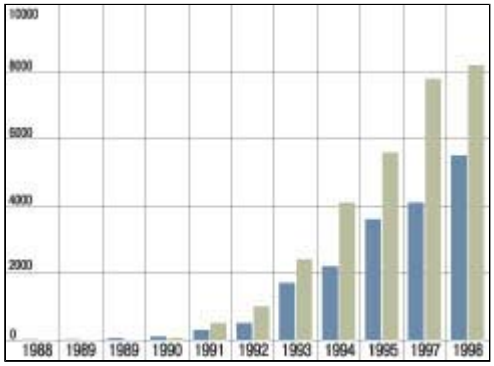
Нечеткая логика в системах управления

Сергей Гриняев, sgreen@hotbox.ru
Опубликовано: 8.10.2001

В последнее время нечеткая технология завоевывает все больше сторонников среди разработчиков систем управления. Взяв старт в 1965 году из работ профессора Лотфи Заде [1], за прошедшее время нечеткая логика прошла путь от почти антинаучной теории, практически отвергнутой в Европе и США, до банальной ситуации конца девяностых годов, когда в Японии в широком ассортименте появились «нечеткие» бритвы, пылесосы, фотокамеры [4, 10]. Сам термин «fuzzy» так прочно вошел в жизнь, что на многих языках он даже не переводится. В России в качестве примера можно вспомнить рекламу стиральных машин и микроволновых печей фирмы Samsung, обладающих искусственным интеллектом на основе нечеткой логики. Тем не менее, столь масштабный скачок в развитии нечетких систем управления не случаен. Простота и дешевизна их разработки заставляет проектировщиков все чаще прибегать к этой технологии. Бурный рост рынка нечетких систем показан на рис. 1.



После поистине взрывного старта прикладных нечетких систем в Японии [2, 3, 5, 6] многие разработчики США и Европы наконец-то обратили внимание на эту технологию. Но время было упущено, и мировым лидером в области нечетких систем стала Страна восходящего солнца [7, 8], где к концу 1980-х годов был налажен выпуск специализированных нечетких контроллеров, выполненных по технологии СБИС [9]. В такой ситуации Intel нашла поистине гениальное решение. Имея большое количество разнообразных контроллеров от MCS-51 до MCS-96, которые на протяжении многих лет успешно использовались во многих приложениях, корпорация решила создать средство разработки приложений на базе этих контроллеров, но с использованием технологии нечеткости. Это позволило избежать значительных затрат на конструирование собственных нечетких контроллеров, а система от Intel, получившая название fuzzy TECH, завоевала огромную популярность не только в США и Европе, но и прорвалась на японский рынок.

Немного теории

Нечеткая логика основана на использовании таких оборотов естественного языка, как «далеко», «близко», «холодно», «горячо». Диапазон ее применения очень широк - от бытовых приборов до управления сложными промышленными процессами. Многие современные задачи управления просто не могут быть решены классическими методами из-за очень большой сложности математических моделей, их описывающих. Вместе с тем, чтобы использовать теорию нечеткости на цифровых компьютерах, необходимы математические преобразования, позволяющие перейти от лингвистических переменных к их числовым аналогам в ЭВМ.

На рис. 2 показаны области наиболее эффективного применения современных технологий управления. Как видно, классические методы управления хорошо работают при полностью детерминированном объекте управления и детерминированной среде, а для систем с неполной информацией и высокой сложностью объекта управления оптимальными являются нечеткие методы управления. (В правом верхнем углу рисунка приведена еще одна современная технология управления - с применением искусственных нейронных сетей, но мы не станем столь глубоко вдаваться в достижения ученых.)

Вернемся к теории и кратко рассмотрим такие понятия, как «нечеткие правила», «нечеткий вывод» да и сам термин «нечеткое управление».

