РОССИЙСКАЯ АКАДЕМИЯ НАУК

Институт проблем управления

На правах рукописи

Выхованец Валерий Святославович

УДК 519.714 + 681.3

КРАТНЫЕ ЛОГИЧЕСКИЕ ВЫЧИСЛЕНИЯ И ИХ ПРИМЕНЕНИЕ ПРИ МОДЕЛИРОВАНИИ ДИСКРЕТНЫХ ОБЪЕКТОВ

Специальность: 05.13.13 – Вычислительные машины, комплексы, системы и сети.

05.13.16 – Применение вычислительной техники, математического моделирования и математических методов в научных исследованиях

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель,
доктор технических наук, профессор В.Д. Малюгин

Москва, 1998

Оглавление

Для таких задач, как управление сложными объектами, обработка изображений и распознавание образов, принятие решений, дискретная оптимизация, моделирование дискретных объектов, робототехника – основная проблема сводится к обработке бинарных и многозначных (логических) данных. При этом бинарные данные кодируются нулем и единицей, а многозначные – целыми положительными числами. Если учесть, что подавляющее большинство современных средств вычислительной техники представляют собой дискретные устройства, которые выполняют ввод, хранение, обработку и вывод данных, изменяющихся по закону дискретных функций, то становится ясно, что логическое вычисление занимает центральное место на всех уровнях архитектурной иерархии вычислительных средств.

Наиболее часто обработка логических данных представляется через некоторую функционально полную систему операций. Известно, что через суперпозицию элементарных функций (операций) в алгебре k–значной логики можно представить функции произвольной сложности. В результате такого представления отпадает необходимость хранения больших массивов данных, поскольку применение конечной последовательности операций к логическим переменным позволяет реализовать логическую обработку с большей эффективностью. Заметим, что не всякая форма представления системы логических функций позволяет добиться эффективной реализации. Для оценки эффективности в этом случае используется комплексный критерий, носящий название "память/быстродействие".

Таким образом, обработка логических данных сводится к эффективному вычислению (реализации) логических функций. Процесс нахождения значения логической функции или системы логических функции на наборе аргументов принято называть логическим

вычислением. Следует заметить, что традиционная архитектура ЭВМ не предусматривает специализированных устройств для логических вычислений. Поддержка логической обработки сводится к логическим (булевым) операциям, выполняемым арифметико-логическим устройством (АЛУ) – на аппаратном уровне, и к использованию команд условного перехода – на уровне программ.

Структура математического аппарата, описывающего логические вычисления, достаточно сложна и определяется, в первую очередь, большими размерностями рассматриваемых пространств и непредсказуемостью поведения дискретных объектов в окрестности некоторой точки этого пространства. Поэтому совершенствование математических методов идет в направлении создания моделей как для принципиально новых, так и для более сложных по размерности объектов. При этом увеличение сложности решаемых задач сопровождается совершенствованием технических средств.

Постоянное и повсеместное применение аппарата булевой алгебры создает впечатление его единственности и завершенности. Тем не менее, предпринимаются большие усилия по развитию исходной алгебры Буля. Не касаясь сугубо математических работ, которые связаны с именами Маркова А.А., Мальцева А.И., Сикорского Р., Биркгофа Г., Гёделя К., Поста Е.Л., Клини С.К., Рассела Б., Чёрча А. и др., рассмотрим прикладное направление математической логики, заложенное трудами Шестакова В.И., Жегалкина И.И., Гаврилова М.А., Шеннона К.

Основополагающее значение в исследовании дискретных систем имеют работы таких ученых, как Ляпунов А.А., Лупанов А.А., Яблонский С.В., Глушков В.М., Журавлев Ю.И., Поспелов Д.А., Закревский А.Д., Варшавскмй В.И., Горбатов В.А., Лазарев В.Г., Баранов С.И., Капитонова Ю.В., Летичевский А.А., Шоломов Л.А., Якубайтис Э.А., Кухарев Г.А., Шмерко В.П., Шалыто А.А., Кондратьев В.Н., Карповский М.Т., Москалев Э.С., Лабунец В.Г.,

Ситников О.П., Малюгин В.Д. и др.

Расширение булевой алгебры осуществлялось путем введения таких понятий как время (непримитивные схемы [1], рекуррентные и временные функции [2]), производные [3], секвенции [4], арифметические и псевдобулевые формы [5], k-значный алфавит [6, 7], пороговая логика [8], логические уравнения [9, 10], спектральное представление [11, 12], функциональная декомпозиция [12], логические свойства непрерывных функций (R-функций) [13], арифметическое представление [14], полиномиальные формы [15], алгебра кортежей [16], линейные арифметические полиномы [17], параллельная логика [18], обобщенные дизъюнктивные формы [19], булевы и операторные программы [20], нечеткая логика [21, 22], порядковая логика [23].

Эффективная реализация логического вычисления связана с проблемой распараллеливания. Рост сложности моделируемых объектов и увеличение размерностей вычислительных задач делает эту проблему особо важной. Стоит обратить внимание на то обстоятельство, что проблема параллельного логического вычисления остается периферийной на различных архитектурных уровнях вычислительных средств, а встречающиеся поразрядные логические операции рассматриваются как вспомогательные.

Так в современных вычислительных средствах распараллеливание обычно осуществялется: на уровне программ (параллельные процессы в многопроцессорной системе), на уровне машинных команд (матричный процессор, процессор с многими АЛУ), на уровне микроопераций (процессор с конвейеризацией команд и конвейеризацией данных), на уровне аппаратных средств (АЛУ с ускоренным переносом и поразрядными логическими операциями, средства параллельного микропрограммирования).

Наименее изученными остаются вопросы, связанные с разновидностью параллельных вычислений – кратными логическими

вычислениями. Под кратными логическими вычислениями будем понимать процесс нахождения значений логической функции (системы логических функций) на разных наборах аргументов. Кратные вычисления возникают при параллельном управлении несколькими объектами, находящимися в разных физических состояниях. Другой важный случай -- воспроизведение системы на нескольких наборах входных воздействий, а также решение логических уравнений. Кратные вычисления необходимы и в случае неопределенности или недостоверности исходных данных, когда, с целью принятия правильного решения, выполняются некоторые вычисления для множества возможных (вероятных) значений.

В булевой алгебре отсутствуют средства явного описания кратных вычислений. Подразумевается, что кратные вычисления сводятся к повторению вычисления известными методами при измененных наборах переменных. С учетом последующей технической реализации это зачастую оказывается неудобно, ибо приходится каждое значение системы функций вычислять отдельно, с большими затратами времени и оборудования.

Если при аппаратно-схемных методах синтеза дискретных устройств решение может быть достигнуто путем дублирования соответствующего оборудования и параллельном вычислении системы логических функций для каждого набора переменных, то при программной реализации кратных вычислений приемлимые решения слабо представлены в теоретических исследованиях. Наиболее распространенным является метод повторения вычислений при измененых значениях переменных.

Решением указанной проблемы могла бы быть разработка эффективных методов кратных вычисления и создание устройств, реализующих эти методы.

Целью данного исследования является:

1) разработка средств формального описания кратных вычислений;

2) обоснование методов реализации кратных вычислений на основе

используемого математического аппарата и исследование их эффективности;

3) реализация кратных вычислений на различных уровнях современных вычислительных средств.

Диссертация состоит из введения, пяти глав, заключения, списка цитируемой литературы и приложений. Нумерация формул, рисунков, таблиц и литературные ссылки выполнены по главам.

В первой главе на основе обзора основных работ исследуется место и роль логической обработки данных. Логическая обработка разделяется на задачи синтеза, моделирования, анализа и вычисления и может быть, в свою очередь, сведена к логическому синтезу и кратным логическим вычислениям. Под кратными вычислениями понимается такая организация вычислительного процесса, когда система функций вычисляется сразу на нескольких наборах аргументов. Обосновывается классификация методов структурной организации кратных вычислений, в основу которой положены принципы повторности вычислений: повторность во времени и в пространстве, параллельная и последовательная повторность. Рассматриваются формы описания логических данных, дается формальная интерпретация различных типов логических вычислений и анализируются случаи возникновения кратных вычислений. Поставлены задачи формального описания кратных вычислений и их эффективной реализации.

Во второй главе показано, что кратные вычисления могут быть описаны в алгебре поразрядных логических операций и реализованы путем вычисления арифметико-логического полинома. Для определения эффективности кратных вычислений оценивается диапазон изменения коэффициентов полинома и вводятся ее информационные характеристики: емкость и энтропия. Рассмотрены эффективные процедуры кратных вычислений, для которых операционным элементом является арифметико-логическое устройство, а аппаратные затраты определяются информа-

ционной емкостью и энтропией вычисляемого полинома.

В третьей главе рассматриваются формы и методы кратных вычислений в k-значной логике. Исследуется обобщенная полиномиальная форма и показано ее использование для эффективного представления кратных вычислений. Рассмотрен случай, когда кратность вычислений изменяется и найдена форма, коэффициенты которой не зависят от кратности. Показана необходимость использования абсолютных критериев эффективности, определяемых по отношению к табличному способу реализации логических функций. Проверка эффективности кратных вычислений осуществляется путем сравнения сложности полинома (количества ненулевых коэффициентов) с эффективной сложностью, задающей ее граничное значение. В результате вычислительного эксперимента исследована зависимость эффективной сложности от значности логики и количества независимых переменных, оценено количество полиномов, имеющих эффективную реализацию.

В четвертой главе рассматриваются аналитические методы синтеза полиномиальных форм, основанные на дискретном ортогональном преобразовании. Это позволяет получать формы в расширенном базисе степенных и логических операций. С целью упрощения формирования базиса дискретного преобразования, рассмотрен синтез мультипликативных форм, основанный на свойстве обращаемости кронекеровского произведения матриц. Для учета потребностей практики рассмотрен синтез полиномиальных форм с различной значностью переменных и функций. Исследована эффективность аналитической конструкции и для произвольной таблицы логических данных получены эффективные полиномиальные формы.

Пятая, заключительная глава посвящена вопросам реализации кратных вычислений. Описывается разработанный комплекс программ для логических исследований, предназначенный для поиска форм представления логических данных. Рассмотрен язык логического программирова-

ния, позволяющий связать два подхода к синтезу форм: символьный и матричный. Разработаны принципы программной реализации кратных вычислений с привязкой к различным этапам проектирования прикладной программы. Приведены структуры логических процессорных элементов и показана реализация на их основе различных методов структурной организации кратных вычислений. Предложена архитектура процессора, выполняющего кратные вычисления с наибольшей эффективностью.

В заключении подводится итог выполненных исследований и делаются выводы по работе в целом.

Методы исследования. Выполненная работа основана на методах булевой алгебры и алгебры k-значной логики, спектральной теории логических функций, теории конечных атоматов. При проведении вычислительного эксперимента применялись методы структурного и объектно-ориентированного программирования, теории синтаксического анализа и компиляции.

Научная новизна диссертационной работы заключается в разработке, обосновании и исследовании методов параллельных логических вычислений на основе обобщенной методики синтеза полиномиальных форм в расширенном базисе операций и при смешанной значности переменных и функций, в ориентации на эффективную реализацию кратных вычислений на различных уровнях конвейерных и параллельных вычислительных средств.

Практическая ценность полученных результатов состоит в эффективной реализации логических вычислений на основе использования комплекса программ для логических исследований, библиотеки классов для объектно-ориентированной среды программирования и структур логических процессорных элементов.

Внедрение результатов исследования проводилось в рамках научно-исследовательских работ и опытно-конструкторских разработок, выполненных в 1993-1997 гг. в научно-исследовательской лаборатории

"Математическое моделирование" Тираспольского университета по теме № 03–93 и в 1995–1997 гг. в 31 лаборатории Института проблем управления РАН (г. Москва) по теме № 375–95/31. Результаты исследования приняты к реализации в АО "Электромаш" (г. Тирасполь) и используются в учебном процессе.

Апробация работы. Основные положения диссертации докладывались и обсуждались на научно–технических конференциях профессорско–преподавательского состава Тираспольского университета (1994–1997 гг.), на расширенных семинарах 31 лаборатории Института проблем управления (1995–1997 гг.), Всероссийской конференции "Информационно–управляющие системы и специализированные вычислительные устройства для обработки и передачи данных" (Махачкала, 1996 г.), на семинаре кафедры ИУ–3 МГТУ им. Н.Э. Баумана (1997 г.).

Публикации. Непосредственно по теме диссертации опубликовано 6 работ. При изложении материала сделаны ссылки на следующие работы автора: [1, 2] – к главе 2, [1, 2] – к главе 3, [5] – к главе 4, [12] – к главе 5. Все статьи написаны и опубликованы самостоятельно. В статье [1] к главе 3 автору принадлежит разработка вопросов, связанных с кратными вычислениями, в статье [10] к главе 5 – способ взаимодействия операционных устройств, выполняющих кратные вычисления.

Глава 1. Логическая обработка данных

Логическая обработка, преобразование и хранение логических данных занимает центральное место на всех уровнях архитектуры вычислительных средств. Роль логической обработки существенно возросла в связи с широким внедрением в практику автоматизированных систем, реализующих в том или ином виде технологии получения и накопления данных и знаний, адаптивных робототехнических комплексов, систем технического зрения.

Основу логической обработки данных составляет логическое вычисление. Под логическим вычислением будем понимать процесс нахождения значения системы логических функции на наборах ее аргументов. Саму систему логических функций будем рассматривать как совокупность функций алгебры логики, заданную на общем множестве независимых переменных и представленную в одной из возможных форм. Среди логических функций будем выделять отдельный класс – булевы (двузначные) функции, заданные в булевой алгебре. В свою очередь, логические данные будем представлять как натуральные числа ограниченного диапазона, определяемого значностью алгебры логики, и разделять на бинарные (двузначные) и многозначные.

Реализация логических вычислений определяется как формами представления логических данных, так и архитектурой используемых для этого вычислительных средств. Поэтому предварительно целесообразно уяснить место, роль и особенности логических вычислений в технических системах.

§ 1. Логическая обработка в технических системах

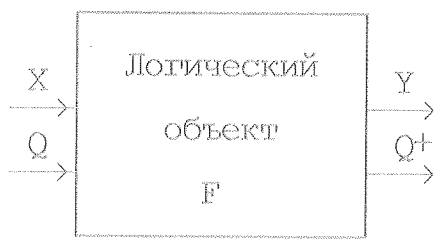
Современное состояние информационной технологии характеризуется принципиальным изменением роли логических вычислений. Они являются доминирующими в логических задачах и определяющими, например, при анализе и классификации объектов – обязательном этапе функционирования систем принятия решений, логического управления, технического зрения (рис. 1.1). Хотя логическое вычисление, преобразование и хранение логических данных занимает центральное место на всех уровнях архитертуры вычислительных средств, традиционные вычислительные средства не содержат специализированных устройств для логических вычислений [1].

Рис. 1.1. Логическая обработка данных в прикладных задачах

Наибольшее распространение в настоящее время получили вычислительные устройства дискретного действия. Современные электронные вычислительные машины (ЭВМ) являются дискретными и рассматриваются как универсальные моделирующие устройства. Техника дискретной обработки достигла такого уровня развития, который позволяет решать многие практические задачи. Развитые в последнее время методы дискретного анализа позволяют получать численный результат с заданной точностью с помощью выполнения некоторой конечной последовательности операций над дискретными данными.

Основным математическим аппаратом, используемом при проектировании вычислительных устройств дискретного действия является аппарат алгебры логики и теории автоматов, а использование этих устройств вынуждает исследователей сводить решаемую задачу, в конечном итоге, к логической обработке данных.

На основе конечно-автоматного представления дадим классификацию задач логической обработки. Основными видами логических задач в этом случае являются логический синтез, логическое моделирование, логический анализ и логическое вычисление (рис. 1.2). Под логическим объектом будем понимать некоторый дискретный детерминированный статический или динамический объект (конечный автомат, процесс) с конечным числом внутренних состояний.



Синтез	$(X, Q, Y) \rightarrow F$
Моделирование	$(X, Y) \rightarrow (F, Q)$
Анализ	$(F, Q, Y) \rightarrow (X, Q)$
Вычисление	$(X, Q, F) \rightarrow (Y, Q)$

Рис. 1.2. Основные виды логических задач

Решение логических задач может быть формализовано средствами алгебры многозначной логики и представлено в виде системы

логических функций

$$\left[\begin{array}{c} Y \\ \hline Q^{+} \end{array}\right] = F\left(\left[\begin{array}{c} X \\ \hline Q \end{array}\right]\right), \quad \text{или} \quad \begin{cases} Y = F_1(X, Q); \\ Q^{+} = F_2(X, Q), \end{cases} \qquad (1.1)$$

где Y – вектор наблюдаемых значений системы логических функций (реакция логического объекта), X – вектор независимых переменных (входное воздействие на логический объект), Q – вектор внутренних переменных (состояние логического объекта), Q^{+} – вычисляемое следующее состояние логического объекта; F – система логических функций (логический объект), которая может быть представлена в любой из известных форм: в виде таблицы, в виде выражений в алгебре k-значной логики, в логической, арифметической или арифметико-логической форме и т.д., F_1 – функции выхода, F_2 – функции переходов.

На практике, логическая обработка представляется в виде системы булевых функций в результате кодирования значений функций и переменных двоичными числами.

Логическое вычисление. При логическом вычислении решается задача получения реакции логического объекта, находящегося в некотором состоянии Q, на заданное входное воздействие X, или, что то же самое, вычисление системы логических функций на заданных наборах независимых переменных.

К задачам логического вычисления сводится подавляющее большинство прикладных задач логической обработки. В конечном итоге под логической обработкой данных часто понимается ее реализация посредством логического вычисления. Если заданы несколько наборов переменных, то решение задачи сводится к повторению вычисления системы функций на измененных наборах.

Логический анализ. К логическому анализу относятся задачи, сводимые к решению систем логических уравнений и позволяющие по реакции Y или по реакции Y и состоянию Q логического объекта

определить возможные его состояния Q и входные воздействия X.

К логическому анализу сводится большой класс научных и технических задач, среди которых анализ дискретных автоматов, их контроль и диагностика, принятие решений в экспертных системах, и т.д. Наиболее жесткие требования к средствам решения логических уравнений предъявляются в системах логического управления, в гибких автоматизированных производствах, в системах технического зрения, что связано с функционированием этих систем в реальном времени.

Характерной особенностью задач логического анализа – большая вычислительная сложность их решения, связанная с переборным принципом поиска неизвестных, а сокращение числа переборов приводит к значительному усложнению алгоритма [5]. Например известен метод логического анализа, основанный на тождественных преобразованиях формул. Решение задач логического анализа наиболее часто сводится к повторению вычислений системы на всевозможных наборах переменных и сравнению полученных результатов с искомым.

Логический синтез. При логическом синтезе решается задача получения формального описания системы (1.1), когда известны все состояния логического объекта Q и реакция Y на каждое входное воздействие X для каждого возможного состояния Q.

Задачи логического синтеза возникают при проектировании дискретных устройств и служат основой для их реализации в виде схем из логических элементов. Необходимость в формализованном описании логического объекта вызвана тем, что известные методы синтеза дискретных устройств в том или ином виде повторяют полученные при этом формульные выражения [2, 3]. Проверка адекватности полученных формул осуществляется путем их вычисления на множестве наборов переменных.

Логическое моделирование. При анализе поведения логических объектов часто возникает задача распознавания тех или иных свойств

изучаемого логического объекта в процессе экспериментов с ним. Эксперимент в этом случае заключается в подаче на вход логического объекта различных наборов входных переменных и наблюдении за его реакцией. При проведении таких экспериментов характерной является задача расшифровки внутреннего устройства логического объекта.

