Применение языка Python при проектировании нечеткого контроллера

Игорь КУТЕПОВ
kutepovigor@rambler.ru

В статье рассматривается пример построения вычислителя на основе нечеткой логики и применение языка Python для написания интегрированной среды разработки.

Введение

В последнее время системы на основе нечеткой логики завоевывают все больше сторонников среди разработчиков систем управления.

Нечеткая логика — наиболее удобный способ построения систем управления сложными объектами и технологическими процессами. Она также нашла применение в бытовой электронике, диагностических и других экспертных системах. Несмотря на то, что математический аппарат нечеткой логики впервые был разработан в США, активное развитие этого метода началось в Японии, а сейчас волна интереса к этому методу вновь достигла США и Европы. В конце 1990-х годов в Японии появился широкий ассортимент «нечетких» бритв, пылесосов и фотокамер. Однако японские специалисты до сих пор занимаются этой темой, доказательством чего является все возрастающее количество патентов по нечеткой логике.

Нечеткое управление оказывается особенно полезным, когда технологические процессы слишком сложны для анализа с помощью общепринятых количественных методов или когда доступные источники информации интерпретируются неточно или неопределенно. Экспериментально показано, что нечеткое управление дает лучшие результаты по сравнению с получаемыми при общепринятых алгоритмах управления. Нечеткие методы помогают управлять домной и прокатным станом, автомобилем и поездом, распознавать речь и изображения, проектировать роботов, обладающих осязанием и зрением.

Для ряда задач электроэнергетики, решаемых в настоящее время, также возникает необходимость в применении вычислений на основе нечеткой логики. Примером применения нечеткой логики может служить построение системы защиты силового трансформатора [2]. Для решения поставленной задачи был разработан алгоритм защиты силового трансформатора на основе нечетких вычислений и формирования управляющих сигналов для исполнительных устройств системы защиты, а также спроектирована тестовая модель для проверки качества предложенного способа защиты.

Для удобства дальнейшего изложения введем следующие определения:

- Микроконтроллер — микросхема, предназначенная для управления электронными устройствами. Микроконтроллер сочетает функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять простые задачи.
- Контроллер — устройство на основе микропроцессора (микроконтроллера), служащее для управления технологическими процессами на производстве или для решения технологических задач автоматизированных систем управления. Контроллер собирает и обрабатывает данные с датчиков по программе, заданной пользователем, после чего выдает управляющие сигналы на исполнительные устройства.

Существующие нечеткие процессоры и микроконтроллеры

Некоторые производители элементной базы начали встраивать нечеткую логику непосредственно в процессоры. Так, в 1986 году компания AT&T Bell Labs начала создавать процессоры со встроенной нечеткой логикой обработки информации. В связи с популярностью и распространенностью традиционных микроконтроллеров в Европе и США ведутся интенсивные работы по интеграции нечетких команд в ассемблеры для программирования промышленных контроллеров (чипы Motorola 68HC11, 68HC12, 68HC21, Intel MCS96), а также разрабатываются различные варианты нечетких сопроцессоров, работающих совместно с центральным процессором по общей шине данных и помогающих в обработке информации и оптимизации использования правил (Siemens Nixdorf).

Некоторые существующие аппаратные нечеткие микроконтроллеры и процессоры:

- ST52 Dualogic (STMicroelectronics) — семейство 8-битных микроконтроллеров, содержащих в одном корпусе традиционное вычислительное ядро, ядро для нечетких вычислений и периферийные схемы. Поддерживает специальный набор инструкций для работы с нечеткой логикой и позволяет определять несколько независимых наборов правил для нескольких различных алгоритмов.
- ST62 (STMicroelectronics) — 8-битный микроконтроллер со встроенной, однократно программируемой памятью для автомобильной промышленности, продолжение семейства Dualogic. Имеет расширенный рабочий температурный диапазон (от –40 до +125 °C). Гарантированный срок хранения данных для памяти EPROM и EEPROM — не менее 20 лет.
- 68HC12 (Motorola) — нечеткий микроконтроллер, базирующийся на ядре стандартного микроконтроллера 68HC11 и содержащий специальные функции для реализации нечеткой логики. Предназначен для использования с программным пакетом FuzzyTech и позволяет увеличить скорость выполнения приложений, созданных в этом пакете, до 15 раз и компактность кода до 6 раз по сравнению с реализацией на обычном ядре 68HC11.
- VY86C570 (Togai InfraLogic) — нечеткий сопроцессор. 12-битное ядро FCA (Fuzzy Computational Acceleration, акселерация нечетких вычислений]), 4K×12 бит памяти OCTD (Observation, Conclusion & Temporary Data), память RB (Rule Base) и интерфейсная логика в одном корпусе.
- SAE 81C99 (Siemens) — нечеткий процессор, способный выполнять восемь программируемых алгоритмов, обрабатывать 256 входных переменных и формировать до 64 выходных значений максимум по 16 384 правилам. Может использоваться как отдельное устройство или в качестве сопроцессора для 8- и 16-разрядных микроконтроллеров. Скорость работы — 10 млн правил в секунду.