Классическая логика развивается с древнейших времен. Ее основоположником считается Аристотель. Логика известна нам как строгая и сугубо теоретическая наука, и большинство ученых (кроме разработчиков последнего поколения компьютеров) продолжают придерживаться этого мнения. Вместе с тем классическая или булева логика имеет один существенный недостаток - с ее помощью невозможно описать ассоциативное мышление человека. Классическая логика оперирует только двумя понятиями: ИСТИНА и ЛОЖЬ, и исключая любые промежуточные значения. Аналогично этому булева логика не признает ничего кроме единиц и нулей. Все это хорошо для вычислительных машин, но попробуйте представить весь окружающий вас мир только в черном и белом цвете, вдобавок исключив из языка любые ответы на вопросы, кроме ДА и НЕТ. В такой ситуации вам можно только посочувствовать. Решить эту проблему и призвана нечеткая логика. С термином «лингвистическая переменная» можно связать любую физическую величину, для которой нужно иметь больше значений, чем только ДА и НЕТ. В этом случае вы определяете необходимое число термов и каждому из них ставите в соответствие некоторое значение описываемой физической величины. Для этого значения степень принадлежности физической величины к терму будет равна единице, а для всех остальных значений - в зависимости от выбранной функции принадлежности. Например, можно ввести переменную ВОЗРАСТ и определить для нее термы ЮНОШЕСКИЙ, СРЕДНИЙ и ПРЕКЛОННЫЙ. Обсудив с экспертами значения конкретного возраста для каждого терма, вы с полной уверенностью можете избавиться от жестких ограничений логики Аристотеля.

Получившие наибольшее развитие из всех разработок искусственного интеллекта, экспертные системы завоевали устойчивое признание в качестве систем поддержки принятия решений. Подобные системы способны аккумулировать знания, полученные человеком в различных областях

деятельности. Посредством экспертных систем удается решить многие современные задачи, в том числе и задачи управления. Однако большинство систем все еще сильно зависит от классической логики.

Одним из основных методов представления знаний в экспертных системах являются продукционные правила, позволяющие приблизиться к стилю мышления человека. Любое правило продукций состоит из посылок и заключения. Возможно наличие нескольких посылок в правиле, в этом случае они объединяются посредством логических связок И, ИЛИ. Обычно продукционное правило записывается в виде: «ЕСЛИ (посылка) (связка) (посылка)… (посылка) ТО (заключение)».

Главным же недостатком продукционных систем остается то, что для их функционирования требуется наличие полной информации о системе.

Нечеткие системы тоже основаны на правилах продукционного типа, однако в качестве посылки и заключения в правиле используются лингвистические переменные, что позволяет избежать ограничений, присущих классическим продукционным правилам.

Целевая установка процесса управления связывается с выходной переменной нечеткой системы управления, но результат нечеткого логического вывода является нечетким, а физическое исполнительное устройство не способно воспринять такую команду. Необходимы специальные математические методы, позволяющие переходить от нечетких значений величин к вполне определенным. В целом весь процесс нечеткого управления можно разбить на несколько шагов: фаззификация, разработка нечетких правил и дефаззификация.

Рассмотрим подробнее эти шаги на примере поставляемой с пакетом fuzzy TECH модели контейнерного крана. Пусть вам, как маститому крановщику, необходимо перегрузить контейнер с баржи на железнодорожную платформу. Вы управляете мощностью двигателя тележки крана, заставляя ее двигаться быстрее или медленнее. От скорости перемещения тележки, в свою очередь, зависит расстояние до цели и амплитуда колебания контейнера на тросе. Вследствие того, что стратегия управления краном сильно зависит от положения тележки, применение стандартных контроллеров для этой задачи весьма затруднительно. Вместе с тем математическая модель движения груза, состоящая из нескольких дифференциальных уравнений, может быть составлена довольно легко, но для ее решения при различных исходных данных потребуется довольно много времени. К тому же исполняемый код программы будет большим и не поворотливым. Нечеткая система справляется с такой задачей очень быстро - несмотря на то, что вместо сложных дифференциальных уравнений движения груза весь процесс движения описывается терминами естественного языка: «больше», «средне», «немного» и т. п. То есть так, будто вы даете указания своему товарищу, сидящему за рычагами управления.