Постановка задачи логического моделирования может быть сведена к получению системы (1.1) по известным воздействиям X и реакции Y логического объекта на эти воздействия. В связи с тем, что логический объект может находится в одном из конечного числа состояний, необходимо иметь серию опытов $(\hat{X}_i, \hat{Y}_i)$, $i=1, 2, \ldots, L$, откуда извлекается количество возможных состояний Q и вид функций, описывающих его функционирование. Упорядоченные пары $(\hat{X}_i, \hat{Y}_i)$ являются результатом L наблюдений за логическим объектом и могут быть получены в условиях активного или пассивного эксперимента.

При проведении экспериментов с логическим объектом возникают частные задачи, связанные с эффективным планированием активного эксперимента и обработкой экспериментальных данных в условиях пассивного эксперимента. Эффективное планирование активного эксперимента заключается в таком проведении эксперимента, когда необходимая информация возникает с наименьшими затратами.

Теоретические основы кратных экспериментов с конечными автоматами подробно рассмотрены в [4], где под кратными экспериментами понимаются такие эксперименты, когда возможна одновременная подача различных входных наборов переменных на различные экземпляры одного и того же атомата. Кратные эксперименты возникают также при наличии у исследуемого автомата "кнопки возврата", переводящей его в исходное состояние после подачи каждой очередной последовательности наборов переменных.

При логическом моделировании по результатам наблюдений строится автоматная таблица, из которой синтезируется формульное

описание – модель объекта. Верификация модели осуществляется ее вычислением на множестве наборов переменных.

Кратные логические вычисления. Расмотренные выше логические задачи решаются путем синтеза формульного описания некоторого логического объекта и повторения вычислений полученной модели при различных (измененных) значениях независимых переменных, т.е. посредством кратных (повторных) вычислений системы (1.1). Таким образом, логическая обработка данных может быть сведена к логическому синтезу и кратным логическим вычислениям.

Под кратными логическими вычислениями будем понимать процесс нахождения значений одной или системы логических функций сразу на множестве наборов ее аргументов. Величина кратности, очевидно, лежит в диапазоне от 1 до 2^n, где n – количество независимых переменных, что означает, соответственно однократные и полнократные вычисления.

Кратные логические вычисления обладают рядом специфических особенностей, которые не позволяют в большинстве случаев воспользоваться известными методами и приемами вычислений. Независимость функций в системе логических функций, отсутствие у них своего рода "непрерывности", когда наблюдаются "малые" изменения функции при "малом" изменении независимых переменных, превышение числа функций по сравнению с количеством допустимых параллельных каналов вычислительного средства, неэффективность использования многоразрядных аппаратных средств и т.п.

Основой известных реализаций кратных вычислений служит однократный случай, в связи с чем рассмотрим особенности вычисления логических функций на одном наборе переменных.

§ 2. Методы реализации логических вычислений

Реализация логического вычисления может быть выполнена вычислительными средствами дискретного действия. Под вычислительными средствами дискретного действия обычно понимаются устройства, работающие с дискретными входными и выходными сигналами в дискретном времени. Как известно [6], проектирование дискретных устройств связано с решением двух основных задач: созданием компактного описания, отражающего функционирование системы (задача логического синтеза) и построением наиболее экономичной реализации этого описания, при условии выполнения требований по заданным критериям качества реализации.

Аппаратная реализация. Первой и наиболее распространенной реализацией логического вычисления является его реализация в виде схемы из логических элементов (в виде комбинационной схемы) [7, 8].

Если задана некоторая функция f формулой Ф над множеством функций $\Omega=\{\varphi_1, \ldots, \varphi_p\}$, то формуле Ф можно поставить в соответствие схему Г из логических элементов Ω. Построение схемы Г определяется индукцией по глубине формулы:

1) если $\Phi=\varphi(X_{i_1}, \ldots, X_{i_m})$, где $\varphi\in\Omega$, $X_{i_1}, \ldots, X_{i_m}$ – независимые переменные, то схема Г состоит из одного элемента φ, входы и выходы которого отождествлены с входами и выходами комбинационной схемы Г;

2) если $\Phi=\varphi(\Phi_1, \ldots, \Phi_m)$, где Φ_i – переменная X_j или функция, уже реализованная схемой Γ_i, то схема Г строится так: к i-му входу элемента φ присоединяется выход схемы Γ_i (если Φ_i – функция) или входная переменная (если Φ_i равно X_j); выходом схемы Г объявляется выход логического элемента φ.

Схема, полученная описанным методом, имеет вид ориентированного дерева: ее входы соответствуют концевым вершинам,

а выход – корню дерева. Между множеством формул над Ω и множеством древовидных схем из логических элементов, реализующих функции Ω, существует взаимно однозначное соответствие. Число знаков операций в формуле Φ равно числу элементов в Γ. Отсюда получаем важный, хотя и очевидный факт: для того чтобы произвольная функция могла быть реализована схемой в базисе Ω, необходимо и достаточно, чтобы множество функций Ω было функционально полным. Ясно, что для реализации системы функций ответ в точности тот же.

Проблема получения описания логической функции в виде конечных формул в используемом функционально полном базисе сводится к разложению системы логических функций, заданной, как правило, в виде таблицы истинности, в суперпозицию (композицию) некоторых других более простых функций (формул). При этом сравнение формул можно осуществлять по количеству используемых знаков переменных и/или операций. Такая задача называется декомпозицией логической функции [9].

Так функцию $f(X_1, \ldots, X_n)$ от n переменных можно выразить как функцию от других функций g и h

$$f(X_1, \ldots, X_n)=g(h(Y_1, \ldots, Y_s), Z_1, \ldots, Z_s),$$

где $Y=\{Y_1, Y_2, \ldots, Y_s\}$ и $Z=\{Z_1, Z_2, \ldots, Z_r\}$ – подмножества переменных $X=\{X_1, X_2, \ldots, X_n\}$ такие, что $Y \cup Z=X$. В случае $Y \cap Z=\emptyset$ декомпозиция называется непересекающейся, в отличие от пересекающейся, когда $Y \cap Z \neq \emptyset$. В табл. 1.1 приведены возможные виды декомпозиции.

Основными задачами декомпозиции являются нахождение необходимых и достаточных условий существования композиций и разработка методов декомпозиции конкретного вида. В качестве промежуточных задач укажем на такие: выбор целесообразного разбиения множества аргументов, по которым проводится декомпозиция, и построение всех возможных композиций заданного вида для выбора

лучшей.

Таблица 1.1

Наименование декомпозиции	Формальное представление
Однократная	$f(X)=g(h(Y), Z)$
Многократная	$f(X)=g_u(\ldots g_2(g_1(Y_1), Y_2) \ldots, Y_u), Z)$
Итеративная	$f(X)=g(h_1(Y), \ldots, h_t(Y), Z)$

В общем случае декомпозицию логических функций принято рассматривать как логико-комбинаторную задачу, точный алгоритм решения которой достаточно трудоемок, ибо связан с перебором большого числа решений. Поэтому для декомпозиции произвольной логической функции не найдены эффективные алгоритмы. К тому же, как показано в [9], нет хорошего соответствия между оценками сложности композиции логических функций и технической сложностью схем.

Асимптотические оценки. Известен подход к минимизации схем из логических элементов, при котором ставится задача минимизации формульного представления в заданном функционально полном базисе.

С. В. Яблонский в своей работе [10] выдвинул гипотезу о том, что всякому алгоритму, предназначенному для решения задачи синтеза минимальных форм логических функций присуща высокая трудоемкость, обусловленная принципиальной трудностью задачи логического синтеза в общем виде.

Известно очень небольшое количество классов функций, для которых найдены минимальные формы представления. Исследования в этой области дают косвенные подтверждения того, что поиск минимальных представлений невозможен без большого перебора возможных вариантов и практически нереализуем [11]; оценить по заданной функции число операций в минимальном представлении, не

проводя синтеза, также не удается.

Основной результат здесь формулируется в виде теоремы Шеннона–Лупанова [12]:

1) для любого булевого базиса Ω число операций $L_\Omega(f)$ минимальной формулы, необходимых для представления произвольной булевой функции f от n независимых переменных

$$L_\Omega(f)\sim C_\Omega 2^n/n,$$

где C_Ω – константа, зависящая от базиса;

2) для любого $\varepsilon>0$ доля функций f, для которых

$$L_\Omega(f)\le(1-\varepsilon)C_\Omega 2^n/n,$$

стремится к нулю с ростом n. Здесь знак $\sim$ обозначает асимптотическое равенство: $a(n)\sim b(n)$, если $\lim\limits_{n\to\infty}\{a(n)/b(n)\}=1$.

Смысл второго утверждения теоремы в том, что с ростом n почти все функции реализуются со сложностью, близкой к верхней границе. Теорема Шеннона–Лупанова может быть обобщена для алгебры k–значной логики. Оценка сложности представления функции в произвольном логическом базисе Ω в этом случае выглядит так:

$$L_\Omega(f)\sim C_\Omega k^n/n.$$

Таким образом, задача минимизации схемы из логических элементов приводит к наибольшему эффекту при небольшом количестве независимых переменных n. С ростом n эффективность минимизации произвольной функции резко падает, что связано с экспоненциальным ростом сложности формулы.

Программная реализация. Реализация логического вычисления схемой из логических элементов – это исторически первый и наиболее исследованный вид реализации. С развитием вычислительной техники и особенно с появлением микропроцессоров произошел массовый переход от аппаратной к программной реализации, при котором логическое вычисление отражалось не в структуру вычислительного устройства, а в структуру программы, выполняемой универсальным алгоритмическим

устройствой (рис. 1.3).

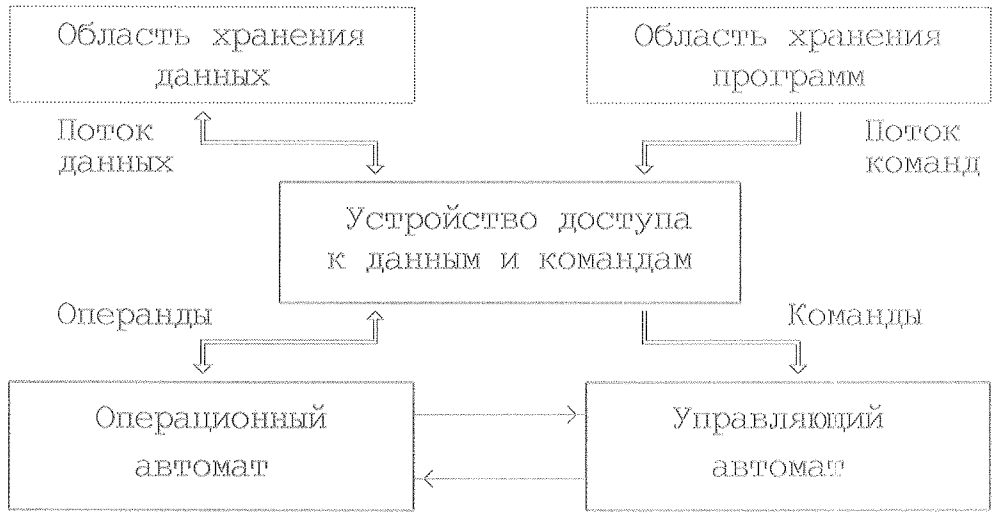


Рис. 1.3. Модель алгоритмического устройства

Под программой П будем понимать пронумерованную последовательность команд $c_1, c_2, \ldots, c_s$, взятых из некоторой фиксированной системы команд C, $c_i \in C$. Команды программы расположены в памяти программ и имеют доступ к конечному множеству пронумерованных ячеек памяти, в каждой их которых могут храниться обрабатываемые данные, принимающие значения в диапазоне от 0 до $k-1$, где k – значность алгебры логики.

Процессом вычисления программы $П=(c_1, c_2, \ldots, c_s)$ называется последовательность шагов $c(1), c(2), \ldots, c(t)$, на каждом из которых выполняется одна команда программы. Число t называется временем выполнения программы. Программа П вычисляет (реализует) логическую функцию $f(X_1, \ldots, X_n)=Y$, если для любого набора переменных $\sigma=(\sigma_1, \ldots, \sigma_n)$ при начальном сотоянии памяти $X_1=\sigma_1, \ldots, X_n=\sigma_n$ программа через конечное число шагов останавливается и при этом в ячейке Y находится величина $f(\sigma_1, \ldots, \sigma_n)$.

Если под сложностью схемы обычно понимается число логических

элементов, то сложность программы можно понимать в различных смыслах: число команд в тексте программы; объем требуемой промежуточной памяти; время вычисления, которое будет характеризоваться двумя величинами: средним временем $t_{ср}$ и максимальным временем $t_{макс}$:

$$t_{ср}(П)=\frac{1}{k^n}\sum_{\sigma}\tau_p(\sigma), \qquad t_{макс}(П)=\max_{\sigma}\tau_p(\sigma),$$

где сумма и максимум берутся по всем k^n наборам σ, а τ_p – время работы программы П на наборе σ.

Операторные программы. Если логическая функция f, которую необходимо реализовать, представлена в виде формулы Ф в некотором функционально полном базисе Ω, то ее вычисление можно представить в виде схемы из логических элементов Г. Любой схеме $Г_f$, реализующей функцию f и содержащей N логических элементов, нетрудно поставить в соответствие операторную программу, реализующую функцию f и состоящую из N команд, следующим образом. Для этого нумеруют элементы схемы числами $1, 2, \ldots, N$ так, чтобы при любом пути от входа к выходу номера элементов возрастали, при этом номер 1 получает один из входных элементов, а номер N – выходной элемент.

Пусть элемент e_i схемы Г реализует функцию φ_i и к его входам присоединены выходы элементов $e_{j_1}, \ldots, e_{j_p}$ (некоторые из них, возможно, являются входами схемы). Элементу e_i ставят в соответствие ячейку памяти a_i и команду $a_i=\varphi(a_{j_1}, \ldots, a_{j_p})$. В итоге получают программу, не содержащую условных переходов, в которых порядок команд в точности соответствует нумерации элементов в схеме, а система команд С соответствует используемому базису Ω.

Проблема синтеза операторных программ в основном сводится к проблемам синтеза схем из логических элементов: в частности, вопросы функциональной полноты системы команд и минимизации собственного текста операторной программы совпадают с задачами о

функциональной полноте базиса Ω и минимизации схемы из логических элементов.

Поскольку операторная программа не содержит команд условных переходов, время ее вычисления на любом наборе одно и то же; отсюда $t_{макс}(П)=t_{ср}(П)=N$, т.е. оба вида временной сложности совпадают со сложностью текста программы. Количество же команд в программе при достаточно больших n почти для всех реализуемых функций близко к k^n/n. Проблема минимизации памяти для операторных программ за счет многократного использования одной ячейки для нескольких промежуточных результатов является нетривиальной комбинаторной задачей [13].

Логические программы. Другой крайний вид программы, вычисляющей логическую функцию f – это программа, состоящая из команд присваивания $Y=\sigma$ и условных переходов. Такие программы для функций булевой алгебры называются бинарными, для алгебры k–значной логики назовем их логическими. Всякую логическую функцию f от n переменных можно представить в виде [14]:

$$f(X_1, \ldots, X_n)=\bigcup_{i=0}^{k^n-1} f(i)\&X_1^{i_1}\&\ldots\&X_n^{i_n}, \qquad (1.2)$$

где $\cup$ (&) – знак операций обобщенной дизъюнкции (конъюнкции); f(i) – значение функции f в точке i; $i=(i_1, i_2, \ldots, i_n)_k$ – представление числа i в n разрядной k–ичной позиционной системе счисления; $i_j\in\{0, 1, \ldots, I\}$, $(j=\overline{1, n})$ – k–ичные цифры числа i; X_i^j – характеристическая (степенная) функция, являющаяся обобщением некоторых свойств отрицания и определяемая как

$$X_i^j=\begin{cases} 0, & \text{если } X_i\neq j; \\ I, & \text{если } X_j=j. \end{cases} \qquad (1.3)$$

Здесь I=k–1 – логическая единица алгебры. Выражение (1.2) представляет собой аналог совершенной дизъюнктивной нормальной формы для функции k–значной логики, где в качестве операций

конъюнкции и дизъюнкции используются соответственно двуместные операции максимум и минимум.

Если глубина полученной формулы Ф относительно степенных операций не более чем d_f, $d \le (k-1)n_f$, то функцию f можно реализовать логической программой Π_f, вычисляющей Ф за время $t_{макс}(\Pi) = d+1$. Количество команд условного перехода в полученной логической программе будет равно общему количеству степенных операций в формуле Φ_f, а также дополнительно не более чем k команд присваивания $Y=\sigma$. Для описания логических программ используют представление программы в виде графа, в котором вершины соответствуют командам, а ребра – переходам (рис. 1.4).

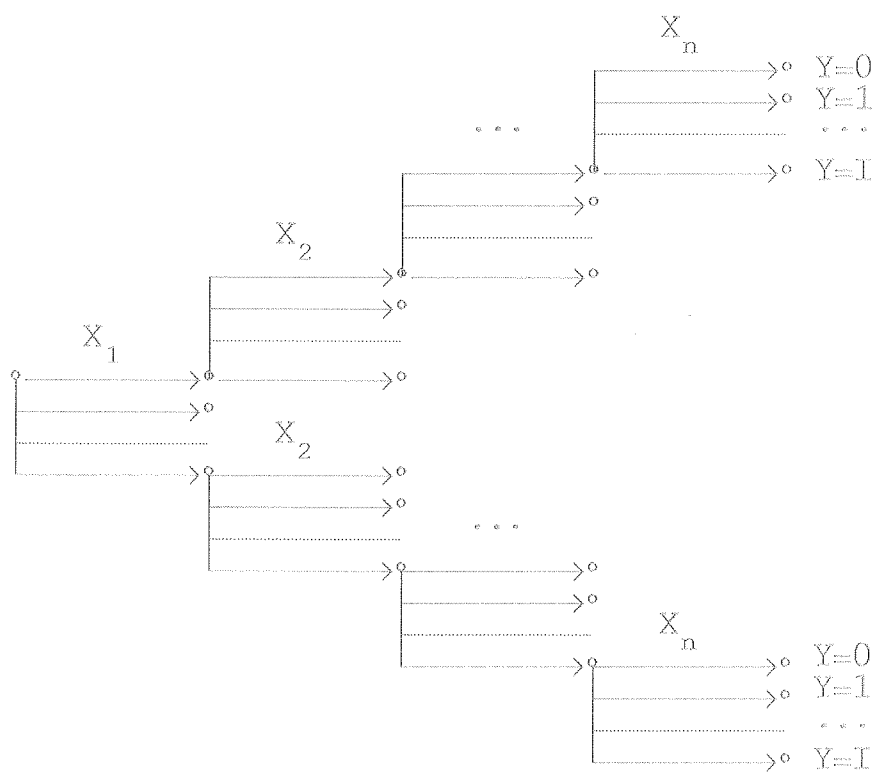


Рис. 1.4. Обобщенная граф-схема логической программы

Описанный метод не гарантирует минимальность полученных программ по числу команд. В общем случае сложность логических программ характеризуется асимптотическими оценками. По аналогии с

оценкой числа команд в бинарной программе функцией Шеннона $L(n)\sim 2^n/n$ [13] можно получить следующую оценку числа команд в логической программе:

1) $L(n)\sim k^n/n$, причем существует метод синтеза логических программ, удовлетворяющий этой оценке и для которых $t_{макс}\sim(k-1)n$;

2) для любого $\varepsilon>0$ доля логических функций f, для которых $L(f)\le(1-\varepsilon)k^n/n$ и $t_{ср}(f)\le(1-\varepsilon)(k-1)n$, стремиться к 0 при $n\to\infty$.