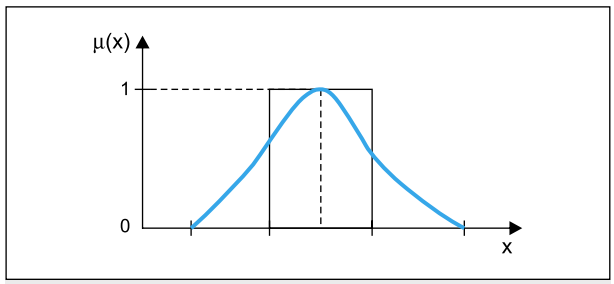

Рис. 1. Нечеткое множество и четкое классическое множество

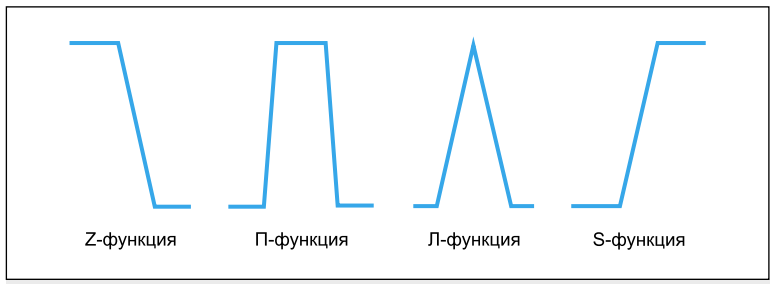

Рис. 2. Виды стандартных функций принадлежности

Помимо аппаратных решений, сейчас существуют такие системы моделирования, как MATLAB и LabVIEW, в которых есть программные модули для проектирования и моделирования контроллеров на основе нечетких вычислений.

Становится ясно, что имеется возможность построения систем нечеткого управления на основе простого и дешевого нечеткого контроллера, который бы с легкостью интегрировался в более крупную систему, и написания для него интегрированной среды разработки (integrated development environment, IDE).

Основы нечеткой логики

Рассмотрим основные понятия нечеткой логики и алгоритмы нечеткого логического вывода.

Нечеткая логика — это раздел математики, являющийся обобщением классической логики и теории множеств. Понятие нечеткой логики было впервые введено профессором Лотфи Заде в 1965 году. Понятие множества было расширено допущением, что функция принадлежности элемента к множеству может принимать любые значения в интервале [0…1], а не только 0 или 1. Такие множества были названы нечеткими.

Теория нечетких множеств — это расширение классической теории множеств и используется в нечеткой логике. Ее также разработал Лотфи Заде.

В классической теории принадлежность элементов множеству оценивается в бинарных терминах в соответствии с четким условием: элемент принадлежит либо нет данному множеству. Напротив, теория нечетких множеств разрешает градуированную оценку отношения принадлежности элементов множеству, то есть это отношение описывается при помощи функции принадлежности $\mu \rightarrow [0,1]$.

Нечеткие множества — это расширение классической теории множеств, поскольку на некотором множестве функция принадлежности может действовать так же, как индикаторная функция, отображая все элементы либо в 1, либо в 0, как в классическом варианте.

Нечеткое множество на классическом множестве X определяется как:

$$\tilde{A} = \{(x, \mu_A(x)) \mid x \in X\}. \quad (1)$$

Функция принадлежности $\mu_A(x)$ количественно градуирует принадлежность элементов x фундаментальному множеству X. Отображение элемента в значении 0 означает, что элемент не принадлежит данному множеству, значение 1 соответствует полной принадлежности элемента множеству. Значения, лежащие строго между 0 и 1, характеризуют «нечеткие» элементы.

На рис. 1 черным цветом показано множество X, а синим — нечеткое множество $\tilde{A}$. Вид функции принадлежности может быть абсолютно произвольным. В последнее время сформировалось понятие о так называемых стандартных функциях принадлежности (рис. 2).

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.

α-отсечением (или множеством α-уровня) нечеткого множества $\tilde{A}$ называется четкое подмножество универсального множества U, элементы которого имеют степени принадлежности, большие или равные α: $A_\alpha = \{u: \mu_A(u) \geq \alpha\}$, $\alpha \in [0,1]$. Значение α называют α-уровнем.

Дефаззификацией называется процедура преобразования нечеткого множества в четкое число. В теории нечетких множеств процедура дефаззификации аналогична нахождению характеристик положения (математического ожидания, моды, медианы) случайных величин в теории вероятности. Простейшим способом выполнения процедуры дефаззификации является выбор четкого числа, соответствующего максимуму функции принадлежности.

Существуют такие методы дефаззификации, как:

- Centroid — центр тяжести;
- LOM (Largest Of Maximums) — наибольший из максимумов;
- SOM (Smallest Of Maximums) — наименьший из максимумов;
- MOM (Mean Of Maximums) — центр максимумов.

Дефаззификация нечеткого множества по методу центра тяжести осуществляется по формуле:

$$y = \frac{\int_{x0}^{xn} x\mu(x)dx}{\int_{x0}^{xn} \mu(x)dx}. \quad (2)$$

Физическим аналогом этой формулы является нахождение центра тяжести плоской фигуры, ограниченной осями координат и графиком функции принадлежности нечеткого множества. В случае дискретного универсального множества дефаззификация нечеткого множества по методу центра тяжести осуществляется по формуле:

$$y = \frac{\sum_{i=1}^{k} x_i \mu(x_i)}{\sum_{i=1}^{k} \mu(x_i)}. \quad (3)$$

Дефаззификация нечеткого множества по методу центра максимумов осуществляется по формуле:

$$y = \frac{\int_G x dx}{\int_G dx}, \quad (4)$$

где G — множество всех элементов из интервала $[x_0, x_k]$, имеющих максимальную степень принадлежности нечеткому множеству.