Фаззификация (переход к нечеткости)

Точные значения входных переменных преобразуются в значения лингвистических переменных посредством применения некоторых положений теории нечетких множеств, а именно - при помощи определенных функций принадлежности.

Рассмотрим этот этап подробнее. Прежде всего, введем понятие «лингвистической переменной» и «функции принадлежности».

Лингвистические переменные

В нечеткой логике значения любой величины представляются не числами, а словами естественного языка и называются ТЕРМАМИ. Так, значением лингвистической переменной ДИСТАНЦИЯ являются термы ДАЛЕКО, БЛИЗКО и т. д.

Конечно, для реализации лингвистической переменной необходимо определить точные физические значения ее термов. Пусть, например, переменная ДИСТАНЦИЯ может принимать любое значение

из диапазона от 0 до 60 метров. Как же нам поступить? Согласно положениям теории нечетких множеств, каждому значению расстояния из диапазона в 60 метров может быть поставлено в соответствие некоторое число, от нуля до единицы, которое определяет СТЕПЕНЬ ПРИНАДЛЕЖНОСТИ данного физического значения расстояния (допустим, 10 метров) к тому или иному терму лингвистической переменной ДИСТАНЦИЯ. В нашем случае расстоянию в 50 метров можно задать степень принадлежности к терму ДАЛЕКО, равную 0,85, а к терму БЛИЗКО - 0,15. Конкретное определение степени принадлежности возможно только при работе с экспертами. При обсуждении вопроса о термах лингвистической переменной интересно прикинуть, сколько всего термов в переменной необходимо для достаточно точного представления физической величины. В настоящее время сложилось мнение, что для большинства приложений достаточно 3-7 термов на каждую переменную. Минимальное значение числа термов вполне оправданно.Такое определение содержит два экстремальных значения (минимальное и максимальное) и среднее. Для большинства применений этого вполне достаточно. Что касается максимального количества термов, то оно не ограничено и зависит целиком от приложения и требуемой точности описания системы. Число же 7 обусловлено емкостью кратковременной памяти человека, в которой, по современным представлениям, может храниться до семи единиц информации.

В заключение дадим два совета, которые помогут в определении числа термов:

n исходите из стоящей перед вами задачи и необходимой точности описания, помните, что для большинства приложений вполне достаточно трех термов в переменной;

n составляемые нечеткие правила функционирования системы должны быть понятны, вы не должны испытывать существенных трудностей при их разработке; в противном случае, если не хватает словарного запаса в термах, следует увеличить их число.

Функции принадлежности

Как уже говорилось, принадлежность каждого точного значения к одному из термов лингвистической переменной определяется посредством функции принадлежности. Ее вид может быть абсолютно произвольным. Сейчас сформировалось понятие о так называемых стандартных функциях принадлежности (см. рис. 3).

Стандартные функции принадлежности легко применимы к решению большинства задач. Однако если предстоит решать специфическую задачу, можно выбрать и более подходящую форму функции принадлежности, при этом можно добиться лучших результатов работы системы, чем при использовании функций стандартного вида.

Подведем некоторый итог этапа фаззификации и дадим некое подобие алгоритма по формализации задачи в терминах нечеткой логики.

Шаг 1. Для каждого терма взятой лингвистической переменной найти числовое значение или диапазон значений, наилучшим образом характеризующих данный терм. Так как это значение или значения являются «прототипом» нашего терма, то для них выбирается единичное значение функции принадлежности.

Шаг 2. После определения значений с единичной принадлежностью необходимо определить значение параметра с принадлежностью «0» к данному терму. Это значение может быть выбрано как значение с принадлежностью «1» к другому терму из числа определенных ранее.

Шаг 3. После определения экстремальных значений нужно определить промежуточные значения. Для них выбираются П- или Л-функции из числа стандартных функций принадлежности.

Шаг 4. Для значений, соответствующих экстремальным значениям параметра, выбираются S- или Z-функции принадлежности.