Таким образом, сложность логических программ по числу команд асимптотически равна сложности операторных программ.

Смешанная система команд. Для задач логического вычисления система команд C может быть получена на основе разложения логической функции по произвольной переменной. Пусть задана логическая функция алгебры k-значной логики от n независимых переменных $f(X_1, \ldots, X_i, \ldots, X_n)$. Можно показать, что эта функция может быть представлена в виде разложения по переменной X_i:

$$f(X_1,\ldots, X_i,\ldots, X_n)=\bigcup_{\sigma=0}^{I} f(X_1, \ldots, \sigma, \ldots, X_n)\&X_i^{\sigma},$$

где степенная функция X_i^{σ} определена в соответствии с (1.3).

Выражение $f(X_1, \ldots, \sigma, \ldots, X_n)$ при различных $\sigma=\overline{0,I}$ можно рассматривать как k различных функций от n-1 переменной. Для реализации вычисления функции f достаточно реализовать систему команд C, включающую команды двух видов:

1) $b=\varphi(a_1, \ldots, a_p)$: ячейке b присвоить значение φ или выполнить операцию φ над содержимым ячеек $a_1, \ldots, a_p$ и результат положить в ячейку b; ячейка b называется промежуточной ячейкой памяти;

2) $jump(a_i, j, u, w)$: если a_i не равно j, то перейти к выполнению команды c_u, иначе перейти к команде c_w.

Операция φ, реализуемая командой первого типа, – это логическая функция p переменных, в частности она может быть любой

из констант алгебры логики. Команда второго типа – команда условного перехода – реализует степенную функцию. Если u – номер следующей команды, то переход можно считать одноадресным; если u равно w, то безусловным.

Ранее было показано, что сложность логических программ по числу команд асимптотически равна сложности операторных программ. Если в программе использовать и операторные команды и команды условных переходов, то для булевой алгебры эта оценка понижается вдвое [15]. Это определяет использование на практике смешанной системы команд. Расширение состава команд при программной реализации логического вычисления и использование как команд условного перехода, так и операторных команд связывают с использованием возможностей микропроцессоров.

Таким образом, известные методы реализации логических вычислений своей структурой повторяют формульное описание логической функции, а основой повышения эффективности вычислений служит расширение используемого базиса операций.

§ 3. Формы представления логических функций

Форма аналитического описания (представления) логической функции во многом предопределяет эффективность решения задач логического вычисления и эффективность использования аппаратных и программных средств. Так, метод синтеза формы должен быть простым в реализации, а сама форма – удобной для формализованного решения задачи, алгоритмизации ее и отображения в вычислительную структуру или структуру программы. Рассмотрим основные формы представления логических функций.

Нормальные формы. К нормальным формам представления булевой

функции $f(X)$ от n переменных $X=(X_1, \ldots, X_n)$ относятся совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ) [16]:

$$\begin{cases} \text{СДНФ:} \quad f(X)=\bigvee\limits_{i=0}^{2^n-1} f(i)\&X_1^{i_1}\&X_2^{i_2}\&\ldots\&X_n^{i_n}, \\ \text{СКНФ:} \quad f(X)=\bigwedge\limits_{i=0}^{2^n-1} f(i)\vee X_1^{i_1}\vee X_2^{i_2}\vee\ldots\vee X_n^{i_n}, \end{cases} \tag{1.4}$$

где $\vee$ ($\&$) – знак дизъюнкции (конъюнкции); $f(i)$ – значение функции f в точке i; $i=(i_1, i_2, \ldots, i_n)_2$ – представление числа i в n разрядной двоичной позиционной системе счисления; i_j, ($j=\overline{1, n}$) – двоичные цифры числа i; X_i^j – переменная X_i в логической степени j.

Определения степенных операций для СДНФ и СКНФ задаются следующим образом:

$$\text{СДНФ:}\ X_i^j=\begin{cases} \bar{X}_i, & \text{при } j=0; \\ X_i, & \text{при } j=1, \end{cases} \qquad \text{СКНФ:}\ X_i^j=\begin{cases} \bar{X}_i, & \text{при } j=1; \\ X_i, & \text{при } j=0, \end{cases} \tag{1.5}$$

где $\bar{X}_i$ – обозначает операцию отрицания X_i.

СКНФ (СДНФ) была исторически первой формой представления булевых функций. В качестве базиса для нормальных форм используется система операций $\Omega=\{\&, \vee, \bar{}\}$. Достоинством нормальных форм является простота их синтез: коэффициентами в (1.4) являются значения функции, взятые из таблицы истинности. Минимизация нормальных форм проводится с использованием законов алгебры логики и осуществляется путем тождественых преобразований.

Логическая полиномиальная форма. На ряду с нормальными формами булевых функций нашли свое применение логические полиномиальные формы. Любая булевая функция может быть представлена в виде полинома Жегалкина [17, 18]:

$$f(X)=\sum_{i=0}^{2^n-1} a_i\&X_1^{i_1}\&\ldots\&X_n^{i_n} \pmod 2, \quad X_i^j=\begin{cases} X_i, & \text{при } j=1; \\ 1, & \text{при } j=0, \end{cases} \tag{1.6}$$

где $i=(i_1, i_2, \ldots, i_n)_2$; $a_i\in\{0, 1\}$ – коэффициенты полинома.

Базисом для логической полиномиальной формы является базис Жегалкина $\Omega=\{1, \oplus, \&\}$, где $\oplus$ – операция суммы по модулю два или неэквиваленция.

Задача построения полинома (1.6) сводится к нахождению коэффициентов a_i, $(i=\overline{0, 2^n-1})$. Для любых конституент единицы $K_i(X)$,

$$K_i(X)=X_1^{i_1}\&X_2^{i_2}\&\ldots\&X_n^{i_n}, \qquad (1.7)$$

имеет место соотношение $K_i(X)\vee K_j(X)=K_i(X)\oplus K_j(X)$. Выражение (1.7) позволяет осуществить переход от СДНФ к полиному Жегалкина. Для этого достаточно заменить в СДНФ из (1.4) знак дизъюнкции на знак неэквиваленции и выполнить подстановку вида $\bar{X}_i=1\oplus X_i$, с последующим тождественным преобразованием полученного выражения в алгебре Жегалкина.

Полиномиальная форма в поле Галуа. Известна также форма представления логической функции в поле Галуа $GF(k^n)$, где k – простое число [19, 20]. Представление логической функции от n независимых переменных в поле Галуа основано на интерпретации значений переменных и функций как элементов поля.

Роль переменной в этих полях играет примитивный корень α неприводимого многочлена степени n. Первые n степеней примитивного корня $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$ являются линейно независимыми относительно операций сложения и умножения по модулю k^n и могут служить базисом поля Галуа. Это значит, что любой элемент поля представим в виде:

$$A=a_0\alpha^0+a_1\alpha^1+\ldots+a_{n-1}\alpha^{n-1}=\sum_{i=0}^{n-1} a_i\alpha^i, \qquad (1.8)$$

где $a_i\in\{0, \ldots, k-1\}$ и любое n разрядное k–ичное число $A=(a_{n-1}, \ldots, a_0)_k$ можно рассматривать как элемент поля $GF(k^n)$.

Представление системы логических функций описывается выражением:

$$F(X)=F(0)+\sum_{i=1}^{k^n-1} G_i X^i, \quad G_i=\sum_{j=1}^{k^n-1} \alpha_j^{-i}\left[F(0)-F(\alpha_j)\right], \qquad (1.9)$$

где $F(i)$ – значение системы логических функций в точке i, представленое в виде взвешенной суммы значений логических функций f_j, ($j=\overline{1, q}$) по степеням примитивного корня α (1.8). Заметим, что полиномиальная форма в поле Галуа (1.9) позволяет совместно описать не более чем n логических функций. Практическое применение полиномиальной формы в поле Галуа связано с синтезом логических схем [21].

Арифметическая форма. Наряду с классическим аппаратом булевой алгебры для решения широкого круга задач применяется арифметическая логика и соответствующая ей арифметическая полиномиальная форма [22].

Система операций $\Omega=\{1, +, -, \times\}$, заданных на множестве натуральных чисел N является функционально полной в булевой алгебре. Действительно, функции сигнатуры булевой алгебры могут быть выражены следующим образом:

$$X\&Y=XY, \qquad X\vee Y=X+Y-XY, \qquad \overline{X}=1-X,$$

где $X, Y\in\{0, 1\}$. Это позволяет произвольную булевую функцию $f(X)$ представить в виде арифметического полинома:

$$f(X)=\sum_{i=0}^{2^n-1} a_i X_1^{i_1} X_2^{i_2}\ldots X_n^{i_n}, \qquad X_i^j=\begin{cases} X_i, & \text{при } j=1; \\ 1, & \text{при } j=0. \end{cases} \qquad (1.10)$$

В работе [23] дано обобщение арифметической формы (1.10) для случая k–значной логики:

$$f(X)=\sum_{i=0}^{k^n-1} a_i X_1^{i_1} X_2^{i_2}\ldots X_n^{i_n}, \qquad X^j=\underbrace{XX\ldots X}_{j \text{ раз}}.$$

Достоинством арифметических полиномиальных форм является возможность использовать для их реализации современные вычислительные средства, в систему команд которых всегда включаются

арифметические команды, а также возможность их обобщения на систему логических функций (см. § 4).

Спектральное представление. Логическую функцию от n переменных можно рассматривать как отображение $B_k^n \to B_k$, где B_k^n – n-кратное декартовое произведение $B_k=\{0, 1, \ldots, k-1\}$. В спектральных методах представления данное отображение заменяется отображением $R \to R$ ($Z \to Z$), где R (Z) – поле действительных (целых) чисел.

Тогда функцию $Y=f(X_1, X_2, \ldots, X_n)$, $X_i, Y \in B_k$, можно формально представить в виде $Y=f(X)$, где $X, Y \in Z$. Так как произвольная функция, как правило, не имеет аналитического описания, то остается представить ее разложением в какой-то системе базисных функций. Для этого поступают следующим образом [24, 25]. Функцию $Y=f(X)$ доопределяют на интервале $[0, 1)$ до кусочно-постоянной функции $f(X/k^n)$ следующим образом:

$$f(X/k^n)=\varphi(i), \quad i \le X < i+1, \quad i=1, k^n-1.$$

Полученная функция ограничена, имеет конечное число разрывов первого рода и представима в виде ортогонального ряда

$$f(X/k^n)=\sum_{i=0}^{k^n-1} a_i \psi_i(X/k^n), \qquad (1.11)$$

где $\{\psi_i\}$ – полная ортогональная система базисных функций.

В последнее время, в связи с интенсивным развитием цифровой вычислительной техники, внимание исследователей стали привлекать полные системы ортогональных функций Уолша, Хаара, Виленкина–Крестенсона, Виленкина–Понтрягина и др., для которых существуют быстрые вычислительные процедуры [26].

Методы синтеза дискретных устройств, в которых используется разложение в системе ортогональных функций (1.11), относят к спектральным методам [27]. Сложность устройства, реализующего

логическую функцию при спектральном представлении определяется числом ненулевых коэффициентов a_i или сложностью блока памяти для их хранения. Минимизация сложности устройства достигается посредством линейного преобразования аргументов [8].

Дискретное ортогональное преобразование. Выбор той или иной формы представления логической функции определяется простотой синтезируемого выражения. Необходимость расширения форм представления логических функций диктуется объективными обстоятельствами, обусловленными развитием вычислительной техники. В связи с этим ставится задача выбора оптимальной формы представления функции в некотором базисе операций. В зависимости от используемого базиса можно получать представления различной степени сложности, из которых выбираются наиболее приемлимые решения.

Математическим аппаратом синтеза форм представления логических функций служит аппарат дискретного преобразования Фурье (ДПФ). Теоретический и практический интерес к этому преобразованию не ослабевает, что вызвано рядом причин. Во-первых, ДПФ широко проникло в различные области науки и техники [28]. Во-вторых, это связано с необходимостью пересмотра теоретических и прикладных аспектов его реализации, поскольку изменяются концепции построения вычислительных средств [29, 30].

Пара ДПФ в базисе D_N определяется матричными соотношениями вида

$$\begin{cases} A=D_N F; \\ F=D_N^{-1} A, \end{cases} \qquad (1.12)$$

где F – характеристический вектор (отсчеты сигнала, столбец таблицы истинности функции) размерности N; A – вектор коэффициентов формы (спектр сигнала) той же размерности; D_N – квадратная матрица преобразования размерности N×N (базис преобразования). Во втором выражении матрица преобразования D_N^{-1} есть матрица, обратная D_N.

Для решения задачи синтеза новых форм логических функций применяются различные базисы ДПФ. В основе построения матрицы базиса D_N^{-1} лежит ядро D_k^{-1}, элементы $d^{-1}(i, j)$, $i, j=\overline{0, k-1}$, которого формируются по определенному правилу. По матрице D_k^{-1} (D_k) строится матрица D_N^{-1} (D_N) с помощью рекуррентного правила. Ядра и правила формирования их элементов для наиболее распространенных булевых базисов приведены в табл. 1.2 (здесь знак $\otimes$ обозначает операцию кронекеровского произведения матриц [28]).

Выбор базиса ДПФ – достаточно сложная задача. Теоретически выбор базиса, в наибольшей степени удовлетворяющего структуре сигнала (оптимальный базис), решается с помощью теоремы Карунена–Лоэва [31]. Однако использование этой теоремы связано со значительными трудностями. Поэтому на практике используется ограниченное количество базисов.

На рис. 1.5 представлены используемые классы форм логических функций [1]. Множество полиномиальных форм порождается ДПФ в классе Уолше–подобных базисов, причем изменения типа операций приводит к арифметическим либо логическим полиномиальным формам. Процедура изменения ядра значительно расширяет представления о полиномиальных формах. Так синтезируются полиномы, в которые переменные входят с инверсией или с циклическим отрицанием. Остальные классы форм являются неполиномиальными, хотя в каждый из них можно добавить некоторые полиномиальные свойства посредством выбора соответствующего базиса ДПФ. В итоге выбор оптимальной формы на практике сводится к учету особенностей функции в рамках одной из известных форм.

Применение известных форм представления для произвольной логической функции позволяет учесть достаточно сложные особенности этой функции, например, избежать применения комплексной арифметики или выполнить вычисление на основе только арифметических операций. Однако это приводит к формам, далеким от минимальных (оптимальных),

что связано с ограниченным выбором операций, через которые реализуется синтезируемое аналитическое выражение.

Таблица 1.2

Базис	Ядро (D_2^{-1}, D_2)	Рекуррентное правило	Аналитическая конструкция
Логический	$D_2^{-1}=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$	$D_{2^{t+1}}^{-1}=D_2^{-1}\otimes D_{2^t}^{-1}$ $D_{2^{t+1}}=D_2\otimes D_{2^t}$	$L(X)=\bigoplus_{i=0}^{2^n-1} l_i \& X_1^{i_1} \& \ldots \& X_n^{i_n}$
	$D_2=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$	$L=D_N F$, $F=D_N^{-1} L$	$X_i^j=\begin{cases} X_i, & j=1;\\ 1, & j=0.\end{cases}$
Арифметический	$D_2^{-1}=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$	$D_{2^{t+1}}^{-1}=D_2^{-1}\otimes D_{2^t}^{-1}$ $D_{2^{t+1}}=D_2\otimes D_{2^t}$	$A(X)=\sum_{i=0}^{2^n-1} a_i X_1^{i_1} X_2^{i_2} \ldots X_n^{i_n}$
	$D_2=\begin{bmatrix}1 & 0\\ -1 & 1\end{bmatrix}$	$A=D_N F$, $F=D_N^{-1} A$	$X_i^j=\begin{cases} X_i, & j=1;\\ 1, & j=0.\end{cases}$
Уолша	$D_2^{-1}=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$D_{2^{t+1}}^{-1}=D_2^{-1}\otimes D_{2^t}^{-1}$ $D_{2^{t+1}}=D_2\otimes D_{2^t}$	$W(X)=\frac{1}{2^n}\sum_{i=0}^{2^n-1} w_i(-1)^{\theta_i(X)}$
	$D_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$W=\frac{1}{2^n} D_N F$, $F=D_N^{-1} W$	$\theta_i(X)=\sum_{j=1}^{n} i_j X_j$
Хаара	$D_2^{-1}=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$D_{2^{t+1}}^{-1}=\left[\begin{array}{c\|c} D_{2^t}^{-1} & \sqrt{2^t} I_{2^t} \\ \hline D_{2^t}^{-1} & -\sqrt{2^t} I_{2^t}\end{array}\right]$	$H(X)=\frac{1}{2^n}\left(h_0+\right.$
	$D_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$D_{2^{t+1}}=\left[\begin{array}{c\|c} D_{2^t} & D_{2^t} \\ \hline \sqrt{2^t} I_{2^t} & -\sqrt{2^t} I_{2^t}\end{array}\right]$	$\left.+\sum_{j=0}^{n-1}\theta_j \sum_{i=2^j}^{2^{j+1}} h_i X_1^{i_1}\ldots X_j^{i_j}\right)$
		$I_{2^t}=\begin{bmatrix}1 & 0 & \cdot & \cdot & \cdot & 0\\ 0 & 1 & \cdot & \cdot & \cdot & 0\\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\ 0 & 0 & \cdot & \cdot & \cdot & 1\end{bmatrix}$	$\theta_j=\sqrt{2^j}(-1)^{x_{j+1} i_{n-j}}$
		$H=\frac{1}{2^n} D_N F$, $F=D_N^{-1} H$	$X_i^j=\begin{cases} X_i, & j=1;\\ \overline{X_i}, & j=0.\end{cases}$

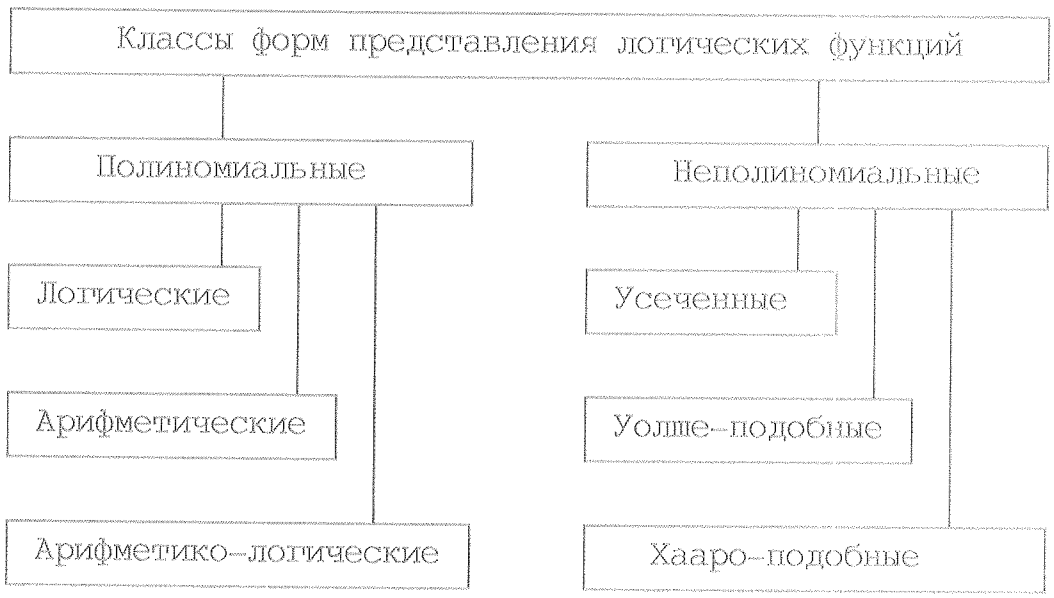


Рис. 1.5. Формы представления логических функций

Следует ожидать, что расширение количества операций, используемых в аналитической конструкции формы позволит реализовать логическое вычисление более эффективным образом.