В методе центра максимумов находится среднее арифметическое элементов универсального множества, имеющих максимальные степени принадлежностей. Если множество таких элементов конечно, то формула (2) упрощается:

$$y = \frac{\sum_{x_j \in G} x_j}{|G|}, \quad (5)$$

где $|G|$ — мощность множества G.

В дискретном случае дефаззификация по методам наибольшего и наименьшего из максимумов осуществляется по формулам $y = \max(G)$ и $y = \min(G)$ соответственно. Согласно последним трем формулам ясно, что если функция принадлежности имеет только один максимум, то его координата и является четким аналогом нечеткого мно-

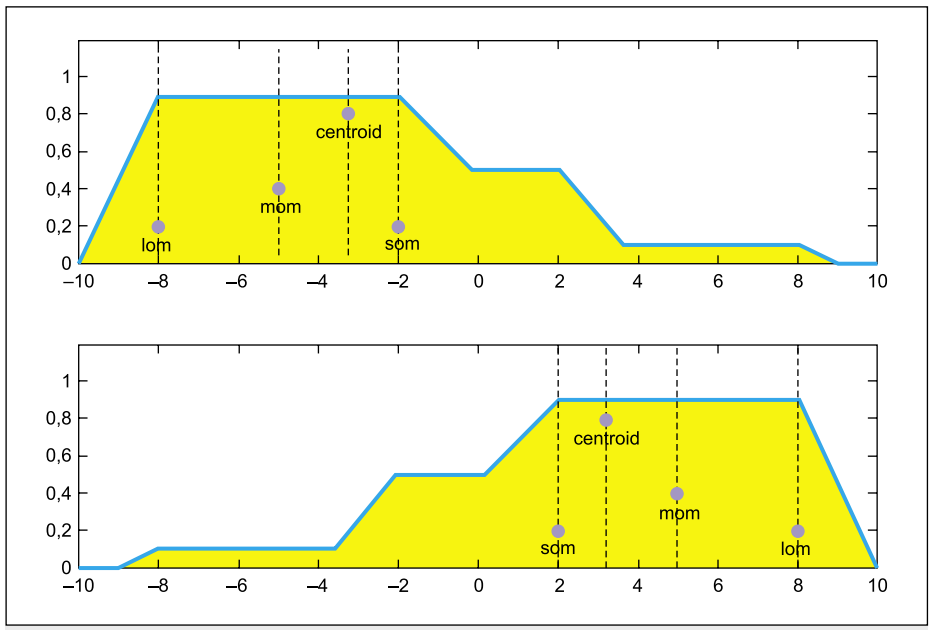

Рис. 3. Методы дефаззификации

жества. Графически методы дефаззификации представлены на рис. 3.

Нечеткой базой знаний называется совокупность логических высказываний типа: если x_1 есть A И (ИЛИ) x_2 есть B И (ИЛИ) … x_n есть C, то y есть Z, где A, B, C, Z — заданные функции принадлежности для переменных x_1, x_2, x_n, y.

Если использована связка И, применяется операция min. Если же посылки объединены связкой ИЛИ, необходимо применить операцию max.

Нечетким логическим выводом называется аппроксимация зависимости с помощью нечеткой базы знаний и операций над нечеткими множествами.

Операции с нечеткими множествами

Определим базовые операции (действия) над нечеткими множествами. Аналогично действиям с обычными множествами определим пересечение, объединение и отрицание нечетких множеств. В своей работе по нечетким множествам Лотфи Заде предложил оператор минимума для пересечения и оператор максимума для объединения двух нечетких множеств. Легко понять, что эти операторы совпадают с обычными (четкими) объединением и пересечением, только рассматриваются степени принадлежности 0 и 1. Чтобы пояснить это, рассмотрим пример.

Пусть А — нечеткий интервал от 5 до 8 и В — нечеткое число около 4, как показано на рис. 4.

Проиллюстрируем операцию пересечения нечетких множеств (синяя линия на рис. 5а), вычисляемого по формуле:

$$\mu_{A\cap B} = \min(\mu_A(x), \mu_B(x)). \qquad (6)$$

Проиллюстрируем операцию объединения нечетких множеств (синяя линия на рис. 5б), вычисляемого по формуле:

$$\mu_{A\cap B} = \max(\mu_A(x), \mu_B(x)). \qquad (7)$$

Проиллюстрируем операцию отрицания нечеткого множества, вычисляемого по формуле (синяя линия на рис. 5в):

$$\mu_{\bar{A}} = 1-\mu_A(x). \qquad (8)$$

Далее рассмотрим основные алгоритмы нечеткого вывода [1].

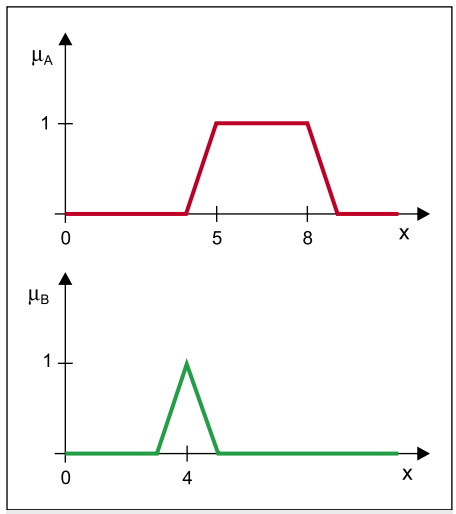

Рис. 4. Исходные данные

Алгоритм Мамдани

Предположим, что система описывается следующими правилами:

- если x есть A_1 И y есть B_1, то z есть C_1;
- если x есть A_2 И y есть B_2, то z есть C_2,

где x и y — входные переменные; z — выходная переменная; A_1, A_2, B_1, B_2, C_1, C_2 — заданные функции принадлежности.