Если удалось подобным образом описать стоящую перед вами задачу, вы уже целиком погрузились в мир нечеткости. Теперь необходимо что-то, что поможет найти верный путь в этом лабиринте. Таким путеводителем вполне может стать база нечетких правил. О методах их составления мы поговорим ниже.

Разработка нечетких правил

На этом этапе определяются продукционные правила, связывающие лингвистические переменные. Совокупность таких правил описывает стратегию управления, применяемую в данной задаче.

Большинство нечетких систем используют продукционные правила для описания зависимостей между лингвистическими переменными. Типичное продукционное правило состоит из антецедента (часть ЕСЛИ …) и консеквента (часть ТО …). Антецедент может содержать более одной посылки. В этом случае они объединяются посредством логических связок И или ИЛИ.

Процесс вычисления нечеткого правила называется нечетким логическим выводом и подразделяется на два этапа: обобщение и заключение.

Пусть мы имеем следующее правило:

ЕСЛИ ДИСТАНЦИЯ=средняя И

УГОЛ=малый, ТО **МОЩНОСТЬ=средняя**.

Обратимся к примеру с контейнерным краном и рассмотрим ситуацию, когда расстояние до платформы равно 20 метрам, а угол отклонения контейнера на тросе крана равен четырем градусам. После фаззификации исходных данных получим, что степень принадлежности расстояния в 20 метров к терму СРЕДНЯЯ лингвистической переменной ДИСТАНЦИЯ равна 0,9, а степень принадлежности угла в 4 градуса к терму МАЛЫЙ лингвистической переменной УГОЛ равна 0,8.

На первом шаге логического вывода необходимо определить степень принадлежности всего антецедента правила. Для этого в нечеткой логике существуют два оператора: MIN(…) и MAX(…). Первый вычисляет минимальное значение степени принадлежности, а второй - максимальное значение. Когда применять тот или иной оператор, зависит от того, какой связкой соединены посылки в правиле. Если использована связка И, применяется оператор MIN(…). Если же посылки объединены связкой ИЛИ, необходимо применить оператор MAX(…). Ну а если в правиле всего одна посылка, операторы вовсе не нужны. Для нашего примера применим оператор MIN(…), так как использована связка И. Получим следующее:

MIN(0,9;0,8)=0,8.

Следовательно, степень принадлежности антецедента такого правила равна 0,8. Операция, описанная выше, отрабатывается для каждого правила в базе нечетких правил.

Следующим шагом является собственно вывод или заключение. Подобным же образом посредством

операторов MIN/MAX вычисляется значение консеквента. Исходными данными служат вычисленные на предыдущем шаге значения степеней принадлежности антецедентов правил.

После выполнения всех шагов нечеткого вывода мы находим нечеткое значение управляющей переменной. Чтобы исполнительное устройство смогло отработать полученную команду, необходим этап управления, на котором мы избавляемся от нечеткости и который называется *дефаззификацией*.

Дефаззификация (устранение нечеткости)

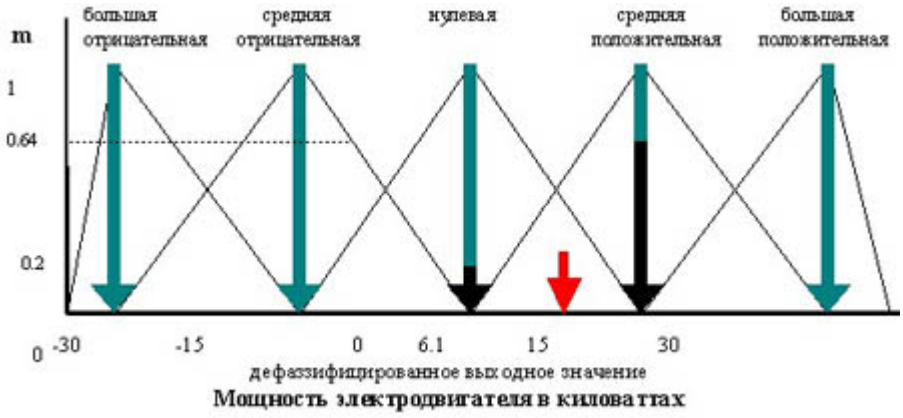
На этом этапе осуществляется переход от нечетких значений величин к определенным физическим параметрам, которые могут служить командами исполнительному устройству.