§ 4. Параллельные логические вычисления

Помимо реализации одной логической функции на практике часто встречается задача реализации системы логических функций, заданных на общем множестве независимых переменных. Ставится задача распараллеливания логического вычисления [32, 33].

Пусть система связанных между собой логических функций представлена в виде:

$$f_j=\psi_j(\mathbb{X}_j,\ \mathbb{F}_j),\quad j=\overline{1,\ q}, \tag{1.13}$$

где

$$\mathbb{X}_j\subseteq\{X_1,\ X_2,\ \ldots,\ X_n\},\quad X_i\in\{0,\ 1,\ \ldots,\ I\},\quad i=\overline{1,\ n};$$

$$\mathbb{F}_j\subseteq\{f_1,\ f_2,\ \ldots,\ f_q\},\quad f_j\in\{0,\ 1,\ \ldots,\ I\},\quad j=\overline{1,\ q}.$$

О системе функций (1.13) можно утверждать, что всякая функция представима как

$$f_j=f_j(X_1, X_2, \ldots, X_n),\ j=\overline{1, q}, \qquad (1.14)$$

т.е. зависит только от входных аргументов и потому может быть реализована независимо от других функций системы. Количество реализуемых функций q, как правило, значительно превосходит количество параллельных каналов (единиц оборудования), имеющегося в составе вычислительного средства.

Исходя из изложенного, основную проблему параллельных логических вычислений сформулируем так: для системы логических функций (1.14) установить такую взаимную зависимость, с учетом которой требуется использовать минимальный ресурс, достаточный для совместной реализации. Цель распараллеливания – посредством формального преобразования привести систему логических функций к форме, применение которой повысит эффективность реализации (повысит эффективность использования ресурсов вычислительного средства или (и) минимизирует время вычислений).

Как показано в работах [34, 35], распараллеливание логических вычислений возможно, а при определенных условиях эффективно, если перейти к формам совместного описания системы логических функций. Наибольшую известность получили такие формы совместного описания системы (1.14) как обобщенные логические [18], арифметические [22] и обобщенные дизъюнктивные [36]. Логические полиномиальные формы в поле Галуа не нашли широкого применения из-за ограничений по количеству представляемых функций (см. § 3).

Арифметические формы систем функций. Одно из достоинств арифметических форм, и в частности полиномиальных, заключается в возможности обобщения на систему логических функций. При этом сохраняется не только структура аналитической конструкции, но и способ ее синтеза, что имеет принципиальное значение на практике. К арифметическим формам относят те формы, в аналитической конструкции которых используется операция арифметического сложения

(арифметическая полиномиальная, Хаара, Уолша и др.). Формы совместного описания системы логических функций, следуя [37], далее будем называть D-формами.

Основной метод синтеза D-форм системы (1.14) заключается в использовании взвешенной суммы арифметических форм представления функций системы:

$$P(X)=\sum_{j=1}^{q} f_j(X)k^{j-1}=\sum_{i=0}^{k^n-1} A_i\theta_i(X), \qquad (1.15)$$

где k – значность алгебры логики, A_i – коэффициенты формы совместного описания, $\theta_i(X)$ – аналитическое выражение, зависящее от выбора базиса (ортогональные функции). Для коэффициентов A_i справедливо выражение:

$$A_i=\sum_{j=1}^{q} a_{ij}k^{j-1}, \qquad (1.16)$$

в котором a_{ij} – i-й коэффициент арифметической формы j-й функции системы.

По коэффициентам A_i D-формы (1.15) можно восстановить исходную систему. Взаимно однозначная связь между вектором коэффициентов A_i и системой (1.14) реализуется посредством ДПФ в выбранном базисе, т.е. аналогично тому, как связываются коэффициенты арифметической формы логической функции и характеристический вектор F парой ДПФ. Для получения вектора коэффициентов D-формы необходимо в выражение (1.12) подставить характеристический вектор системы логических функций, формируемый в соответствии со следующим выражением:

$$F(i)=\sum_{j=1}^{q} F_j(i)k^{j-1}, \qquad (1.17)$$

в котором $F_j(i)$ – значение j-й функции в точке i. Здесь, как и ранее, соответствие между числом i и значениями независимых переменных устанавливается по k-ичному представлению i.

Заметим, что в случае стохастических систем D-форма имеет ряд

преимуществ, поскольку решение задачи можно свести к вероятному описанию коэффициентов формы. Возможности арифметических форм для описания систем логических функций связывают также с особенностями их структурной минимизации [38]. Для арифметических форм минимизация возможна путем: доопределения системы функций дополнительными функциями; перестановки функций в исходной системе; расширения набора операций при использовании как переменных, так и их инверсий. Там же отмечается, что процедура минимизации носит эвристический (переборный) характер.

Обобщенная дизъюнктивная форма. Другой формой, позволяющей описать произвольную систему булевых функций является обобщенная дизъюнктивная форма (ОД-форма) [36]. Основой для построения ОД-формы служит СДНФ. Каждая функции f_j системы (1.14) представляется в СДНФ (1.10), затем все q функций объединяются в единое выражение

$$P(X)=\bigcup_{j=1}^{q} 2^{j-1} f_j(X),$$

и после приведения подобных термов получают

$$P(X)=\bigcup_{i=0}^{2^n-1} d_i X_1^{i_1}\&X_2^{i_2}\&\ldots\&X_n^{i_n},$$

где коэффициенты d_i являются значениями системы функций в точках i области определения (в соответствии с (1.17) при k=2):

$$d_i=(f_q(i),\ \ldots,\ f_2(i),\ f_1(i))_2.$$

Здесь $f_j(i)\in\{0,\ 1\}$ – значение функции f_j в точке i.

Сложность представления системы булевых функций в ОД-форме определяется количеством ненулевых строк в таблице истинности системы, и для достаточно сложных систем, когда нулевых строк нет, количество дизъюнкций равно 2^n. Дальнейшее увеличение количества функций в системе приводит только к увеличению объема памяти, необходимого для хранения коэффициентов d_i, причем этом объем памяти совпадает с объемом памяти табличного представления. В

конечном итоге, ОД-форма достаточно сложной системы булевых функций по своей эффективности совпадает с табличным представлением.

Сложность представления систем функций. Для оценки сложности представления систем логических функций, определяемой числом операций в минимальной представлении, необходимо учесть то обстоятельство, что каждая такая функция имеет свое минимальное выражение в используемом базисе. Совместная минимизация такой системы (выделение общих подвыражений для всех или части функций [39]) не всегда приводит к успеху. В результате чего может быть сформулирована следующая асимптотическая оценка: для всех форм совместного описания систем логических функций при достаточно большом количестве функций q в системе ($q>>n$, где n -- количество независимых переменных) сложность формулы L_Ω в некотором базисе Ω стремится к величине k^n:

$$L_\Omega(F)\sim C_\Omega k^n,$$

где C_Ω – константа, зависящая от выбора базиса Ω, k – значность алгебры логики.

В конечном итоге, минимизация систем логических функций возможна путем выбора базиса Ω, но это не приводит к какому-либо существенному эффекту при достаточно большом количестве независимых переменных из-за экспоненциального роста сложности формулы. Выполнение структурной минимизация арифметических форм позволяет получить больший эффект, но это связано с увеличением объема вычислений по причине переборного характера известных алгоритмов минимизации и с увеличением объема памяти для хранения коэффициентов минимизируемой формы.

§ 5. Кратные логические вычисления

Потребность в выполнении требований по заданным критериям качества реализации логического вычисления приводит к использованию параллельного логического вычисления. Параллельное логическое вычисление заключается в такой организации вычислительного процесса, когда одновременно вычисляется вся система функций на одном наборе переменных (параллелизм по логическим функциям).

Помимо задачи распараллеливания вычислений ставится задача повторения вычислений на различных наборах переменных. Как определено в § 1, под кратными логическими вычислениями понимается процесс одновременного нахождения нескольких значений одной или системы логических функций на разных наборах ее аргументов (параллелизм по наборам переменных). Величина кратности m лежит в диапазоне от 1 до k^n, где n – число аргументов, k – значность алгебры логики.

Кратные вычисления возникают при параллельном управлении несколькими одинаковыми объектами, находящимися в разных физических состояниях; при моделировании (воспроизведении) системы на нескольких входных воздействиях, например вычисление модели при тестировании и технической диагностике дискретных устройств; в системах управления, где требуется быстрая реакция на изменение внешних условий, можно с упреждением просчитать нужные действия на возможных (вероятных) входных наборах, например, находящихся на единичном расстоянии по Хеммингу от текущего, и при поступлении нового набора уже иметь нужную реакцию. Кратные вычисления возникают также при решении логических уравнений.

Проанализируем основные разновидности современных высокопроизводительных системы обработки данных (СОД) по эффективности реализации кратных вычислений (рис. 1.6). Применением кратных вычислений возможно на различных уровнях СОД (с различной гранулярностью по [40]), отличающихся мерой крупности операций обработки [41]: крупноблочный уровень: процесс – процессор; среднеблочный уровень: команда – устройство управления; мелкоблочный уровень: операция – операционное устройство. Для краткости будем называть эти уровни соответственно программным, командным и операционным.

Рис. 1.6. Высокопроизводительные системы обработки данных

Критерии качества реализации. Для оценки эффективности различных методов кратных вычислений введем следующие критерии эффективности: эффективность по времени вычисления ξ; эффективность по сложности вычислений η; комплексная эффективность ϑ. В качестве базового метода, с которым будем производить сравнение, выберем однократный случай, время и сложность вычислений в котором принимаем за единицу времени и сложности.

Эффективность по времени вычисления ξ определяет во сколько раз время m-кратных вычислений для исследуемого метода меньше чем повторение вычислений m раз базовым методом. Если известно время m-кратных вычисления τ_m для исследуемого метода, то его эффективность по времени межет быть выражена как

$$\xi=\frac{m\cdot 1}{\tau_m}=\frac{m}{\tau_m}.$$

Эффективность по сложности вычислений η определяет во сколько раз затраты на аппаратные средства (объем памяти при программной реализации) превосходят аналогичные затраты в базовом методе. Если известны аппаратные затраты m-кратных вычислений ν_m для исследуемого метода, то его эффективность по сложности вычисления оценим выражением

$$\eta=\frac{1}{\nu_m}.$$

Комплексная эффективность η характеризует эффективность метода кратных вычислений с учетом эффективностей по времени вычисления ξ и по сложности η. Целесообразно комплексную эффективность ϑ определить следующим образом:

$$\vartheta=\xi\eta=\frac{m}{\tau_m\nu_m},$$

где τ_m (ν_m) – время выполнения (сложность реализации) m-кратного вычисления. В качестве единиц измерения времени и сложности, как и ранее, примем время и сложность однократного вычисления.

Классификация методов кратных вычислений. Вычислительные средства, выполняющие кратные вычисления могут быть разделены по особенностям реализации вычислительного процесса. В основу классификации положим следующие структурые принципы повторности: параллельная или последовательная повторность; повторность в пространстве или во времени. На основе введенных структурных принципов можно выделить параллельные и последовательные кратные вычисления осуществляемые как в пространстве, так и во времени (рис. 1.7).

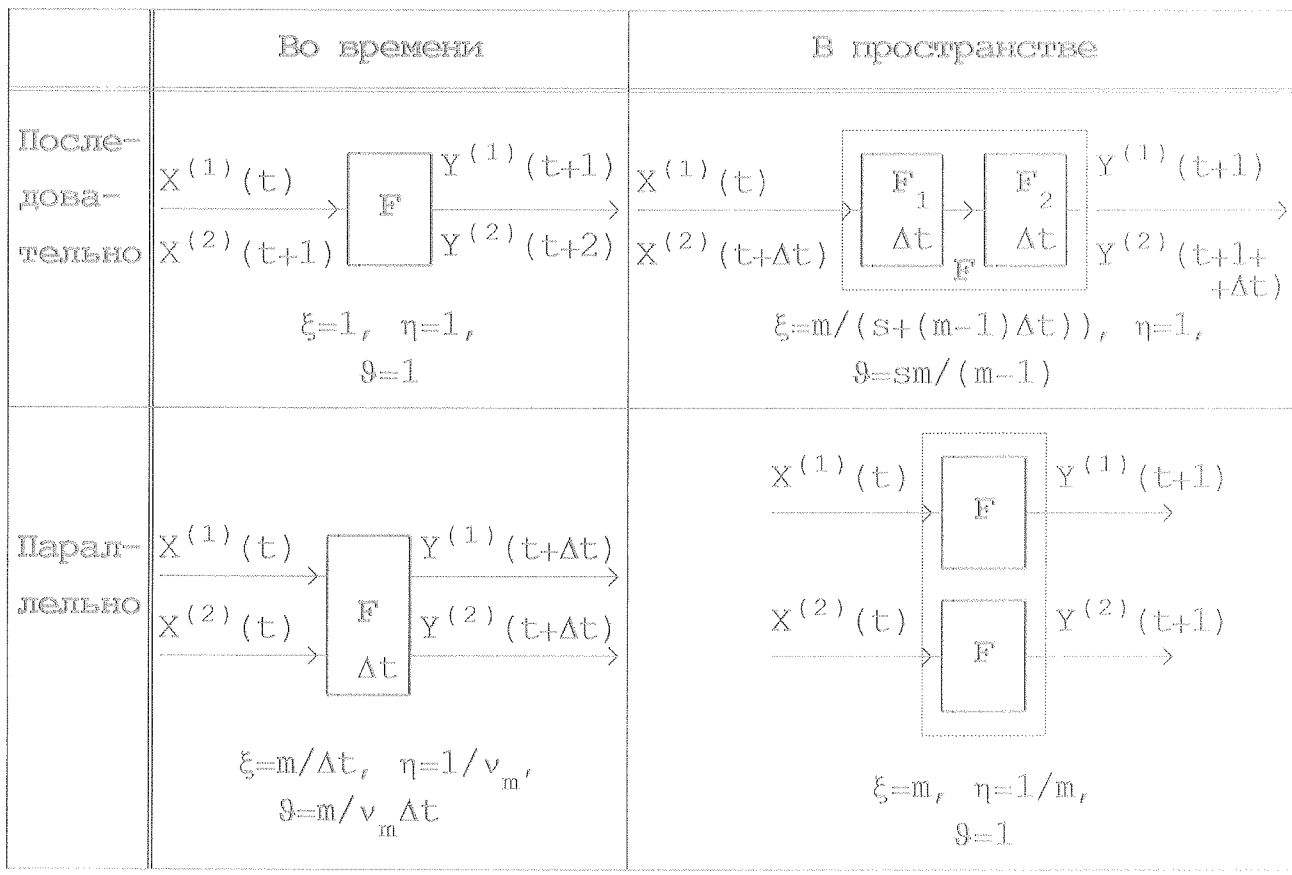


Рис. 1.7. Методы структурной организации кратных вычислений

Последовательные вычисления во времени представляют собой такую организацию вычислений, когда различные наборы переменных $X^{(r)}$, $(r=\overline{1, m})$, подаются на вход вычислительного средства в различные моменты времени t, отстоящие друг от друга на время,

достаточное для однократного вычисления, т.е. последовательность входных наборов переменных

$$\left\{X^{(1)}(t),\ X^{(2)}(t+1),\ \ldots,\ X^{(m)}(t+m-1)\right\}$$

порождает соответствующую последовательность результатов

$$\left\{Y^{(1)}(t+1),\ Y^{(2)}(t+2),\ \ldots,\ Y^{(m)}(t+m)\right\}.$$

Эффективность последовательных во времени кратных вычислений можно определить как

$$\xi_1=\frac{m}{m}=1,\quad \eta_1=\frac{1}{1}=1,\quad \vartheta_1=\xi_1\eta_1=1.$$

К последовательным вычислениям во времени можно отнести все методы, реализующие однократные вычисления и повторяемые количество раз, равное кратности m. Последовательные вычисления во времени – основной метод реализации кратных вычислений в системах обработки данных с фон–Неймановской архитектурой [42]. Основной архитектурный принцип построения таких систем – наличие единого обрабатывающего устройства на крупноблочном уровне, выполняющего последовательность команд программы. По такому принципу реализуются основные элементарные обрабатывающие устройства (процессорные элементы) на мелкоблочном уровне во всех известных СОД: в матричных, систолических, ассоциативных, транспьютерных, мультипроцессорных, волновых и т.д. (см. рис. 1.6).

Последовательные вычисления в пространстве соответствуют конвейерной организации вычислений [43]. В этом случае различные наборы переменных $X^{(r)}$ подаются на вход вычислительного средства также в различные моменты времени, но отстоящие друг от друга на время Δt, которого достаточно для вычислений, осуществляемого самой медленной ступенью конвейера. Первый набор переменных $X^{(1)}$ поступает на первую ступень конвейера в некоторое время t. Через время Δt ступень завершает свою работу и на ее вход может быть подан второй набор переменных $X^{(2)}$. Результат вычисления

$Y_1^{(1)}(t+\Delta t)$, полученный после первой ступени, подается на вход второй ступени и т.д. Первый результат появляется на выходе через время $s\Delta t$, равное суммарному времени работы всех ступеней; каждый последующий – через время Δt. Таким образом, последовательность входных наборов переменных

$$\left\{X^{(1)}(t),\ X^{(2)}(t+\Delta t),\ \ldots,\ X^{(m)}(t+(m-1)\Delta t)\right\}$$

порождает соответствующую последовательность результатов

$$\left\{Y^{(1)}(t+s\Delta t),\ Y^{(2)}(t+(s+1)\Delta t),\ \ldots,\ Y^{(m)}(t+(s+m-1)\Delta t)\right\}.$$

Эффективность кратных вычислений, осуществляемых последовательно в пространстве, может быть оценена как

$$\xi_2=\frac{m}{(s+m-1)\Delta t},\quad \eta_2=\frac{1}{v_1+v_2+\ldots+v_s},$$

$$\vartheta_2=\xi_2\eta_2=\frac{m}{(v_1+v_2+\ldots+v_s)(s+m-1)\Delta t},$$

где m – кратность вычислений, s – количество ступеней конвейера, v_i – сложность вычисления на i-й ступени, Δt – время работы одной ступени.

Для реализации конвейерного метода кратных вычислений необходимо систему функций декомпозировать на s более простых подсистем F_p, $p=\overline{1,\ s}$, причем таким образом, чтобы результаты вычисления предыдущей подсистемы F_j являлись значениями независимых переменных следущей подсистемы F_{j+1}:

$$F(X)=F_s(F_{s-1}(\ldots F_2(F_1(X))\ldots)).$$

В общем случае, описанная декомпозиция называется многократной (см. табл. 1.2) и рассматривается как комбинаторная задача, точный алгоритм которой достаточно трудоемок, ибо связан с большим числом перебираемых решений [9]. Сложность декомпозиции системы (1.14) связана с необходимостью одинаковой декомпозиции каждой из функций этой системы. В практическом плане создание эффективных алгоритмов декомпозиции произвольной системы логических функций остается по

прежнему актуальным.