На первом этапе находят степени принадлежности для каждого правила: $A_1(x_0)$, $B_1(x_0)$, $A_2(y_0)$, $B_2(y_0)$.

На втором этапе находят α-уровни для каждого из правил по формулам:

$$\alpha_1 = \min\{A_1(x_0), B_1(x_0)\}, \qquad (9)$$

$$\alpha_2 = \min\{A_2(x_0), B_2(x_0)\}. \qquad (10)$$

Затем находят усеченные функции принадлежности правил по формулам:

$$C_1(z) = \min\{\alpha_1, C_1(z)\}, \qquad (11)$$

$$C_2(z) = \min\{\alpha_2, C_2(z)\}. \qquad (12)$$

На третьем этапе производят объединение найденных усеченных функций, что приводит к получению итогового нечеткого подмножества для выходной переменной с функцией принадлежности по формуле:

$$\mu_\Sigma(z) = \max\{C_1(z), C_2(z)\}. \qquad (13)$$

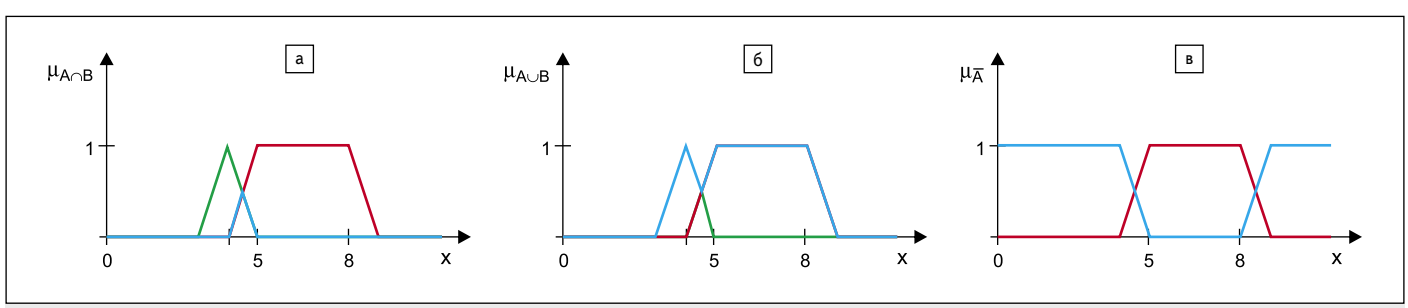

Рис. 5. Нечеткие множества: а) пересечение; б) объединение; в) отрицание

На четвертом этапе проводят операцию дефаззификации, например по методу центра тяжести.

Работа алгоритма показана на рис. 6. Для большей наглядности на рисунке приведен пример для трех нечетких правил.

Алгоритм Такаги-Сугэно

Этот алгоритм использует набор правил в следующей форме (для примера приведем два правила):

- если x есть A_1 И y есть B_1, то $z = a_1x+b_1y$;
- если x есть A_2 И y есть B_2, то $z = a_2x+b_2y$.

Особенностью алгоритма является то, что в качестве функций принадлежности выходной переменной используются полиномы, обычно это полиномы нулевого и первого порядка. В таких случаях используют названия «алгоритм Такаги-Сугэно» нулевого и первого порядка соответственно.

В алгоритме Такаги-Сугэно нулевого порядка (называемом также упрощенным алгоритмом нечеткого вывода) правила имеют вид: если x_1 есть A_1 и x_2 есть A_2… и x_n есть A_n, то $z = c$, где $c = \text{const}$.

Отметим, что алгоритм Такаги-Сугэно нулевого порядка можно интерпретировать как частный случай алгоритма Мамдани, когда функции принадлежности заключений правил имеют вид:

$$\mu_i(x) = \begin{cases} 1, & \text{при } x = c_i \\ 0, & \text{при } x \neq c_i \end{cases}. \qquad (14)$$

Алгоритм состоит из следующих шагов:

- На первом шаге так же, как и в алгоритме Мамдани, находят степени принадлежности для каждого нечеткого правила.
- На втором шаге вычисляют α-уровни для каждого из правил по формулам (9) и (10), а также индивидуальные выходы правил по формулам:

$$z_1^* = a_1x_0+b_1x_0, \qquad (15)$$

$$z_2^* = a_2x_0+b_2x_0. \qquad (16)$$

- На третьем шаге вычисляют четкое значение выходной переменной по формуле:

$$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1+\alpha_2}. \qquad (17)$$

Работа алгоритма показана на рис. 7.