Результат нечеткого вывода, конечно же, будет нечетким. В примере с краном команда для электромотора крана будет представлена термом СРЕДНЯЯ (мощность), но для исполнительного устройства это ровно ничего не значит.

Для устранения нечеткости окончательного результата существует несколько методов. Рассмотрим некоторые из них. Аббревиатура, стоящая после названия метода, происходит от сокращения его английского эквивалента.

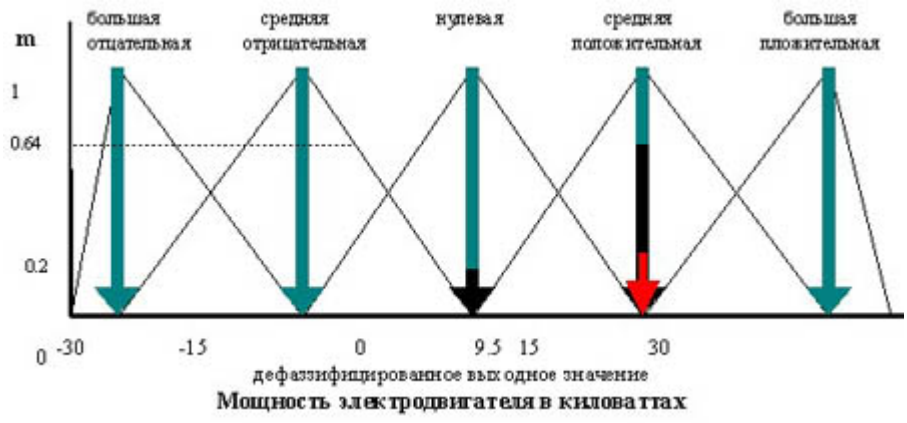
Метод центра максимума (CoM)

Так как результатом нечеткого логического вывода может быть несколько термов выходной переменной, то правило дефаззификации должно определить, какой из термов выбрать. Работа правила CoM показана на рис. 4.



Метод наибольшего значения (MoM)

При использовании этого метода правило дефаззификации выбирает максимальное из полученных значений выходной переменной. Работа метода ясна из рис. 5.

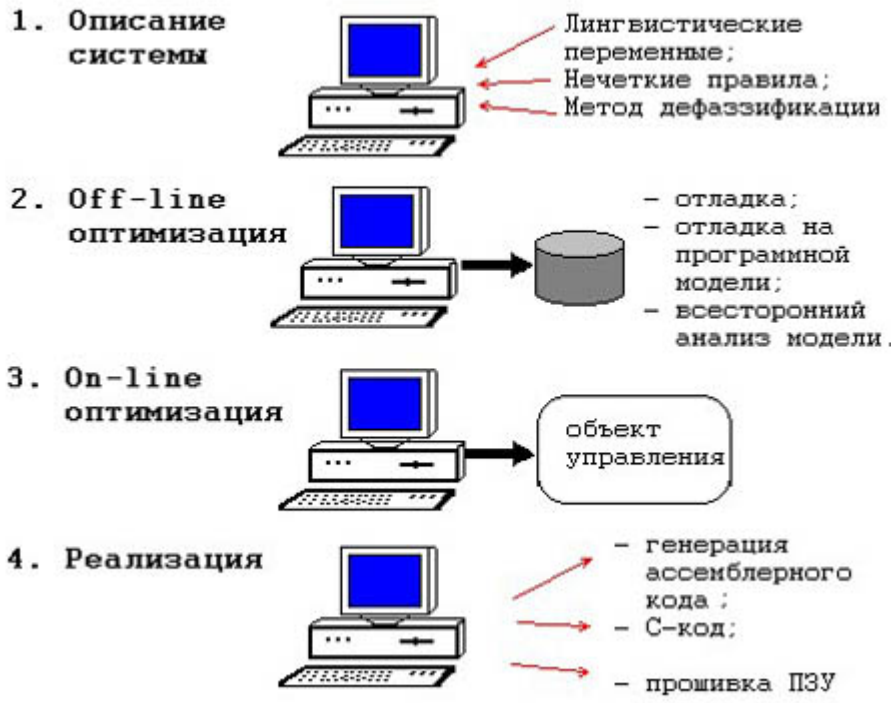


Метод центроида (CoA)