В технических системах [44] получили распространения такие методы конвейеризации, в которых каждая подсистема вычисляется за время, меньшее в s раз чем время однократного вычисления ($\Delta t \approx 1/s$), а общая сложность вычислений незначительно превосходит сложность однократного вычисления ($\nu_1+\nu_2+\ldots+\nu_s \approx 1$). С учетом этого получаем:

$$\xi_2 = \frac{m}{1+(m-1)\Delta t} \underset{(m \gg s)}{\approx} \frac{sm}{(m-1)}, \quad \eta_2 \approx 1, \quad \vartheta_2 = \xi_2 \eta_2 \approx \frac{sm}{(m-1)}.$$

Как видно из последнего выражения, при достаточно большом m, комплексная эффективность ϑ конвейерного метода стремяться к s – количеству ступеней конвейера.

Из проблем, которые здесь возникают следует упомянуть проблему, связанную с требованиями к реализации системы F_j на j-й ступени конвейера. Во-первых, такое вычисление по возможности должно иметь комплексную эффективность, равную s в сравнении с реализацией исходной системы функций F. Если это не выполняется, то комплексная эффективности метода в целом будет меньше s, что может привести и к неэффективной реализации. Во-вторых, времена вычислений на каждой ступени должны быть выравнены, что также вносит дополнительные ограничения при декомпозиции вычисляемой системы функций. На практике имеется небольшое число задач, для которых найдена эффективная декомпозиция соответствующих систем функций [44].

Примерами СОД, в которых реализованы последовательные вычисления в пространстве могут служить векторно-конвейерные, систолические и волновые системы, нейронные сети (среднеблочный уровень), а также процессорные элементы (процессоры) с конвейеризацией команд (среднеблочный уровень) и/или конвейеризацией операций (мелкоблочный уровень).

Параллельные вычисления в пространстве могут быть представлены такой организацией вычислений, когда имеется несколько

вычислительных средств, каждое из которых выполняет вычисление системы F для своего набора переменных X [45, 46]. В этом случае результаты кратных вычислений Y появляются на выходе через время, равное времени однократного вычисления, но при этом требуется дублирование вычислительных средств в количестве, равном кратности m. Таким образом, входные наборы переменных

$$\left\{X^{(1)}(t),\ X^{(2)}(t),\ \ldots,\ X^{(m)}(t)\right\}$$

подаются одновременно и одновременно появляются на выходе соответствующие результаты

$$\left\{Y^{(1)}(t+1),\ Y^{(2)}(t+1),\ \ldots,\ Y^{(m)}(t+1)\right\}.$$

Эффективность параллельных в пространстве кратных вычислений равна:

$$\xi_3=m,\quad \eta_3=\frac{1}{m},\quad \vartheta_3=\xi_3\eta_3=1.$$

Примерами СОД с параллельными вычислениями в пространстове являются мультипроцессорные системы (крупноблочный уровень); векторно-параллельные (матричные), ассоциативные, потоковые, систолические и волновые системы и нейронные сети (среднеблочный уровень); процессорные элементы (процессоры) со сверхдлинным командным словом, суперскалярной обработкой данных (среднеблочный уровень); с поразрядными логическими операциями (мелкоблочный уровень).

Особенностью современных микропроцессоров является реализация в них поразрядных логических операций, что является основой для выполнения векторных логических программ [47]. Под векторной двуместной $\circ$ (векторной одноместной $\neg$) операцией над булевыми векторами $X=[X_1, \ldots, X_m]$ и $Y=[Y_1, \ldots, Y_m]$ понимается такая операция, что $Z=X\circ Y$ ($Z=\neg X$), где $Z=[Z_1, \ldots, Z_m]$, $Z_i=X_i\circ Y_i$ ($Z_i=\neg X_i$) для всех $i\in\{1, \ldots, m\}$ и при $X_i, Y_i, Z_i\in\{0, 1\}$.

Аппаратной векторной реализацией системы булевых функций

является комбинационная схема, состоящая из векторных логических элементов. Векторный логический элемент реализует функционал L_m над векторными переменными $X, Y, \ldots, Z$, т.е.

$$L_m(X, Y, \ldots, Z)=(A_1, A_2, \ldots, A_m),$$

где $A_i=f(X_i, Y_i, \ldots, Z_i)$ для всех $i\in\{1, \ldots, m\}$. Методы синтеза векторных комбинационных схем мало отличаются от методов синтеза традиционных комбинационных схем.

Показано [там же], что любой векторной комбинационной схеме, содержащей N элементов, можно поставить в соответствие векторную операторную программу, вычисляющую систему логических функций и состоящую из N векторных команд, каждая из которых реализует векторную операцию.

Таким образом, векторное логическое вычисление является методом структурной реализации кратных вычислений одной функции. Значение кратности равно m – разрядности векторного логического элемента или разрядности используемого микропроцессора. Система команд такого микропроцессора должна поддерживать векторные (поразрядные) логические операции. Очевидно, что комплексная эффективность векторного логического вычисления равна единице.

Векторное вычисление не единственный метод распараллеливания вычислений. Так, в работе [48] рассмотрен метод распараллеливания вычислений на крупноблочном уровне, где произвольный алгоритм A_f^n, вычисляющий логическую функцию f от n переменных заменен на систему независимых алгоритмов $A_f^{n,j}$, $j=\overline{0, n}$, каждый из которых вычисляет значение $f(\sigma)$, если

$$\sigma\in B_2^{n,j}\subset B_2^n,\quad B_2^n=\underbrace{B_2\times B_2\times\ldots\times B_2}_{\text{n раз}},\quad B_2=\{0, 1\},$$

и содержит j единичных значений; и вычисляет нуль, в противном случае. Общая схема вычислений состоит из блока, определяющего общее число единиц во входном векторе σ и n+1 параллельных блоков,

реализующих алгоритмы $A_f^{n,j}$. В этом случае оказывается, что время вычисления

$$t^{n,j}<(n+]\log_2(n+1)[),$$

а процесс вычисления ведется на нескольких автономных процессорах, между которыми нет обмена данными.

Использование описанного метода позволяет реализовать кратные вычисления только путем увеличения количества процессоров до величины $m(n+1)$, т.е. группа из $(n+1)$ процессоров используется для каждого набора данных σ_i, $i=\overline{1, m}$. Комплексная эффективность этого метода меньше единицы, т.к. многоразрядный процессор используется для вычисления только одной функции. При использовании однобитового процессора комплексная эффективность метода становится равной единице.

Параллельные вычисления во времени соответствуют такой организации вычислений, когда на вход вычислительного средства одновременно подаются все m наборов переменных

$$\left\{X^{(1)}(t),\ X^{(2)}(t),\ \ldots,\ X^{(m)}(t)\right\}$$

и через некоторое время Δt одновременно получаются все m результатов вычислений

$$\left\{Y^{(1)}(t+\Delta t),\ Y^{(2)}(t+\Delta t),\ \ldots,\ Y^{(m)}(t+\Delta t)\right\}.$$

Очевидно, что для отличия параллельных вычислений во времени от параллельных вычислений в пространстве (по аналогии с последовательными вычислениями во времени и в пространстве) комплексный критерий эффективности

$$\vartheta_4=\frac{m}{\Delta t\cdot v_m},$$

должен быть строго больше единицы, т.е. $\vartheta_4>1$.

Таким образом, к параллельным вычислениям во времени могут быть отнесены такие методы кратных вычислений, при реализации которых комплексный критерий эффективности больше единицы и

исходные данные (результаты) подаются (снимаются) одновременно.

Анализ современных вычислительных систем показывает, что основными методами структурной организации кратных вычислений являются последовательные вычисления во времени (повторная обработка), последовательные вычисления в пространстве (конвейеризация или многостадийная обработка) и параллельные вычисления в пространстве (распараллеливание или многоэлементная обработка).

Исследование возможности реализации параллельных вычислений во времени остается актуальным. Как было показано ранее, основным структурным свойством параллельных вычислений во времени (рис. 1.7) является одновременная подача наборов переменных и одновременное получение результата. По аналогии с рассмотренными методами реализации логического вычисления (см. § 2) можно предположить, что поиск методов параллельных во времени кратных вычислений следует осуществлять в области совместного описания систем логических функций.

§ 6. Обобщенная форма кратных вычислений

Дадим формальную интерпретацию различных типов логических вычислений над функцией от n переменных.

Пусть задана произвольная система q логических функций от n независимых переменных в алгебре k–значной логики B_k:

$$F(X)=\left(f_1(X),\ \ldots,\ f_q(X)\right),\qquad X=\left(X_1,\ \ldots,\ X_n\right), \tag{1.18}$$

где f_j – логические функции, $j=\overline{1,\ q}$; X_i – независимые переменные, $i=\overline{1,\ n}$; $f_j,\ X_i\in B=\{0,\ 1,\ \ldots,\ I\}$; I – логическая единица алгебры, $I=k-1$.

Следуя [34], систему логических функций (1.18) удобно представлять в виде кортежа функций и переменных, где под кортежем

будем понимать упорядоченную совокупность объектов произвольной природы:

$$F(X)=\left(f_1(X)\#f_2(X)\#\ldots\#f_q(X)\right),\quad X=\left(X_1\#X_2\#\ldots\#X_n\right),$$

или, что то же самое,

$$F(X)=\mathop{\#}_{j=1}^{q} f_j(X),\qquad X=\mathop{\#}_{i=1}^{n} X_i,$$

где знак # – разделительный, определяющий некоторый способ ортогонального объединения f_j и X_i.

Здесь под ортогональным объединением будем понимать такой способ совместного описания совокупности математических объектов, который обладает свойствами функциональности, инъективности и сюръективности, т.е. позволяющий установить взимно однозначное соответствие между упорядоченными множествами математических объектов и формами их совместного описания.

Например, для независимых переменных X_i ортогональное объединение может быть представлено в виде вектора:

$$X=\left[X_1,\ X_2,\ \ldots,\ X_n\right]^T,$$

где знак т означает операцию транспонирования. Возможно также ортогональное объединение независимых переменных $X_i<k$ на основе представления произвольного натурального числа в k–ичной позиционной системе счисления:

$$X=\sum_{i=1}^{n} X_i k^{i-1}=\left(X_n\ \ldots\ X_2\ X_1\right)_k.$$

К формам ортогонального объединения систем булевых функций могут быть отнесены, например, арифметические формы (1.15), в которых объединение основано на выделении для значения каждой функции соответствующего разряда в двоичном представлении натурального числа и которое является результатом вычисления аналитической конструкции формы. Другой формой ортогонального объединения булевых функций может быть описание системы функций в

алгебре кортежей [35].

Систематизируем задачи логического вычисления, встречающиеся на практике и описываемые системой (1.18) с использованием введенных обозначений (табл. 1.3).

Таблица 1.3

Кратность	Функция одной переменной	Функция n переменных	Система q функций от n переменных
1	$Y=f(X)$	$Y=f(X),$ $X=\#_{i=1}^{n} X_i,$	$Y=F(X),$ $F(X)=\#_{j=1}^{q} f_j(X),$ $X=\#_{i=1}^{n} X_i,$ $Y=\#_{j=1}^{q} Y_j.$
m	$\tilde{Y}=\tilde{f}(\tilde{X}),$ $\tilde{X}=\#_{r=1}^{m} X^{(r)},$ $\tilde{Y}=\#_{r=1}^{m} Y^{(r)}.$	$\tilde{Y}=\tilde{f}(\tilde{X}),$ $\tilde{X}=\#_{r=1}^{m}\#_{i=1}^{n} X_i^{(r)},$ $\tilde{Y}=\#_{r=1}^{m} Y^{(r)}.$	$\tilde{Y}=\tilde{F}(\tilde{X}),$ $\tilde{F}(\tilde{X})=\#_{j=1}^{q} \tilde{f}_j(\tilde{X}),$ $\tilde{X}=\#_{r=1}^{m}\#_{i=1}^{n} X_i^{(r)},$ $\tilde{Y}=\#_{r=1}^{m}\#_{j=1}^{q} Y_j^{(r)}.$

Здесь знак ~ используется для указания на совокупность кратных значений математического объекта, а верхний индекс в круглых скобках (r) обозначает r-е значение математического объекта, определяемое с некоторой кратностью m.

Заметим, что в случае m-кратного вычисления используются уже не линейные, а плоские кортежи наборов переменных $\tilde{X}$ и значений системы функций $\tilde{Y}$:

$$\tilde{X}=\begin{pmatrix} X_1^{(1)} & \# & X_1^{(2)} & \#\cdots\# & X_1^{(m)} \\ \# & & \# & & \# \\ X_2^{(1)} & \# & X_2^{(2)} & \#\cdots\# & X_2^{(m)} \\ \# & & \# & & \# \\ \cdots & & \cdots & \cdots & \cdots \\ \# & & \# & & \# \\ X_n^{(1)} & \# & X_n^{(2)} & \#\cdots\# & X_n^{(m)} \end{pmatrix}, \quad \tilde{Y}=\begin{pmatrix} Y_1^{(1)} & \# & Y_1^{(2)} & \#\cdots\# & Y_1^{(m)} \\ \# & & \# & & \# \\ Y_2^{(1)} & \# & Y_2^{(2)} & \#\cdots\# & Y_2^{(m)} \\ \# & & \# & & \# \\ \cdots & & \cdots & \cdots & \cdots \\ \# & & \# & & \# \\ Y_q^{(1)} & \# & Y_q^{(2)} & \#\cdots\# & Y_q^{(m)} \end{pmatrix}.$$

Как видно из табл. 1.3, наиболее общим представлением логических вычислений является m-кратные вычисления системы q функций от n переменных. Варьируя значения n, q и m можно получить все выше перечисленные случаи логического вычисления.

Для реализации кратных вычислений необходимо некоторым образом преобразовать систему логических функций F(X) в форму $\tilde{F}(\tilde{X})$, ортогонально объединяющую q функций и m наборов n переменных. Такая форма совместного описания может быть представлена как

$$\tilde{Y}=\tilde{F}(\tilde{X})=\tilde{F}\left(\begin{matrix} m \\ \# \\ r=1 \end{matrix} X^{(r)}\right)=\begin{matrix} q \\ \# \\ j=1 \end{matrix}\tilde{f}_j\left(\begin{matrix} m & n \\ \# & \# \\ r=1 & i=1 \end{matrix} X_i^{(r)}\right), \qquad (1.19)$$

где $\tilde{f}_j$ ($\tilde{F}$) – некоторая форма представления логической функции (системы функций), позволяющая описать кратные логические вычисления.

Из выражения (1.19) видно, что $\tilde{f}_j$ является логической функцией от n·m независимых переменных. При построении $\tilde{f}_j$ необходимо учесть, что вхождение независимых переменных X_i определяется видом функции f_j, а вхождение m значений каждой из этих перемеменных $X_i^{(r)}$ определяется способом их ортогонального объединения.

Таким образом, реализация кратных вычислений может быть основана на представлении системы функций на множестве наборов переменных в виде единого выражения.

Выводы

1. Логическая обработка данных может быть сведена к логическому синтезу и кратному логическому вычислению.

2. Известные методы реализации логического вычисления своей структурой повторяют формульное представление логической функции, а основой повышения эффективности логического вычисления служит расширение базиса операций при логическом синтезе.

3. Известны три подхода к минимизация систем логических функций: тождественные преобразования формул в заданном базисе операций, выбор оптимального базиса и структурная минимизация формы. Для достаточно сложных систем функций наиболее эффективной является структурная минимизация.

4. Методами структурной организации кратных вычислений являются последовательные во времени, последовательные в пространстве, параллельные во времени и параллельные в пространстве кратные вычисления.

5. К параллельным во времени кратным вычислениям могут быть отнесены такие методы, при реализации которых комплексный критерий эффективности больше единицы, а исходные данные (результаты вычислений) подаются (снимаются) одновременно.

6. Кратные вычисления могут быть описаны в виде единого выражения, объединяющего множество функций на множестве наборов переменных.

Как показано в главе 1, всякая реализация логического вычисления своей структурой повторяет формульную запись. Это позволяет дать формальную интерпретацию различных типов логических вычислений. В качестве вычислительной модели будем использовать вычислительное средство на базе арифметико-логического устройства (рис. 2.1).

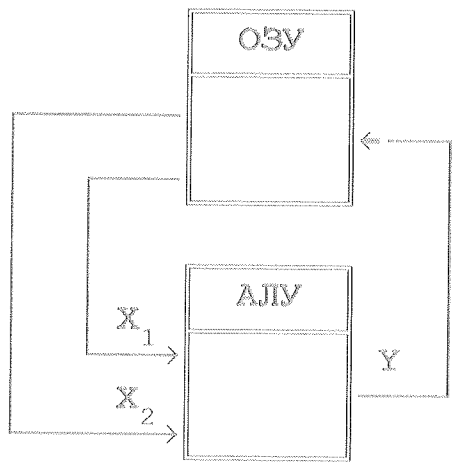


Рис. 2.1. Модель вычислительного механизма

Традиционные логические вычисления. Традиционную реализацию одной булевой функции вида $f(X)=Y$, где $X=(X_1, X_2, \ldots, X_n)$ представим как операцию

$$f(X^{(i)})=Y^{(i)}, \quad i=\overline{0, 2^n-1},$$

что означает нахождение значения функции $Y^{(i)}$ при подстановке в аналитическое выражении для функции f i-го набора аргументов $X^{(i)}$ (рис. 2.1).

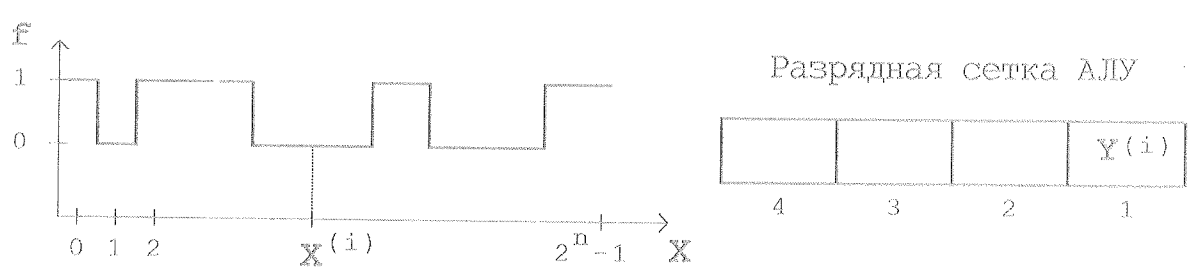


Рис. 2.1. Традиционные логические вычисления

Пример 2.1. Значение функции $f(X)=(X_3\&X_2)\vee X_1$ в точке $X=(010)_2$ может быть вычислено следующим образом: $f(X)=(0\&1)\vee 0=0$.

Параллельные логические вычисления. Второй случай – параллельные вычисления системы булевых функций f_j, $j=\overline{1,q}$, над арифметической или обобщенной дизъюнктивной формой $P(X)$ (см. § 4), что означает операцию:

$$P(X^{(i)})=Y^{(i)}=\#_{j=1}^{q} Y_j^{(i)},$$

где # – знак ортогонального объединения. Ортогональное объединение функций f_j и результатов вычислений Y_j основано на представлении числа в двоичной позиционной системе счисления (рис. 2.2).

Пример 2.2. Для вычисления значения системы

$$\begin{cases} f_1(X)=(X_3\&X_2)\vee X_1; \\ f_2(X)=(X_3\&X_2)\oplus X_1, \end{cases}$$

представим ее в виде арифметического полинома

$$P(X)=3X_1+3X_3X_2-5X_3X_2X_1,$$

подставим в последнее выражение $X=(110)_2$ и находим: $P(X)=(11)_2$.