Вариант построения нечеткого контроллера

Описываемый вариант нечеткого контроллера построен на базе ПЛИС EP3C25E144C8 фирмы Altera и микроконтроллера ARM Cortex-M4 фирмы STMicroelectronics. Микроконтроллер сопряжен с ПЛИС через интерфейс внешней памяти. Непосредственно вычислением алгоритмов нечеткой логики занимается ПЛИС. Микроконтроллер работает под управлением операционной системы реального времени (real-time operating system, RTOS). В ядро RTOS интегрированы функции для работы с нечетким контролле-

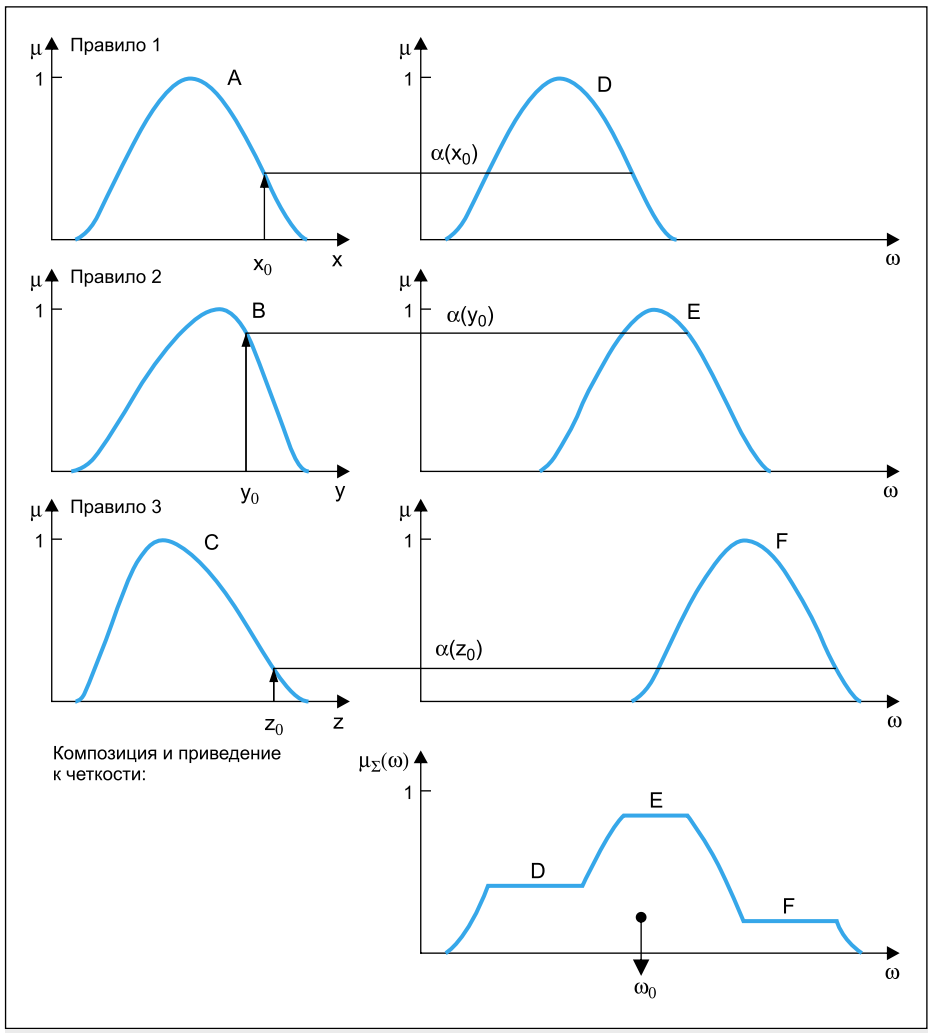

Рис. 6. Работа алгоритма Мамдани

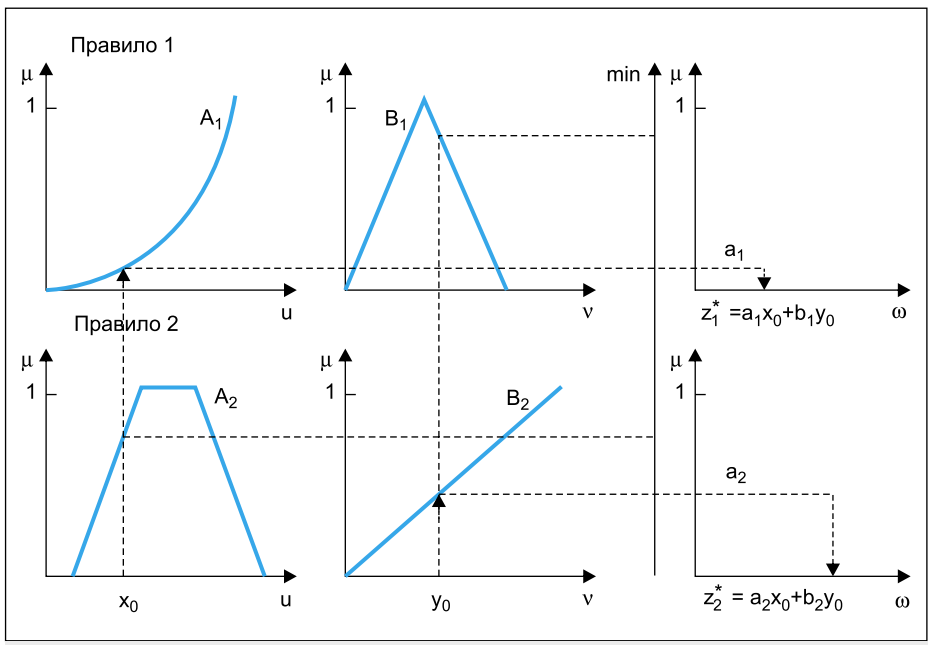

Рис. 7. Работа алгоритма Такаги-Сугэно

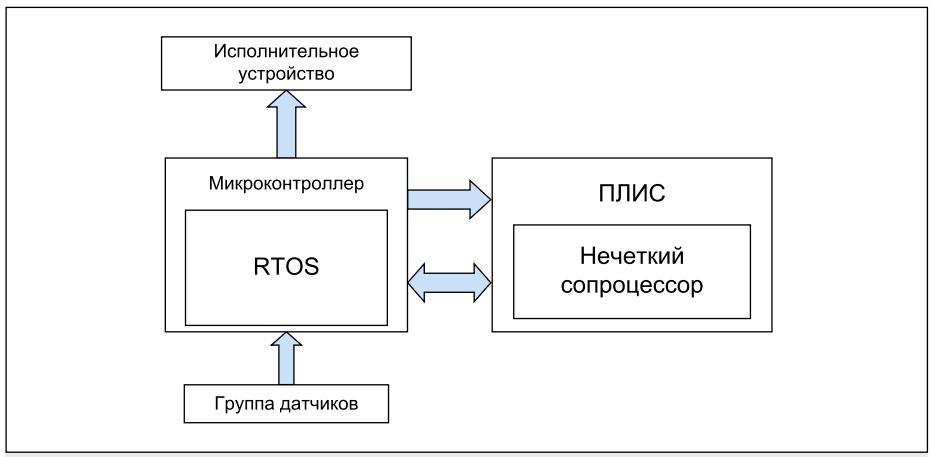

Рис. 8. Структурная схема нечеткого контроллера

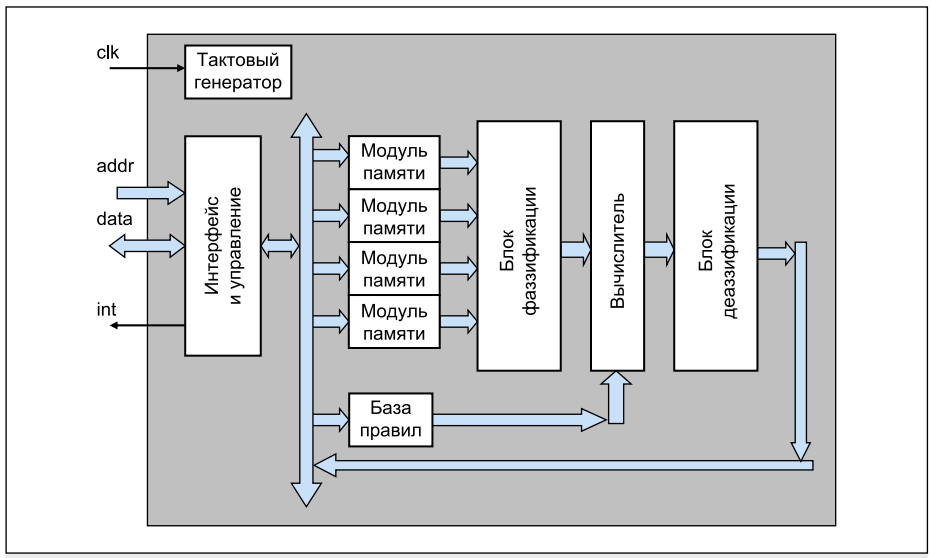

Рис. 9. Структурная схема нечеткого сопроцессора

ром. Таким образом, ПЛИС по отношению к микроконтроллеру является сопроцессором. Структурная схема нечеткого контроллера представлена на рис. 8.

Богатая периферия микроконтроллера позволяет непосредственно подключать к нему датчики и управляющие устройства с различными цифровыми интерфейсами. Также в состав микроконтроллера входит многоканальный аналого-цифровой преобразователь (АЦП) и 2-канальный цифро-аналоговый преобразователь (ЦАП), поэтому ARM Cortex-M4 можно легко интегрировать в контур управления автоматизированной системы. RTOS значительно облегчает разработку программной части, так как все основные функции управления периферией и нечетким контроллером уже реализованы в ядре операционной системы. Поэтому можно сосредоточиться непосредственно на решении поставленной задачи.

