0000001	0000001	1101001
1100011	0000111	0101101
0110001	0001001	1101001
0011001	1100001	1000001
0100101	1001001	1011010
0111001	0010001	1101000
1011001	0110001	0000100
0111001	1010001	0101100

5. 결 론

그림 9.에서 보는 바와 같이 20ns의 주기를 갖는 클럭(Clock)을 사용하였고 한 주기마다 입력이 들어가고 있다. 또 결과값은 한클럭뒤에 출력되고 있음을 알 수 있다. 출력은 2개의 입력에 따라 우리가 원하는 출력임을 표 3.을 통해 확인할 수 있었다.

본 논문의 제어기도 16MHz~20MHz 정도의 동작주파수를 가진다. 또, FPGA를 이용하여 구현함으로써 부가적인 장비가 없이 간단하게 실시간 제어기를 구성할 수 있다는 결론을 얻을 수 있었다.

FPGA가 가지고 있는 단점은 게이트 배열간의 복잡한 연결때문에 동작 주파수가 20MHz~30MHz 이상을 가지기가 힘들다는 것이다. 그 이상의 동작주파수가 필요한 제어기나 매우 복잡한 제어기는 비록 많은 시간과 개발비가 들더라도 FPGA보다는 Full-Custom 이나 Semi-Custom ASIC을 통해 구현해야 할 것이다.

퍼지 제어기가 보다 간소화되고 소형화된다면 앞으로 산업현장이나 일반 가전제품 등에 적용범위가 더욱 넓어 질 것이다.

참 고 문 헌

[1] C. C. Lee, "Fuzzy Logic in Control System : Fuzzy Logic Controller. - Part I", IEEE Trans. Syst. Man Cybern., vol 20, no.2, pp. 404-418, 1990.

[2] C. C. Lee, "Fuzzy Logic in Control System : Fuzzy Logic Controller. - Part II", IEEE Trans. Syst. Man Cybern., vol 20, no.2, pp. 419-435, 1990.

[3] M. Mizumoto, "Fuzzy Controller Under Product-Sum-Gravity Methods And New Fuzzy Control Methods", Fuzzy Control Systems, pp. 276-294, CRC Press, 1994.

[4] Brian Heber, Longya Xu, Yifan Tang, "Fuzzy Logic Enhanced Speed Control of an Indirect Field Oriented Induction Machine Drive", IEEE Tran. Indust. App., 1995.

[5] Kai Hwang,"Computer Arithmetic : Principles, Architecture, And Design", John Wiley & Sons, 1979.

[6] Douglas J. Smith,"HDL Chip Design : A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGA using VHDL or Verilog", Doone Publication, 1996.

[7] Neil H. E. Weste, Kamran Eshraghian,"Principles Of CMOS VLSI Design", Addison Wesley, 1992.

[8] View Synthesis User's Guide, Xilinx, 1992.

[9] VHDL Reference Manual, Xilinx, 1992.