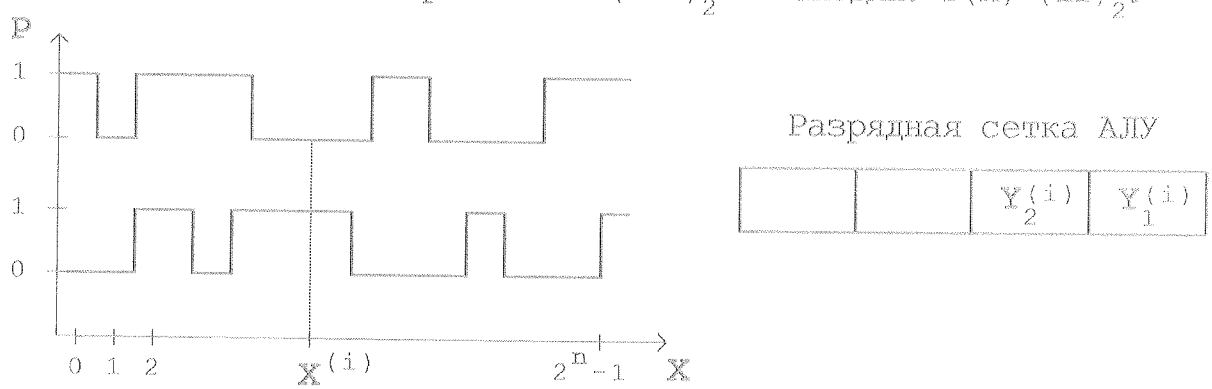


Рис. 2.2. Параллельные логические вычисления

Изменяя номер набора переменных i в первом (втором) случае реализуем последовательные во времени кратные вычисления функции (системы функций). Если удается представить функцию (систему функций) в виде многократной композиции (см. § 5), то возможна реализация кратных вычислений последовательно в пространстве.

Векторные логические вычисления. Более сложный случай –

кратные вычисления булевой функции $f(X)=Y$ означают операцию

$$f\left(\mathop{\#}_{i=1}^{m} X^{(i)}\right)=\mathop{\#}_{i=1}^{m} Y^{(i)},$$

где m – кратность вычислений, при выполнении которой реализуются векторные логические вычисления (см. § 5). Этот случай соответствует параллельным в пространстве кратным вычислениям (рис. 2.3).

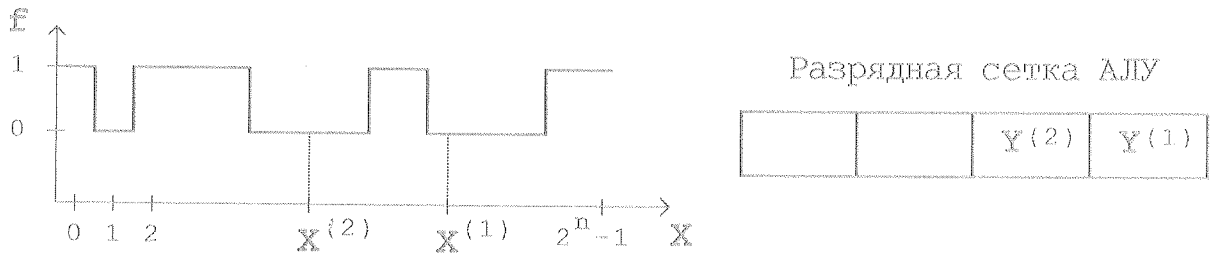


Рис. 2.3. Векторные логические вычисления

Пример 2.3. Значение функции $f(X)=(X_3\&X_2)\vee X_1$ в точках $X^{(1)}=(010)_2$ и $X^{(2)}=(001)_2$ может быть вычислено путем использования поразрядных логических операций. Для этого получим подстановочные значения переменных (булевы вектора): $\tilde{X}_3=(00)_2$, $\tilde{X}_2=(10)_2$, $\tilde{X}_1=(01)_2$, которые используем для кратных вычислений функции:

$$f(\tilde{X})=\left((00)_2\&(10)_2\right)\vee(01)_2=(01)_2.$$

Кратные логические вычисления. И наконец, кратные вычисления системы функций $F(X)=Y$ можно определить как операцию

$$\tilde{F}\left(\mathop{\#}_{i=1}^{m} X^{(i)}\right)=\mathop{\#}_{j=1}^{q}\ \mathop{\#}_{i=1}^{m} Y_j^{(i)}.$$

В последнем случае результатом вычисления будет матрица размерностью $q\times m$, а исходными данными – матрица размерностью $n\times m$. При вычислениях с использованием АЛУ результат представляется в виде линейной последовательности бит (рис. 2.4).

Пример 2.4. Для двукратных вычислений системы из примера 2.2 она предствляется в виде арифметико-логического полинома

$$\tilde{P}(\tilde{X})=5\tilde{X}_1+5(\tilde{X}_3\&\tilde{X}_2)-9(\tilde{X}_3\&\tilde{X}_2\&\tilde{X}_1).$$

Для вычисления системы в точках $X^{(1)}=(111)_2$ и $X^{(2)}=(110)_2$

используем подстановочные значения переменных: $\tilde{X}_3=(11)_2$, $\tilde{X}_2=(11)_2$, $\tilde{X}_1=(01)_2$, и в результате получаем:

$$\tilde{P}(\tilde{X})=5(01)_2+5(11)_2\&(11)_2-9(11)_2\&(11)_2\&(01)_2=(0010)_2.$$

Значения искомых функций на заданных наборах переменных извлекаем из соответствующих разрядов результата вычисления.

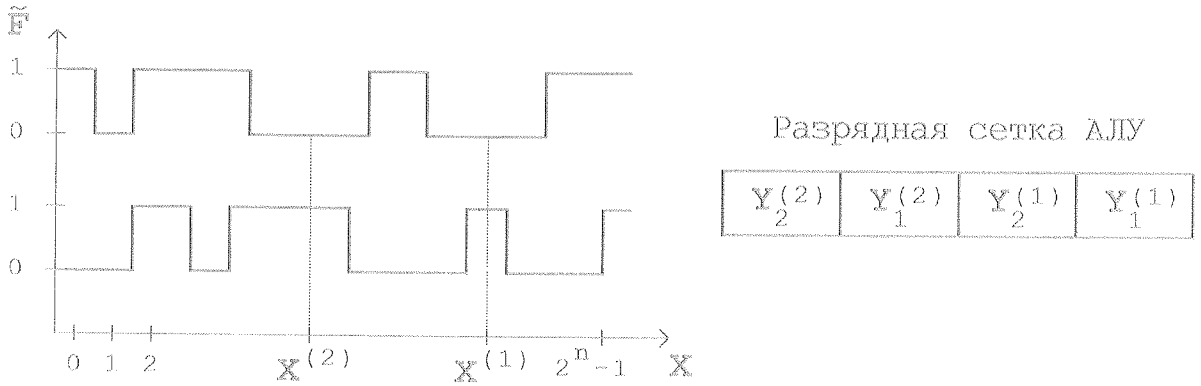


Рис. 2.4. Кратные логические вычисления

Опираясь на ранее полученные результаты [1, 2], исследуем представление кратных вычислений в булевой алгебре.

§ 7. Алгебра поразрядных операций

Рассмотрим более подробно векторные логические вычисления. Пусть требуется вычислить с кратностью m систему q функций от n переменных $X=(X_1, \ldots, X_n)$ в булевой алгебре $B_2=\langle B, \bar{}, \&, V\rangle$:

$$f_j(X_1, \ldots, X_n); \quad j=\overline{1, q}; \quad X_i, f_j \in B=\{0, 1\}. \qquad (2.1)$$

Сведем m-кратные вычисления системы (2.1) к однократному вычислению некоторой системы функций k-значной логики B_k, где $k=2^m$, m – натуральное число. Для этого рассмотрим порождающую систему функций:

$$\{0, I, \bar{X}_i, X_i\&X_j, X_iVX_j\}, \qquad (2.2)$$

где I – логическая единица, $I=2^m-1$; $X_i, X_j \in D=\{0, 1, \ldots, I\}$; $\bar{}$, &, V – соответственно знаки поразрядных логических операций: унарной операции отрицания и бинарных операций конъюнкции и дизъюнкции.

Будем интерпретировать значение переменной или функции как двоичное число, т.е.

$$Y=\sum_{i=1}^{m} Y_i 2^{i-1}=(Y_m \dots Y_1)_2, \quad Y\in D, \quad Y_i\in B \qquad (2.3)$$

где Y_i – i–й разряд числа Y.

Здесь под поразрядными логическими операциями понимаются такие одноместные $\neg$ или двуместные $\circ$ операции, заданные на множестве натуральных чисел N, для которых $\forall X, Y\in D$

$$\neg X=\neg\left(\sum_{i=1}^{n} X_i 2^{i-1}\right)=\sum_{i=1}^{n} (\neg X_i)2^{i-1},$$

$$X\circ Y=\left(\sum_{i=1}^{n} X_i 2^{i-1}\right)\circ\left(\sum_{i=1}^{n} Y_i 2^{i-1}\right)=\sum_{i=1}^{n} (X_i\circ Y_i)2^{i-1}.$$

Использование поразрядных логических операций определяется тем, что современные вычислительные средства содержат в своем составе АЛУ, которое эффективно реализует эти операции.

Пример 2.5. Пусть заданы m=3, D={0, 1, ..., 7} и переменные $X=(011)_2$, $Y=(101)_2$. Тогда $\overline{X}=(100)_2$, $X\&Y=(001)_2$, $X\vee Y=(111)_2$.

Находим, что для произвольных X, Y, Z∈D и поразрядных логических операций конъюнкции &, дизъюнкции $\vee$ и отрицания $\overline{\ }$ справедливы следующие законы:

Таблица 2.1

Законы	Относительно &	Относительно $\vee$
Идемпотентности	$X\&X=X$	$X\vee X=X$
Коммутативности	$X\&Y=Y\&X$	$X\vee Y=Y\vee X$
Ассоциативности	$(X\&Y)\&Z=X\&(Y\&Z)$	$(X\vee Y)\vee Z=X\vee(Y\vee Z)$
Дистрибутивности	$(X\vee Y)\&Z=(X\&Z)\vee(Y\&Z)$	$(X\&Y)\vee Z=(X\vee Z)\&(Y\vee Z)$
Противоречия (тавталогии)	$\overline{X}\&X=0$	$\overline{X}\vee X=I$
Исключения третьего	$\overline{\overline{X}}=X$	$X=\overline{\overline{X}}$

Система функций (2.2) порождает замкнутый класс функций k-значной логики из B_k, который вместе с введенными поразрядными операциями будем рассматривать как алгебру поразрядных операций $\Phi_m=<D_m, \bar{}_m, \&_m, V>$. Очевидно, что алгебра Φ_m изоморфна булевой алгебре B_2 и можно интерпретировать формулы алгебры B_2 в алгебре Φ_m, причем при m=1 алгебра Φ_m вырождается в алгебру B_2.

Пример 2.6. Пусть задана система четырех булевых функций $F=(f_4, f_3, f_2, f_1)$ от трех переменных $X=(X_3, X_2, X_1)$. Установить, принимает ли система функций F значение в B_2, равное 7 при каких-либо допустимых значениях переменных X:

$$\begin{cases} f_1(X)=(\bar{X}_1\&\bar{X}_3)\oplus X_2, \\ f_2(X)=X_2 V(X_1\&X_3), \\ f_3(X)=X_1\&(X_2 V X_3), \\ f_4(X)=X_3\&\bar{X}_2, \end{cases}$$

где $\oplus$ – операция неэквиваленции, определяемая как $X\oplus Y=(\bar{X}\&Y)V(X\&\bar{Y})$.

Рассмотрим систему функций в алгебре Φ_m при $m=2^3$, что соответствует полнократному логическому вычислению. Вычислим значения функций при всевозможных значениях независимых переменных, используя поразрядные булевые операции и представляя значения переменных и функций в форме (2.3) находим:

$$X_1=(01010101)_2, \quad X_2=(00110011)_2, \quad X_3=(00001111)_2,$$

$$\begin{cases} f_1=(\overline{(01010101)_2}\&\overline{(00001111)_2})\oplus(00110011)_2=(10010011)_2, \\ f_2=(00110011)_2 V((01010101)_2\&(00001111)_2)=(00110111)_2, \\ f_3=(01010101)_2\&((00110011)_2 V(00001111)_2)=(00010101)_2, \\ f_4=(00001111)_2\&\overline{(00110011)_2}=(00001100)_2. \end{cases}$$

Анализируя в соответствующих разрядах результаты вычислений определяем, что система функции F в B_2 принимает значение $7=(0111)_2$ в первом и пятом разрядах, что возможно только при X=7 ($X_3=X_2=X_1=1$) или при X=3 ($X_3=0$, $X_2=X_1=1$).

Из примера 2.6 видно, что алгебра Φ_m является математической моделью АЛУ с поразрядными логическими операциями и отражает возможость распараллеливания вычислений системы (2.1) на различных наборах переменных. При этом, кратные вычисления в B_2 системы функций на m наборах переменных соответствуют однократному вычислению изоморфной системы в Φ_m, причем каждая функция вычисляется в отдельности.

§ 8. Арифметико-логическая форма

Арифметико-логическое устройство помимо поразрядных логических операций выполняет арифметические операции над числами с ограниченной разрядностью. Поставим целью повысить эффективность применения АЛУ для кратных вычислений путем использования как поразрядных логических, так и арифметических операций. Для этого рассмотрим следующую систему функций в Φ_m:

$$\{0,\ 1,\ X_i\&X_j,\ X_i+X_j,\ X_i-X_j\}, \qquad (2.4)$$

где + (−) – арифметическая операция сложения (вычитания). Заметим, что в качестве логической операции в (2.4) можно выбрать не только конъюнкцию, но и дизъюнкцию, неэквиваленцию и т.д. В [3] получены рекуррентные выражения для формирования базиса при использовании операции дизъюнкция.

Система функций (2.4) является функционально полной в Φ_m в силу известной теоремы о функциональной полноте [4], так как каждая функция системы (2.2) может быть выражена через функции системы (2.4) следующим образом:

$$\overline{X}_i=1-X_i,\quad X_i\&X_j=X_i\&X_j,\quad X_i\vee X_j=X_i+X_j-X_i\&X_j. \qquad (2.5)$$

Представим функции f_j в базисе (2.4) путем замены операций (2.5). В результате преобразования получим представление каждой функции в арифметико-логической форме $p_j(X)$. После объединения

$p_j(X)$ с помощью производящей функции

$$P(X)=\sum_{j=1}^{q} p_j(X)(I+1)^{j-1}, \qquad (I+1)^0=I, \tag{2.6}$$

и приведения в последней подобных слагаемых получим арифметико-логический полином в алгебре Φ_m

$$P(X)=\sum_{i=0}^{2^n-1} A_i(X_1^{i_1}\&\ldots\&X_n^{i_n})=\sum_{i=0}^{2^n-1} A_i\theta_i(X), \tag{2.7}$$

где A_i – коэффициенты полинома, $A_i\in Z$, Z – множество целых чисел; $(i_n \ldots i_2\ i_1)_2$ – двоичное представление числа i; $\theta_i(X)=X_1^{i_1}\&\ldots\&X_n^{i_n}$ – логическая часть. Определение степенной операции зададим следующим образом

$$X_i^j=\begin{cases} I, \text{ при } j=0, \text{ т.е. переменная } X_i \text{ не входит в } \theta_i(X); \\ X_i, \text{ при } j\neq 0, \text{ т.е. переменная } X_i \text{ входит в } \theta_i(X). \end{cases}$$

Однозначное соответствие между системой (2.1) и порожденным от нее арифметико-логическим полиномом (2.7) в B_2 установлено в [5], однозначное соответствие в Φ_m непосредственно следует из изоморфизма алгебр. В работе [6] рассмотрена алгебра кортежей, где под кортежем понимается упорядоченная совокупность функций вида

$$F(X)=\#_{j=1}^{q} f_j(X)=(f_q(X)\#f_{q-1}(X)\#\ldots\#f_2(X)\#f_1(X)), \tag{2.8}$$

и показано, что в B_2 значения полинома (2.7) и кортежа (2.8), порожденных от одной и той же системы функций совпадают при одинаковых значениях независимых переменных. Это свойство справедливо и в Φ_m, но с тем отличием, что для представления значения каждой функции в кортеже отводится не один, а m двоичных разрядов.

Пример 2.7. Для системы функций из примера 2.6 построим арифметико-логический полином в Φ_m при $m=8$ (в B_2 при $m=1$). Выполнив замену (2.5) и произведя тождественные преобразования в алгебре Φ_m получаем арифметико-логическое представление функций $f_j(X)$:

$$\begin{cases} p_1(X)=1-X_1-X_2-X_3+2(X_1\&X_2)+X_1\&X_3+2(X_2\&X_3)-2(X_1\&X_2\&X_3), \\ p_2(X)=X_2+X_1\&X_3-X_1\&X_2\&X_3, \\ p_3(X)=X_1\&X_2+X_1\&X_3-X_1\&X_2\&X_3, \\ p_4(X)=X_3-X_2\&X_3. \end{cases}$$

Далее объединяем $p_j(X)$ в виде взвешенной суммы (2.6) и после приведения подобных слагаемых получаем полином в Φ_m (в B_2):

$$P(X)=255-X_1+255X_2+16777215X_3+65538(X_1\&X_2)+$$
$$+65793(X_1\&X_3)-16777214(X_2\&X_3)-65794(X_1\&X_2\&X_3)$$
$$\left(P(X)=1-X_1+X_2+7X_3+6(X_1\&X_2)+7(X_1\&X_3)-6(X_2\&X_3)-8(X_1\&X_2\&X_3)\right),$$

где коэффициенты представлены в десятичной системе счисления.

Для $X_1=(01010101)_2$, $X_2=(00110011)_2$, $X_3=(00001111)_2$ значение полинома $P(X)$ в Φ_m равно $(00001100|00010101|00110111|10010011)_2$, что совпадает с результатами из примера 2.6. Для этого необходимо значение полинома рассмотреть как значение кортежа функций $(f_4\#f_3\#f_2\#f_1)$, где каждое значение функции представлено m двоичными разрядами, а значение функции в B_2 на i-м наборе равно i-му разряду значения функции в Φ_m.

Заметим, что если наборы переменных пересекаются, т.е. среди заданных наборов имеются одинаковые, то и результаты кратных вычислений будут содержать одинаковые данные для совпадающих наборов.

Таким образом, вычисление системы булевых функций посредством вычисления арифметико-логического полинома позволяет более полно использовать функциональные возможности арифметико-логического устройства, а вычисление соответствующего арифметико-логического полинома в алгебре поразрядных логических операций позволяет реализовать кратные вычисления системы логических функций.