В описываемом нечетком контроллере используется RTOS собственной разработки, но это не означает, что применение какой-либо стандартной RTOS невозможно. Ничто не мешает использовать такие общеизвестные операционные системы, как FreeRTOS, scmRTOS, ECOS и т. д.

Нечеткий сопроцессор, реализованный на ПЛИС, имеет следующие характеристики:

- Число входов — 4.
- Число выходов — 1.
- Разрядность входных и выходных данных — 9 бит.
- Объем базы правил — 256 правил.
- Поддерживается два типа контроллеров: Мамдани и Такаги-Сугэно нулевого порядка.
- Методы дефаззификации: Centroid, LOM, SOM, MOM.

Структурная схема нечеткого сопроцессора приведена на рис. 9.

Интегрированная среда разработки для этого нечеткого контроллера была написана на языке Python.

Python — активно развивающийся язык программирования. Богатая стандартная библиотека является одной из его привлекательных сторон. Здесь имеются средства для работы со многими сетевыми протоколами и форматами Интернета, например модули для написания HTTP-серверов и клиентов, разборки и создания почтовых сообщений, работы с XML и т. п. Набор модулей для работы с операционной системой позволяет писать кросс-платформенные приложения. Существуют модули для работы с регулярными выражениями, текстовыми кодировками, мультимедийными форматами, криптографическими протоколами, архивами, для сериализации данных, поддержки юнит-тестирования и др.

Помимо стандартной библиотеки, существует множество библиотек, предоставляющих интерфейс ко всем системным вызовам на разных платформах; в частности, на платформе Win32 поддерживаются все вызовы Win32 API, а также COM, в объеме не меньшем, чем у Visual Basic или Delphi. Количество прикладных библиотек для Python в разных областях без преувеличения огромно (веб, базы данных, обработка изображений и текста, численные методы, приложения операционной системы и т. д.).