В этом методе окончательное значение определяется как проекция центра тяжести фигуры, ограниченной функциями принадлежности выходной переменной с допустимыми значениями. Работу правила можно видеть на рис. 6.

Основные шаги разработки нечеткой системы управления с использованием CAD-системы fuzzy TECH 3.0

Процесс разработки проекта нечеткой системы управления на fuzzy TECH разбивается, как уже говорилось, на четыре основных этапа. Все они схематично показаны на рис. 7.



Описание системы

На этом этапе при помощи средств, доступных в fuzzy TECH, задача формализуется. Здесь необходимо описать лингвистические переменные, которые вы будете использовать; их функции принадлежности; описать стратегию управления посредством нечетких правил, которые вы сможете объединить в единую базу правил или знаний о системе. В целом CASE-технология, на основе которой построен пакет, позволяет все эти действия выполнить только посредством общения с экраном ЭВМ, не заглядывая в программный код. Поэтому начальный этап проектирования вы воспримете с легкостью, несмотря на кажущуюся сложность. Можно дать один совет: обратите внимание на некоторые тонкости при разработке. Так, например, вы можете установить разрядность машинного кода, генерируемого пакетом. Это влияет на формат величин, которые можно использовать (см. табл. ниже).

Тип данных	Минимальное значение	Максимальное значение
8-битовый целочисленный	0	255
16-битовый целочисленный	0	32786
32-битный целочисленный	0	2147483648
Двойная точность	$1,7^{-308}$	$1,7^{+308}$

Off-line-оптимизация

На этом этапе следует проверить работоспособность созданной системы посредством всех средств fuzzy TECH. Отметим, что можно использовать заранее созданный программный симулятор вашего объекта управления, подобно модели контейнерного крана. Для связи системы управления с моделью используется специально разработанный протокол связи fTlink, в основу которого положена концепция обмена сообщениями Windows. Все необходимые средства для установления связи с вашей моделью находятся в исходных текстах программ связи, поставляемых с пакетом.

On-line-оптимизация

На этом шаге разрабатываемая система управления и реальный объект управления соединяются физической линией связи (см. рис. 8).

Такой вид отладки позволяет наблюдать поведение системы в реальных условиях и при необходимости вносить изменения в систему управления.

Реализация

На этом этапе необходимо получить окончательный вариант кода для конкретного микроконтроллера и, если нужно, связать его с вашей основной программой. Об оптимальности создаваемого fuzzy TECH кода можно судить по данным табл. ниже.

Основу программного кода, генерируемого пакетом fuzzy TECH, составляет аппаратно-ориентированное на конкретный тип процессора ядро. Поставляемое с пакетом fuzzy TECH MCU-96 программное ядро совместимо с такими контроллерами, как 8096BH, 8096-90, 80196KB/KC/KD, 80196 KR, 80196MC, 80196NT/NQ.

Важное замечание касается структуры генерируемого кода. Он, как правило, состоит из трех основных частей:

- код библиотечных функций;
- сегмент базы правил и функций принадлежности;
- функции нечеткой системы.