§ 9. Коэффициенты арифметико-логической формы

Получим оценку значений коэффициентов арифметико-логической формы булевой функции f_j. Приведем способ вычисления коэффициентов по известному характеристическому вектору функции $F=[f_j(0)\ \dots\ f_j(2^n-1)]^T$ на основе дискретного ортогонального преобразования [7]. Из коэффициентов a_{ij} арифметико-логической формы составим вектор коэффициентов $A=[a_{0j}\ \dots\ a_{2^n-1,j}]^T$. Тогда пара дискретного ортогонального преобразования определяет связь векторов A и F следующим образом:

$$\begin{cases} A=K_{2^n}^{-1}\,F; \\ F=K_{2^n}\,A, \end{cases} \qquad (2.9)$$

где K_{2^n} ($K_{2^n}^{-1}$) – матрицы прямого (обратного) конъюнктивного преобразования размерности $2^n\times 2^n$, определяемые по следующему рекуррентному правилу [там же]:

$$K_{2^0}=K_{2^0}^{-1}=[1],\quad K_{2^{t+1}}=\left[\begin{array}{c|c} K_{2^t} & 0 \\ \hline K_{2^t} & K_{2^t} \end{array}\right],\quad K_{2^{t+1}}^{-1}=\left[\begin{array}{c|c} K_{2^t}^{-1} & 0 \\ \hline -K_{2^t}^{-1} & K_{2^t}^{-1} \end{array}\right]. \qquad (2.10)$$

В работе [3] приведено рекуррентное выражение для матрицы $K_{2^t}^{(V)}$ ортогонального преобразования в дизъюнктивном базисе:

$$G_{2^0}=[-1],\quad G_{2^t}=\left[\begin{array}{c|c} 0 & G_{2^{t-1}} \\ \hline G_{2^{t-1}} & -G_{2^{t-1}} \end{array}\right],\quad K_{2^t}^{(V)}=G_{2^t}+\begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & 0 & 1 \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 \end{bmatrix}.$$

Пример 2.8. При n=3 согласно (2.10) получим

$$K_{2^2}^{-1}=\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix},\quad K_{2^2}^{-1}=\left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ \hline -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 1 \end{array}\right],\quad K_{2^3}^{-1}=\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{array}\right],$$

что позволяет вычислить вектор коэффициентов для функции f_1 из

примера 2.6, для которой характеристический вектор F равен $[10010011]^T$:

$$A=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 2 \\ -1 \\ 1 \\ 2 \\ -2 \end{bmatrix}.$$

Пример 2.9. Представим кратные логические вычисления системы булевых функций, заданных первой таблицей и выполним двукратные вычисления на наборах переменных из второй таблицы:

X_2	X_1	X_0	f_3	f_2	f_1	f_0
0	0	0	1	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	1	0	1	0
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

,

Набор	X_2	X_1	X_0
0	1	0	1
1	0	1	1

.

Используя дискретное преобразование получим арифметико-логические формы для каждой из функций системы:

$$\begin{cases} p_0(X) = I-X_2-X_1\&X_0+X_2\&X_1\&X_0, \\ p_1(X) = I-X_1+X_1\&X_0-X_2\&X_1\&X_0, \\ p_2(X) = X_1-X_2\&X_1\&X_0, \\ p_3(X) = I-X_1+X_2\&X_1\&X_0. \end{cases}$$

Выполнив объединение полиномов в соответствии с (2.6) получим:

$$P(\tilde{X})=\sum_{i=0}^{3} p_i(\tilde{X})\cdot(I+1)^i=64p_3+16p_2+4p_1+p_0,$$

$$P(\tilde{X})=207-52\tilde{X}_1-\tilde{X}_2+3(\tilde{X}_1\&\tilde{X}_0)+45(\tilde{X}_2\&\tilde{X}_1\&\tilde{X}_0).$$

Значения функций на разных наборах переменных имеют следующее распределение в разрядной сетке:

$$P(\tilde{X})=\left(f_3^{(1)}f_3^{(0)}\,|\,f_2^{(1)}f_2^{(0)}\,|\,f_1^{(1)}f_1^{(0)}\,|\,f_0^{(1)}f_0^{(0)}\right)_2.$$

Вычисление полинома $P(\tilde{X})$ представим таблицей:

Переменные	Конъюнкции	Полином
$\tilde{X}_2=(01)_2$	$\tilde{X}_1\&\tilde{X}_0=(10)_2$	$P(\tilde{X})=(01\|10\|11\|00)_2$
$\tilde{X}_1=(10)_2$	$\tilde{X}_2\&\tilde{X}_1\&\tilde{X}_0=(00)_2$	$P(3)=(0\ \|1\ \|1\ \|0\ )_2$
$\tilde{X}_0=(11)_2$		$P(5)=(\ 1\|\ 0\|\ 1\|\ 0)_2$

Анализ выражений (2.10) и примера 2.8 показывает, что количество ненулевых элементов в строке матрицы K_{2t}^{-1} на единицу больше числа единиц двоичного представления номера строки, причем половина единиц имеет знак плюс, другая половина – знак минус (кроме первой строки, всегда содержащей одну единицу). Откуда следует, что для коэффициентов a_{ij} $(i=\overline{0,\ 2^n-1})$ арифметико-логической формы $p_j(X)$ булевой функции f_j справедлива следующая оценка:

$$\begin{cases} a_{0j}\in\{0,\ 1\}, \\ a_{ij}\in\{0,\ \pm1,\ \ldots,\ \pm2^{e(i)-1}\}, \quad i\neq0, \\ |a_{ij}|\le2^{e(i)-1}, \quad i\neq0, \end{cases} \tag{2.11}$$

где $e(i)$ – количество единиц в двоичном представлении числа i, $e(i)\le n$. Распространим выражения (2.11) на алгебру Φ_m:

$$\begin{cases} a_{0j}\in\{0,\ I\}, \\ a_{ij}\in\{0,\ \pm1,\ \ldots,\ \pm2^{e(i)-1}\}, \quad i\neq0, \\ |a_{ij}|\le2^{e(i)-1}, \quad i\neq0. \end{cases} \tag{2.12}$$

Из формулы (2.6) видно, что коэффициенты A_i полинома (2.7) представимы в виде:

$$A_i=\sum_{j=1}^{q}a_{ij}(I+1)^{j-1}, \tag{2.13}$$

откуда следует, что эти коэффициенты в Φ_m имеют следующую оценку:

$$\begin{cases} A_0\in\{0,\ I,\ I(I+1),\ I(I+1)^2,\ I\left((I+1)^2+1\right),\ \ldots,\ \dfrac{(I+1)^q-1}{I}\}, \\ A_i\in\{0,\ \pm1,\ \pm2,\ \ldots,\ I,\ I+1,\ \ldots,\ \pm\dfrac{(I+1)^q-1}{I}2^{e(i)-1}\}, \\ |A_i|\le\dfrac{(I+1)^q-1}{I}2^{e(i)-1}. \end{cases} \tag{2.14}$$

При I=1 из оценки (2.14) получаем оценку коэффициентов полинома, описывающего систему функций в B_2:

$$|A_i| \le (2^q-1)2^{e(i)-1}. \qquad (2.15)$$

Знание диапазона возможных значений коэффициентов арифметико-логической формы позволяет оценить объем памяти, необходимый для их хранения. Сравнительный анализ (2.11) и (2.12) показывает, что объем памяти, необходимый для хранения коэффициентов a_{ij}, $(i \neq 0)$, в Φ_m и в B_2 одинаков. Коэффициенты же A_i согласно (2.14) и (2.15) в Φ_m имеют больший диапазон значений и требуют, соответственно, большего объема памяти. Оценим объем памяти, необходимый для хранения коэффициентов арифметико-логической формы.

§ 10. Информационные характеристики полинома

В работе [8] дана информационная оценка сложности вычисления системы булевых функций, определяемой количеством логических элементов. Для информационной оценки сложности реализации системы (2.1) будем использовать информационные характеристики соответствующей арифметико-логической формы (2.7).

Информационная емкость. Информационной емкостью арифметико-логического полинома или количеством информации, заключенным в коэффициентах полинома A_i будем называть величину:

$$I_p = \sum_{i=0}^{2^n-1} \log_2(2|A_i|+1), \text{ [бит]}.$$

Информационная емкость полинома характеризует его информационную сложность и является количественной мерой для сравнения различных полиномов. Величина I_p определяет минимальный объем памяти, необходимый для хранения коэффициентов полинома и может быть использована для проверки эффективности кодирования этих коэффициентов в памяти вычислительного средства.

Оценим информационную емкость полинома в Φ_m для достаточно сложной системы булевых функций, когда в выражении (2.7) присутствуют все конъюнкции, что согласуется с асимптотическими оценками (см. § 4). Предполагаем, что все коэффициенты принимают максимальные значения. В результате получаем

$$I_p = \sum_{i=0}^{2^n-1} \log_2 \left(\frac{(I+1)^q - 1}{I} 2^{e(i)} - 1 \right) \approx 2^n ((q-1)m + n/2). \qquad (2.16)$$

При выводе оценки (2.16) не учтена имеющая место взаимосвязь между коэффициентами, из-за чего полученная оценка информационной емкости является теоретическим пределом и на практике не достигается.

Информационная энтропия. Информационной энтропией арифметико-логического полинома или средним количеством информации, приходящимся на один коэффициент будем называть величину:

$$H_p = \frac{I_p}{2^n} = \frac{1}{2^n} \sum_{i=0}^{2^n-1} \log_2 (2|A_i| + 1), \text{ [бит/коэф.]}.$$

Информационная энтропия полинома характеризует информационные издержки доставки коэффициентов полинома в вычислительное средство и задает их среднюю длину в битах. Величина H_p определяет пропускную способность шины вычислительного средства, необходимую для своевременной доставки коэффициентов. Оценкой информационной энтропии H_p арифметико-логического полинома в соответствии с (2.16) может служить величина

$$H_p = (q-1)m + n/2. \qquad (2.17)$$

Ясно, что интерпретация полученной оценки такая же как и для оценки информационной емкости (2.16).

Объем памяти. Объем памяти M_p, необходимый для хранения коэффициентов может быть подсчитан как:

$$M_p = \sum_{i=0}^{2^n-1} \left\lceil \log_2(2|A_i|+1) \right\rceil, \text{ [бит]},$$

где $\lceil X \rceil$ – наименьшее целое число, равное или большее X.

Объем памяти M_p, превосходящий по своему значению информационную емкость полинома I_p, характеризует неоптимальное кодирование коэффициентов полинома в памяти вычислительного средства.

Пример 2.10. Подсчитаем информационную емкость и энтропию арифметико-логического полинома из примера 2.7 в $\Phi_m(B_2)$.

Полагаем q=4, n=3. Для B_2 (m=1, I=1) имеем:

$$I_p(B_2) \approx 24,06 \text{ бит}, \quad H_p(B_2) \approx 3,0 \text{ бит/коэф.},$$

для Φ_m (m=8, $I=2^m-1=255$):

$$I_p(\Phi_m) \approx 120,60 \text{ бит}, \quad H_p(\Phi_m) \approx 15,08 \text{ бит/коэф.}$$

Полученные результаты показывают, что кратные вычисления в Φ_m дают выигрыш по требуемой пропускной способности шины вычислительного средства в $\frac{mH_p(B_2)}{H_p(\Phi_m)} \approx \frac{8 \times 3,0}{15,08} \approx 1,5$ раза по сравнению с повторением 8 раз однократного вычисления в булевой алгебре.

§ 11. Оценка эффективности кратных вычислений

Известны несколько путей вычисления полинома (2.7) в B_2: непосредственное вычисление по формуле [9] и вычисления по преобразованному полиному, когда последний предварительно сведен в композицию линейных полиномов [10] или в совокупность управляющей конструкции и линейных полиномов [11]. Условия сводимости к линейной форме даны в работе [12]. Линейные полиномы вида

$$P(X) = A_0 + \sum_{i=1}^{n} A_i X_i,$$

удобны тем, что их вычисление в B_2 сводится к суммированию констант, числом не более n, при переменных, не равных нулю.

Количество линейных полиномов равно 2(n+1) от общего количества функций от n переменных 2^{2^n}, что говорит о невозможности представления в виде линейного полинома подавляющего числа функций.

Рассмотрим вычисление полинома (2.7) в Φ_m по первому способу. В качестве вычислительной модели используем вычислительное средство, имеющее в своем составе арифметико-логическое устройство. Так как в произвольном полиноме некоторые из коэффициентов нулевые, то представим полином в виде:

$$P(X)=A_0+\sum_{i=1}^{s} A_i\theta_i(X), \qquad (2.18)$$

где $s\le 2^n-1$; $\theta_i(X)=X_{i_1}\&\ldots\&X_{i_t}$, $t\le n$, $\theta_i(X)\le I$; $P(X)\le(I+1)^q-1$.

Вычисление полинома (2.18) можно выполнить по следующей рекурренной процедуре:

$$P_0(X)=A_0, \quad P_i(X)=P_{i-1}+A_i\theta_i, \quad P_s(X)=P(X),$$

где на одну операцию арифметического сложения и умножения приходится t поразрядных операций конъюнкции. Следуя [9], распараллелим вычисление конъюнкций. При этом возможны два варианта: первый, когда переменные поступают в вычислительное средство последовательно и вычисление всех конъюнкций производится параллельно; и второй, когда переменные поступают параллельно и вычисление конъюнкций выполняется последовательно, но каждая конъюнкция вычисляется сразу, т.е. параллельно по всем своим переменным.

Зададим структуру полинома (2.18) с помощью матрицы вхождения переменных в конъюнкции $L=[l_{ij}]$, $(i=\overline{1,\ s},\ j=\overline{1,\ n})$,

$$l_{ij}=\begin{cases} 0, & \text{если переменная } X_j \text{ не содержится в } \theta_i(X); \\ 1, & \text{если переменная } X_j \text{ содержится в } \theta_i(X), \end{cases}$$

Вектор $X=[X_j]$, $(j=\overline{1,\ n})$, будем рассматривать как входной набор n переменных, $X_j\in D$.

Последовательное поступление переменных. При последовательном

поступлении переменных, объем промежуточной памяти вычислительного средства составляет Q=sm бит для хранения конъюнкций и P=qm бит – для текущего значения полинома; объем программы U равен I_p бит – для вектора $A=[A_i]$ и L=ns бит – для матрицы $L=[l_{ij}]$, где I_p – информационная емкость вычисляемого полинома. В итоге, общий объем памяти для вычисления арифметико-логического полинома в Φ_m при последовательном поступлении данных составит

$$M_{\text{пс}}=Q+U+L+I_p+P=(n+m)s+qm+I_p, \quad [\text{бит}],$$

откуда видно, что с увеличением кратности вычислений m объем памяти $M_{\text{пс}}$ растет линейно, пропорционально сложности вычисления s и количеству вычисляемых функций q.

Параллельное вычисления вектора конъюнкций $\theta=[\theta_1, \theta_2, \ldots, \theta_s]$ может быть выполнено по следующей рекуррентной процедуре:

$$\theta(0)=\breve{I}, \quad \theta(i)=\theta(i-1)\&(\breve{X}_i \vee [\overline{\breve{l}_i}]),$$

где $\theta(i)$ – вектор конъюнкций θ на i-й итерации, $i=\overline{1, n}$; $\breve{I}$ – вектор, получаемый мультипликацией вектора I s раз; $[\breve{l}_i]$ – вектор, получаемый растяжением i-й строки матрицы L m раз; $\breve{X}_i$ – вектор, получаемый мультипликацией вектора X_i s раз.

Учитывая, что на вход вычислительного средства можно подавать сразу строки матрицы L в инверсном виде, а мультипликацию и растяжение векторов выполнять в процессе трассировки данных, то за $T_1=2n$ тактов на r разрядном АЛУ, $r \geq sm$, вычисляются s конъюнкций θ в виде вектора $\theta(n)$.

Вычисление значения полинома сводится к умножению матриц A и $\theta(n)$:

$$P(X)=[A_0, A_1, \ldots, A_s] \times [\theta_0, \theta_1, \ldots, \theta_s]^T, \quad \theta_0=I,$$

что можно выполнить по следующей рекуррентной процедуре:

$$P(0)=A_0 I, \quad P(i)=P(i-1)+A_i \times \theta_i, \quad i=\overline{1, s}.$$

Учитывая, что операцию умножения A_i на θ_i можно выполнить за не более чем m тактов работы АЛУ, а операцию сдвига – путем

мультиплексирования и трассировки данных, за один такт, то за $T_2=(s+1)m$ тактов получим значения полинома $P(X)$.

Таким образом, общее время вычисления полинома (2.18) при последовательном поступлении переменных и параллельном вычислении конъюнкций составит

$$T_{пс}=T_1+T_2=2n+(s+1)m, \quad [\text{такт}].$$

Из приведенного выражения видно, что с увеличением кратности вычислений m время вычисления $T_{пс}$ растет линейно, пропорционально сложности вычисления полинома s и не зависит от количества вычисляемых функций q.

Параллельное поступление переменных. При параллельном поступлении переменных требуется объем промежуточной памяти $Q=nm$ бит – для значения текущей конъюнкции, $P=qm$ бит – для текущего значения полинома, а объем программы U, как и при последовательном поступлении переменных, составляет I_p бит – для матрицы A и $I=ns$ бит – для матрицы L. Общий объем памяти при параллельном поступлении переменных в итоге составит

$$M_{пр}=Q+U+L+I_p=(m+s)n+qm+I_p, \quad [\text{бит}].$$

С увеличением кратности вычислений m объем памяти $M_{пр}$ растет линейно, пропорционально количеству переменных n и количеству вычисляемых функций q. Заметим, что объем памяти программы U фактически совпадает с информационной емкостью полинома I_p.

Последовательное вычисление конъюнкций θ_i над вектором переменных $\bar{X}=[X_1, X_2, \ldots, X_n]$, $X_j \in D$, может быть выполнено по следующей процедуре:

$$\theta_i=\bar{\bar{X}}\&[\hat{L}_i], \quad i=\overline{1, s}.$$

где $[\hat{L}_i]$ – вектор, получаемый растяжением i-го столбца матрицы L m раз.

Учитывая, что на вход вычислительного средства можно подавать сразу переменные в инверсном виде, а растяжение векторов выполнять

в процессе трассировки данных, то за $T_1=s$ тактов на r разрядном АЛУ, $r \geq nm$, последовательно вычисляются s конъюнкций θ_i.

В связи с тем, что конъюнкции вычисляются последовательно и промежуточная память для хранения всех конъюнкций не предусмотрена, целесообразно по мере получения конъюнкций выполнять вычисление полинома. Количество тактов, необходимых для умножения векторов A и θ в этом случае составит $T_2=(s+1)m$.

В итоге, общее время вычисления полинома при параллельном поступлении переменных и последовательном вычислении конъюнкций составляет

$$T_{пр}=T_1+T_2=s+(s+1)m, \quad [такт],$$

откуда видно, что с увеличением кратности вычислений m время вычисления $T_{пр}$ растет линейно, пропорционально сложности вычисления s, и не зависит от количества переменных n и функций q.

Комплексная эффективность. Определим комплексную эффективность кратных вычислений в Φ_m, для чего воспользуемся оценкой времени однократного вычисления полинома (2.18) из работы [9]: $T=2s+n$. В табл. 2.2 приведены полученные результаты, когда эффективность по сложности вычислений η оценивалась по требуемой пропускной способности шины вычислительного средства с учетом оценок для $H_p(B_2)$ и $H_p(\Phi_m)$, полученных из выражения (2.17).

Таблица 2.2

Относительно последовательных во времени кратных вычислений	Относительно параллельных в пространстве кратных вычислений
$\xi=\frac{m(2s+n)}{s+(s+1)m}$	$\xi=\frac{(2s+n)}{s+(s+1)m}$
$\eta=\frac{mH_p(B_2)}{H_p(\Phi_m)}$	$\eta=m\frac{H_p(B_2)}{H_p(\Phi_m)}$
$\vartheta=\frac{2ms+mn}{(m+1)s+m}\cdot\frac{m(q+n/2-1)}{(q-1)m+n/2}$	$\vartheta=\frac{2ms+mn}{(m+1)s+m}\cdot\frac{m(q+n/2-1)}{(q-1)m+n/2}$

Комплексная эффективность рассмотренных процедур больше единицы, но ее асимптотическое поведение (ϑ стремится к единице) позволяет заключить, что реализация кратных вычислений в алгебре поразрядных операций может быть отнесена к параллельным в пространстве кратным вычислениям.