Библиотека NumPy для работы с многомерными массивами позволяет достичь производительности научных расчетов, сравнимой со специализированными пакетами. SciPy использует NumPy и предоставляет доступ к обширному спектру математических алгоритмов (матричная алгебра — BLAS уровней 1–3, LAPACK, БПФ и т. п.). Numarray специально разработан для операций с большими объемами научных данных.

Python предоставляет простой и удобный программный интерфейс C API для написания собственных модулей на языках Си и C++. Такой инструмент, как SWIG, позволяет почти автоматически получать привязки для использования C/C++ библиотек в коде на этом языке. Возможности этого и других инструментов варьируются от автоматической генерации (C/C++/Fortran)-Python интерфейсов по специальным файлам (SWIG, pyste, SIP, pyfort) до предоставления более удобных API (boost::python, CXX и др.). Инструмент стандартной библиотеки ctypes позволяет программам Python напрямую обращаться к динамическим библиотекам/DLL, написанным на Си. Существуют модули, позволяющие встраивать код на C/C++ прямо в исходные файлы Python, создавая расширения «на лету».

С Python поставляется библиотека tkinter на основе Tcl/Tk для создания кросс-платформенных программ с графическим интерфейсом.

Существуют расширения, позволяющие использовать все основные GUI библиотеки, — wxPython, основанное на библиотеке wxWidgets, PyGTK для Gtk, PyQt и PySide для Qt и др. Некоторые из них также предоставляют широкие возможности по работе с базами данных, графикой и сетями, используя все возможности библиотеки, на которой они основаны.