Найти объем ОЗУ и ПЗУ, потребный для работы и хранения вашей системы, помогут следующие формулы:

- для оперативной памяти,

$$S_v = \sum_{i=1}^{n_i} M\, t_i + \sum_{j=1}^{n_o} t_j + C,$$

где

S_v - объем необходимой оперативной памяти;

n_i - число входных переменных;

n_o - число выходных переменных;

t_i - число термов во входной лингвистической переменной i;

t_j - число термов в выходной лингвистической переменной j;

M- константа, равная 1 для 8-битного кода и 2 - для 16-битного;

C - константа, равная 28 байтам для MCU-96 и 7 байт для MCU-51;

- для постоянной памяти:,

$$S_f \le \sum_{i=1}^{n_i} 4M\,t_i + \sum_{j=1}^{n_o} t_j + \sum_{r=1}^{n_r} (I_r + 2Q_r + 2) + n_i,$$

где

S_f - размер базы правил в байтах;

n_i - число входных переменных;

n_o - число выходных переменных;

n_r - число правил в базе знаний;

t_i - число термов во входной лингвистической переменной i;

t_j - число термов в выходной лингвистической переменной j;

I_r - число входных условий для правила r;

O_r - число выходных условий для правила r;

M - константа, равная 1 для 8-битного кода и 2 - для 16-битного.

Точный размер сгенерированного fuzzy TECH 3.0 кода указывается по окончании процесса компиляции.

Платформа	**20 правил 2 вх. и 1 вых.**	**20 FAM-правил 2 вх. И 1 вых.**	**80 правил 3 вх. и 1 вых.**
MCS-96, 16 бит, 80C196KD, встроенное ПЗУ, 20 Мгц	0,28 мс 0,84 Кбайт ПЗУ 63 байт ОЗУ	0,29 мс 0,87 Кбайт ПЗУ 63 байт ОЗУ	0,43 мс 1.27 Кбайт ПЗУ 69 байт ОЗУ
MCS-51, 8 бит, 80C51, встроенное ПЗУ, 12 МГц	1,4 мс 0,54 Кбайт ПЗУ 25 байт ОЗУ	1,5 мс 0,58 Кбайт ПЗУ 25 байт ОЗУ	4,4 мс 1,0 Кбайт ПЗУ 29 байт ОЗУ

Литература

1. Zade L. A. The concept of a linguistic variable and its application to approximate reasoning. Part 1, 2, 3 // Information Sciences, n. 8 pp.199-249, pp.301-357; n. 9 pp. 43-80.

2. Прикладные нечеткие системы: Перевод с япон./ К. Асаи, Д. Ватада, С. Иваи и др.; под ред. Т. Тэрано, К. Асаи, М. Сугено. - М.: Мир, 1993.

3. Mamdani E. H. Applications of fuzzy algorithms for simple dynamic plant. Porc. IEE. vol. 121, n. 12, pp. 1585-1588, 1974.

4. Smidth F. L. Computing with a human face. New Scientist, 6 may, 1982.

5. Yagashita O., Itoh O., and Sugeno M. Application of fuzzy reasoning to the water purification process, in Industrial Applications of Fuzzy Control, Sugeno M, Ed. Amsterdam: North-Holand 1985,

pp.19-40.

6. Yasunobu S., Miyamoto S., and Ihara H. Fuzzy control for automatic train operation system, in Proc. 4th. IFAC/IFIP/IFORS Int. Congress on Control in Transportation Systems, Baden-Baden, April, 1983.

7. Yasunobu S., and Hasegawa T. Predictive fuzzy control and its applications for automatic container crane operation system, in Proc. 2nd. IFSA Congress, Tokyo, Japan, Julie 1987.

8. F. Fujitec, FLEX-8800 series elevator group control system, Fujitec Co., Ltd., Osaka, Japan, 1988.

9. Watanabe H., and Dettloff. Reconfigurable fuzzy logic processor: A full custom digital VLCI, in Int. Workshop on Fuzzy Systems Applications, Iiruka, Japan, Aug. 1988, pp. 49-50.

10. Sangalli A., and Klir G.R. Fuzzy logic goes to market, New Scientist, 8 Feb., 1992.

[i41520]

Телефон редакции: (095) 232-2263
E-mail редакции: site@computerra.ru
По вопросам размещения рекламы обращаться к Алене Шагиной или Ирине Лашковой по телефону +7 (095) 232-2263 или электронной почте reclama@computerra.ru