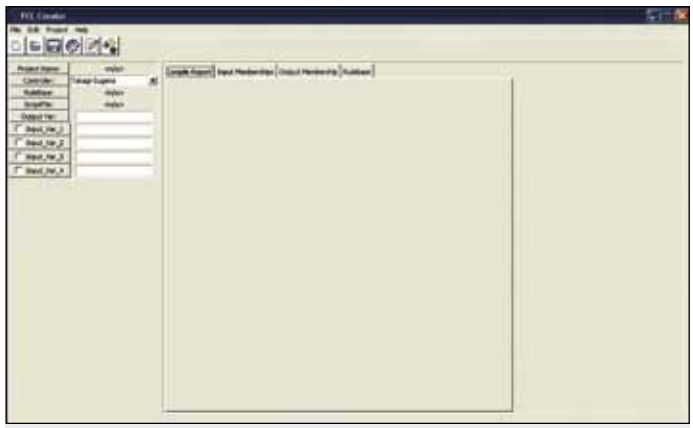

Рис. 10. Общий вид интерфейса IDE

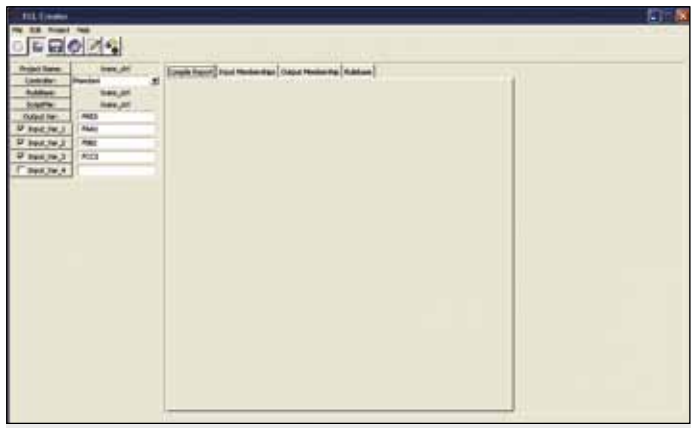

Рис. 11. Пример создания проекта

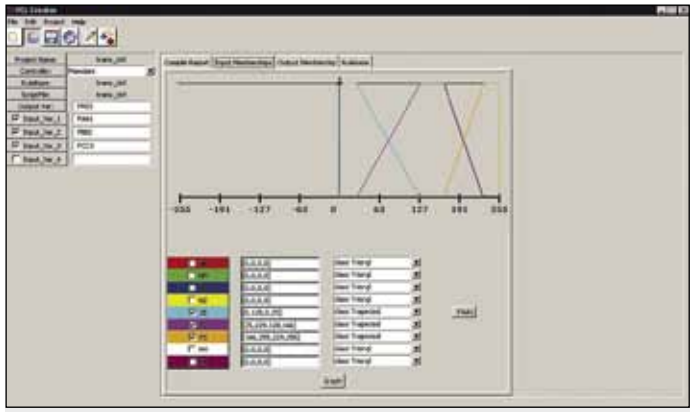

Рис. 12. Пример формирования функций принадлежности входных сигналов

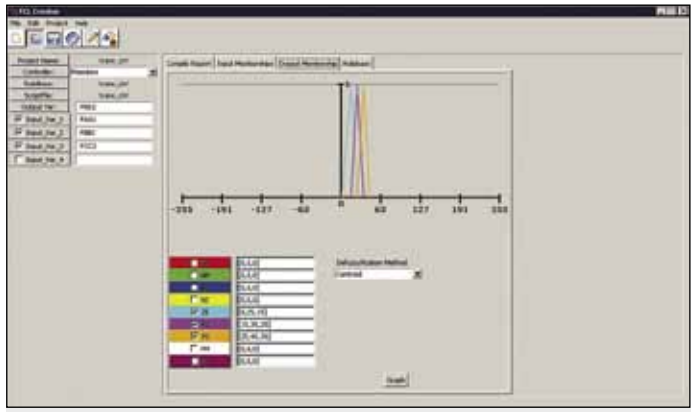

Рис. 13. Пример установки параметров для выходной переменной

Рис. 14. Пример формирования базы правил

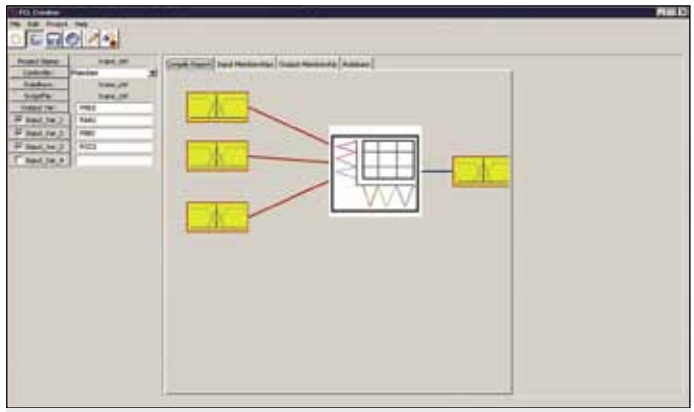

Рис. 15. Завершающая стадия работы с проектом

Для создания игр и приложений, требующих нестандартного интерфейса, можно использовать библиотеку Pygame. Она также имеет обширные средства работы с мультимедиа: с ее помощью можно управлять звуком и изображениями, воспроизводить видео [3, 4].

Итак, Python позволяет с гораздо меньшими затратами труда и времени разрабатывать IDE, быстро создавать как прототипы программных систем, так и сами программные системы, помогает в интеграции программного обеспечения для решения производственных задач. Python свободно доступен для многих платформ, а написанные на нем программы обычно переносятся между платформами без изменений. Это обстоятельство позволяет применять любую имеющуюся аппаратную платформу.

Для написания данной IDE помимо стандартной библиотеки Python использовалась графическая библиотека tkinter на основе Tcl/Tk и Pmw. Общий вид интерфейса представлен на рис. 10.

Результатом работы IDE является создание трех файлов, в которых содержатся функции для конфигурирования нечеткого контроллера и работы с ним на языке Си. Эти три файла можно непосредственно встраивать в рабочий проект. Стоит отметить тот факт, что функции обмена данными с ПЛИС являются аппаратно зависимыми и определяются типом микроконтроллера. Эти функции встраиваются в ядро RTOS, что упрощает их дальнейшее использование в рабочем проекте.

Теперь рассмотрим процесс программирования нечеткого контроллера на примере упомянутой выше задачи [2]. Сделаем проект в IDE для нечеткого сопроцессора:

- На первом этапе создаем проект и выбираем тип контроллера (рис. 11).
- Далее формируем функции принадлежности входных сигналов (рис. 12).

```
void Task1 (void)
{
SetTimerTask(Task2,2);      //Запускаем вторую задачу
    GPIOC->BSRR |= 0x00000200;      // Зажигаем диодик

//Читаем результат из нечеткого контроллера
fuzzy_result  = ReadFromAddress(result_value);

}

vnid Task2 (void)
{
SetTimerTask(Task1,2);      // Запускаем первую задачу
    GPIOC->BRR  |= 0x00000200;  // Гасим диод.


//Записываем входные значения в нечеткий контроллер
WriteToAddress(X1, input_1_value);
WriteToAddress(X2, input_2_value);
WriteToAddress(X3, input_3_value);

}
```

Листинг. Пример вызова функций работы с нечетким сопроцессором

- Формируем функции принадлежности выходного сигнала и выбираем метод дефаззификации (рис. 13).
- После этого формируем базу правил (рис. 14).
- Затем проверяем проект на ошибки (рис. 15).

Если в проекте не обнаружено ошибок, то формируются файлы с Си-кодом, в которых содержатся функции для конфигурирования нечеткого сопроцессора и управления им:

- *fuzzy_core_functions.h* — содержит прототипы функций конфигурирования нечеткого сопроцессора и макросы.
- *fuzzy_core_functions.c* — содержит функции конфигурирования нечеткого сопроцессора.
- *fuzzy_controller.c* — содержит функции управления нечетким сопроцессором.

Дальнейшие действия сводятся к тому, что эти файлы копируются в папку с проектом для микроконтроллера, а именно в подпапку с исходными кодами ядра RTOS. Конфигурирование нечеткого сопроцессора происходит при старте ядра RTOS совместно с остальной периферией микроконтроллера.

Пример вызова функции для записи входного значения в нечеткий сопроцессор показан в листинге.

Заключение

Судя по приведенному примеру построения нечеткого контроллера, применение RTOS и языка Python позволяет создавать системы различного назначения. Такое свойство языка Python, как кросс-платформенность, расширяет возможности его применения. Используя возможности языка Python и таких мощных систем моделирования, как MATLAB, можно эффективно проектировать и моделировать сложные системы.

Литература

1. Круглов В. В., Дли М. И., Голунов Р. Ю. Нечеткая логика и искусственные нейронные сети. М.: Физматлит, 2001.
2. Шин М., Парк Ч., Ким Й. Защита силового трансформатора на основе нечеткой логики // Релейщик. 2012. № 2.
3. http://ru.wikipedia.org/wiki/Python
4. www.python.org