Выводы

1). Алгебра поразрядных логических операций является обобщенной математической моделью арифметико–логического устройства, а арифметико–логический полином в этой алгебое – форма представления системы булевых функции, позволяющая описать кратные логические вычисления.

2). В рассмотренных процедурах кратных вычислений объем внутренней памяти и время вычислений растут, пропорционально кратности, а объем памяти программы определяется информационной емкостью вычисляемого арифметико–логического полинома.

3). Кратные вычисления позволяют до определенного предела наращивать интенсивность использования арифметико–логического устройства, при этом энтропия вычисляемого арифметико–логического полинома не должна превосходить пропускную способность внешней шины вычислительного средства.

4). При вычислении арифметико–логического полинома в алгебре поразрядных логических операций реализуются параллельные в пространстве кратные вычисления.

Глава 3. Кратные вычисления в многозначной логике

Как было установлено ранее, кратные логические вычисления могут быть выражены в замкнутом классе функций k-значной логики, порождаемом арифметическими и поразрядными логическими операциями и представлены в виде арифметико-логического полинома. При реализации этого полинома выполняются параллельные в пространстве кратные вычисления, что позволяет более эффективно использовать современные многоразрядные процессоры.

Опираясь на работы [1, 2] обобщим кратные вычисления на случай многозначной логики, исследуем возможность представления кратных вычислений в различных формах, выявим закономерности и особенности такого представления, оценим эффективность реализации.

Как показано в § 6 представление кратных вычислений системы логических функций $Y=F(X)$,

$$F(X)=(f_q(X), \ldots, f_1(X)), \quad X=(X_n, \ldots, X_1), \qquad (3.1)$$

может быть получено преобразованием этой системы в некоторую форму $\tilde{F}(\tilde{X})$, ортогонально объединяющую n переменных, q функций и m наборов:

$$\tilde{F}(\tilde{X})=\tilde{F}\left(\mathop{\#}_{r=1}^{m} X^{(r)}\right)=\mathop{\#}_{j=1}^{q} \tilde{f}_j\left(\mathop{\#}_{r=1}^{m}\mathop{\#}_{i=1}^{n} X_i^{(r)}\right), \qquad (3.2)$$

где $\tilde{f}_j$ ($\tilde{F}$) – некоторая форма представления функции (системы функций). В соответствии с (3.2) $\tilde{f}_j$ эквивалентна функции от $n \cdot m$ переменных. При построении $\tilde{f}_j$ необходимо учесть, что вхождение переменных X_i определяется видом функции f_j, а вхождение m значений каждой из этих перемеменных $X_i^{(r)}$ определяется способом их ортогонального объединения. Форма $\tilde{F}$, в свою очередь, синтезируется путем ортогонального объединения функций $\tilde{f}_j$.

Задачу описания кратных вычислений условно разобьем на три этапа: ортогональное объединение переменных, ортогональное

объединение функций, ортогональное объединение наборов переменных. Под ортогональным объединением переменных понимается представление произвольной функции в некотором функциональном полном базисе операций (в виде формулы), под ортогональным объединением функций – синтез формы совместного описания системы, а под ортогональным объединением наборов – получение аналитической конструкции формы для множества наборов переменных. Из всех возможных форм необходимо выбрать те формы, которые наиболее компактно описывают кратные вычисления.

§ 12. Матричная форма кратных вычислений

Представим кратные вычисления в матричной форме, для чего рассмотрим кортежи наборов переменных $\tilde{X}$ и результатов вычислений $\tilde{Y}$ в виде матриц:

$$\tilde{X}=\begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1m} \\ X_{21} & X_{22} & \cdots & X_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ X_{n1} & X_{n2} & \cdots & X_{nm} \end{bmatrix}_r \qquad \tilde{Y}=\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1m} \\ Y_{21} & Y_{22} & \cdots & Y_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ Y_{q1} & Y_{q2} & \cdots & Y_{qm} \end{bmatrix}$$

где X_{ir} – значение переменной X_i из набора переменных $X^{(r)}$; Y_{jr} – значение логической функции f_j на наборе $X^{(r)}$,

$$X^{(r)}=\mathop{\#}_{i=1}^{n} X_{ir}.$$

Будем искать преобразование матрицы $\tilde{X}$ в матрицу $\tilde{Y}$ в виде линейного матричного выражения:

$$\tilde{Y}=\tilde{C}+\tilde{F}\times\tilde{X}, \qquad (3.3)$$

или, с учетом размерностей матриц,

$$\left[Y_{jr}\right]_{q\times m}=\left[C_{ji}\right]_{q\times m}+\left[F_{ji}\right]_{q\times n}\times\left[X_{ir}\right]_{n\times m'}$$

В свою очередь матрицы преобразования $\tilde{F}$ и констант $\tilde{C}$ могут быть представлены как:

$$\tilde{F}=\begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1n} \\ F_{21} & F_{22} & \cdots & F_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ F_{q1} & F_{q2} & \cdots & F_{qn} \end{bmatrix}, \qquad \tilde{C}=\begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ C_{q1} & C_{q2} & \cdots & C_{qm} \end{bmatrix},$$

где F_{ji} – некоторые натуральные числа. Чтобы найти взаимосвязь между матрицей $\tilde{F}$ и системой функций (3.1) рассмотрим значение элементов Y_{jr} матрицы $\tilde{Y}$, которая получается в результате вычисления (3.3):

$$Y_{jr}=C_{jr}+\sum_{i=1}^{n} F_{ji}\times X_{ir}, \qquad j=\overline{1,\ q}, \qquad r=\overline{1,\ m}.$$

Анализ последнего выражения показывает, что вид матрицы $\tilde{F}$ не зависит от кратности вычислений m, а ее элементы F_{ji} являются не чем иным, как коэффициентами линейного разложения функции f_j по переменным X_i.

$$f_j(X)=C_j+\sum_{i=1}^{n} F_{ji}X_i.$$

Откуда следует, что матричная форма может быть использована для описания кратных вычислений систем логических функций, допускающих представление в виде линейного арифметического полинома. В свою очередь матрица констант $\tilde{C}$ содержит m одинаковых столбцов $[C_j]$ размерностью $1\times q$,

$$\tilde{C}=[C_j]\otimes[1\ 1\ \ldots\ 1],$$

каждый из которых равен q коэффициентам C_j линейного разложения функции исходной системы. Здесь знак $\otimes$ обозначает операцию кронекеровского произведения матриц.

Условие представления произвольной функции f_j в виде линейного арифметического полинома дано в работе [3]:

$$f_j(i)=\sum_{u=0}^{n-1} f_j(k^u)i_{n-u}+f_j(0)\left(1-\sum_{v=1}^{n} i_v\right),$$

где i_j – j-й разряд k-ичного представления i, а $f_j(i)$ – значение

функции f_j в точке i.

Таким образом, матричная форма кратных вычислений удобна компактностью описания, но годится только для систем логических функций, допускающих представление в виде линейного арифметического полинома. Заметим, что вычисление (3.3) равносильно повторению m раз вычисления линейного полинома. Следовательно реализация кратных вычисления в матричной форме может быть отнесена к последовательным во времени кратным вычислениям. С другой стороны, мощность класса функций, которые могут быть представлены в виде линейных полиномов невелика и составляет k+2n от общего количества k^{k^n} функций k-значной логики [3]. Этим обусловлены ограничения в использовании линейных полиномиальных форм и матричной формы кратных вычислений.

§ 13. Арифметическая форма кратных вычислений

Пусть требуется вычислить с кратностью m произвольную систему q логических функций от n переменных в алгебре k-значной логики B_k:

$$F(X)=\#_{j=0}^{q-1} f_j(X), \qquad X=\#_{i=0}^{n-1} X_i, \tag{3.4}$$

где $f_j(X)$ – логические функции; X_i – независимые переменные; $f_j, X_i \in D$, $D=\{0, 1, \ldots, I\}$, $I=k-1$. Нумерацию функций и переменных для удобства будем производить начиная с нуля.

Ортогональное объединение функций f_j и переменных X_i будем осуществлять на основе взаимно однозначного представления произвольного натурального числа в k-ичной позиционной системе счисления:

$$\left.\begin{aligned} F(X)&=\sum_{j=0}^{q-1} f_j(X)k^j=\left(f_{q-1}(X) \ \ldots \ f_1(X) \ f_0(X)\right)_k, \\ X&=\sum_{i=0}^{n-1} X_i k^i=\left(X_{n-1} \ \ldots \ X_1 \ X_0\right)_k \end{aligned}\right\} \tag{3.5}$$

где $f_j(X)$ – j–й разряд q–разрядного k–ичного представления значений системы $F(X)$; X_i – i–й разряд n–разрядного k–ичного представления X; $X \in [0, k^n-1] \subset N$, $F(X) \in [0, k^q-1] \subset N$; N – множество натуральных чисел.

Выбор арифметических операций для ортогонального объединения функций и переменных диктуется эффективной аппаратной реализацией этих операций в современных вычислительных средствах. Дополнительно заметим, что выражения (3.5) представляют некоторое натуральное число, которое, в свою очередь, может быть взаимно однозначно представлено в системе счисления с произвольным основанием. Это дает возможность реализовать полученные результаты на вычислительных средствах, использующих кодирование данных в алфавите с произвольным числом знаков, в том числе и двоичном.

Ортогональное объединение функций. Для ортогонального объединения функций системы (3.4), следуя [4], представим их в арифметической форме:

$$p_j(X) = \sum_{i=0}^{k^n-1} a_{ij} X_{n-1}^{i_{n-1}} \ldots X_1^{i_1} X_0^{i_0}, \quad j=\overline{0, q-1}, \qquad (3.6)$$

где a_{ij} – i–й коэффициент арифметического полинома для j–й функции; X_i^j – независимая переменная X_i в арифметической степени j; $i=(i_{n-1} \ldots i_0)_k$ – k–ичное представление числа i.

Приведем способ конструирования арифметического полинома по известным характеристическим векторам F_j функций системы (3.4) на основе дискретного ортогонального преобразования [5]. Из коэффициентов a_{ij} арифметического полинома (3.6) составим вектор коэффициентов $A_j=[a_{j0}\ a_{j1} \ldots a_{j(k^n-1)}]$. Тогда пара ортогональных преобразований в арифметическом базисе D_{k^n} определяет связь векторов следующим образом:

$$\begin{cases} A_j = D_{k^n} \times F_j; \\ F_j = D_{k^n}^{-1} \times A_j, \end{cases} \qquad j=\overline{0, q-1} \qquad (3.7)$$

где D_{k^n} ($D^{-1}_{k^n}$) – матрицы прямого (обратного) ортогонального преобразования размерности $k^n \times k^n$, которые определяются по рекуррентному правилу:

$$\begin{cases} D^{-1}_{k^{n+1}} = D^{-1}_{k} \otimes D^{-1}_{k^n}, \\ D_{k^{n+1}} = D_{k} \otimes D_{k^n} \end{cases}$$

где $\otimes$ – знак операции кронекеровского произведения матриц; матрицы D_k и D^{-1}_k связаны матричным уравнением (условие ортогональности)

$$D_k \times D^{-1}_k = E_k.$$

Здесь E_k – единичная матрица размерности $k \times k$, а (i,j)-й элемент d^{-1}_{ij} матрицы D^{-1}_k определяется как $d^{-1}_{ij} = i^j$, $(i,j = \overline{0, k-1})$, т.е. как i в арифметической степени j.

В работе [6] показано, что ортогональное объединение выражений (3.6) возможно в виде взвешенной суммы, в результате чего получают совместное описание функций $f_j(X)$ в виде единого арифметического полинома. Вычисление полученного полинома дает значения каждой из функций системы на каком-то одном наборе переменных, реализовано в булевой алгебре [7] и получило название параллельного логического вычисления. По результатам, полученным в § 11 эти формы могут быть использованы только для реализации последовательных во времени или параллельных в пространстве кратных вычислений.

Для описания кратных вычислений использование только ортогонального объединения логических функций недостаточно, так как это объединение учитывает только вхождения независимых переменных, не затрагивая вхождения в итоговый арифметический полином наборов переменных.

Ортогональное объединение наборов переменных. Для ортогонального объединения функций и наборов переменных запишем (3.6) для всех m наборов $X^{(r)} = \#_{i=0}^{n-1} X_{ir}$, $r = \overline{0, m-1}$:

$$p_j(X^{(r)}) = \sum_{i=0}^{k^n-1} a_{ij} X_{(n-1)r}^{i_{n-1}} \cdots X_{1r}^{i_1} X_{0r}^{i_0}, \quad j=\overline{0,q-1} \qquad (3.8)$$

и объединим полученные выражения в виде взвешенной суммы с коэффициентами $k^{\varphi(r,j)}$, порожденными некоторой функцией двух переменных φ:

$$P(\tilde{X}) = \sum_{r=0}^{m-1} \sum_{j=0}^{q-1} p_j(X^{(r)}) k^{\varphi(r,j)} \qquad (3.9)$$

Для обеспечения ортогональности объединения выражений для $p_j(X^{(r)})$ в соответствии с (3.9), на функцию φ накладываются ограничения. Дискретную функцию $\varphi \in N$ n переменных, заданную в области $U \subset N^n$ будем называть весовой, если в различных точках области определения U значения функции различны, т.е. $\forall X, Y \in U$, если $X \neq Y$, то и $\varphi(X) \neq \varphi(Y)$. Весовая функция φ задает размещение значений $p_j(X^{(r)})$ в разрядной сетке результата. Номер k-ичного разряда, в котором располагается значение j-й функции от r-го набора переменных равен $\varphi(r, j)$. Различный выбор весовой функции φ позволяет реализовать различное размещение значений функций вычисляемой системы в разрядной сетке результата.

Подставим в выражение (3.9) представление функций в форме (3.8) и получим арифметическую форму, описывающую кратные вычисления для m наборов переменных $\tilde{X}$, $\tilde{X} = \#_{r=0}^{m-1} X^{(r)} = \#_{r=0}^{m-1} \#_{i=0}^{n-1} X_{ir}$:

$$P(\tilde{X}) = \sum_{r=0}^{m-1} \sum_{j=0}^{q-1} k^{\varphi(r,j)} \sum_{i=0}^{k^n-1} a_{ij} X_{(n-1)r}^{i_{n-1}} \cdots X_{1r}^{i_1} X_{0r}^{i_0}.$$

С целью упрощения последнего выражения потребуем, чтобы весовая функция φ двух переменных представлялась как сумма двух функций μ и ν, каждая из которых, в свою очередь, является весовой:

$$\varphi(r, j) = \mu(r) + \upsilon(j), \qquad (3.10)$$

и что, очевидно, можно сделать различными способами. Тогда выражение для $P(\tilde{X})$ с учетом разделения переменных r и j имеет вид:

$$P(\tilde{X})=\sum_{i=0}^{k^n-1} A_i(\mu)\theta_i(\tilde{X},\ \upsilon), \qquad (3.11)$$

где $A_i(\mu)=\sum_{j=0}^{q-1} a_{ij}k^{\mu(j)}$, $\quad \theta_i(\tilde{X},\ \upsilon)=\sum_{r=0}^{m-1} k^{\upsilon(r)}X^{i_{n-1}}_{(n-1)r}\cdots X^{i_1}_{1r}\ X^{i_0}_{0r}$.

Выражение (3.11) является представлением кратных вычислений в арифметической форме и соответствует разложению системы логических функций в ряд по ортогональным функциям $\theta_i(\tilde{X})$ от $m\cdot n$ независимым переменным $\tilde{X}$.

Пример 3.1. Представим в арифметической форме в трехзначной логике (k=3) двукратные вычисления (m=2) системы функций, заданных в таблице истинности:

X_1	X_0	f_1	f_0
0	0	0	0
0	1	0	1
0	2	0	2
1	0	0	1
1	1	1	1
1	2	2	2
2	0	0	2
2	1	2	2
2	2	1	2

Представим функции системы в виде арифметических полиномов, для чего построим матрицы ортогонального преобразования D_{3^2} и $D^{-1}_{3^2}$, а качестве характеристических векторов функций возьмем столбцы из таблицы истинности:

$$D_3^{-1}=\begin{bmatrix}1&0&0\\1&1&1\\1&2&4\end{bmatrix},\quad D_9^{-1}=\begin{bmatrix}1&0&0\\1&1&1\\1&2&4\end{bmatrix}\otimes\begin{bmatrix}1&0&0\\1&1&1\\1&2&4\end{bmatrix},$$

$$D_9^{-1}=\left[\begin{array}{ccc|ccc|ccc}1&0&0&0&0&0&0&0&0\\1&1&1&0&0&0&0&0&0\\1&2&4&0&0&0&0&0&0\\\hline1&0&0&1&0&0&1&0&0\\1&1&1&1&1&1&1&1&1\\1&2&4&1&1&1&1&2&4\\\hline1&0&0&2&0&0&4&0&0\\1&1&1&2&2&2&4&4&4\\1&2&4&2&4&8&4&8&16\end{array}\right],\quad D_9=\frac{1}{4}\left[\begin{array}{ccc|ccc|ccc}4&0&0&0&0&0&0&0&0\\-6&8&-2&0&0&0&0&0&0\\2&-4&2&0&0&0&0&0&0\\\hline-6&0&0&8&0&0&-2&0&0\\9&-12&3&-12&16&-4&3&-4&1\\-3&6&-3&4&-8&4&-1&2&-1\\\hline2&0&0&-4&0&0&2&0&0\\-3&4&-1&6&-8&2&-3&4&-1\\1&-2&1&2&4&-2&1&-2&1\end{array}\right],$$

Вектор коэффициентов полиномов $p_0(X)$ и $p_1(X)$ получим в соответствии с (3.7)

$$A_0=\left[0\ 1\ 0\ 1\ -\frac{5}{2}\ 1\ 0\ 1\ -\frac{1}{2}\right]^T, \quad A_1=\left[0\ 0\ 0\ 0\ 1\ \frac{3}{4}\ 0\ \frac{3}{4}\ -\frac{3}{4}\right]^T,$$

что определяет следующий вид искомых полиномов:

$$\begin{cases} p_0(X)=X_1+X_2-\frac{5}{2}X_1X_2+X_1^2X_2+X_1X_2^2-\frac{1}{2}X_1^2X_2^2 \\ p_1(X)=X_1X_2+\frac{3}{4}X_1^2X_2+\frac{3}{4}X_1X_2^2-\frac{3}{4}X_1^2X_2^2 \end{cases}$$

Выберем весовые функции: $\mu(j)=mj$ и $\upsilon(r)=r$, что обеспечит смежное расположение значений одной функции на m наборах переменных и последовательное расположение этих значений для всех функций в разрядной сетке результата вычислений:

$$P(\tilde{X})=\left(f_1(X^{(1)})\ f_1(X^{(0)})\ f_0(X^{(1)})\ f_0(X^{(0)})\right)_3.$$

Вектор коэффициентов арифметической формы будет равен:

$$A=\left[0\ 1\ 0\ 1\ \frac{13}{2}\ \frac{31}{4}\ 0\ \frac{31}{4}\ \frac{25}{4}\right]^T,$$

а в итоге получим искомую арифметическую форму:

$$P(\tilde{X})=\frac{1}{4}\left(4(X_{00}+3X_{01})+4(X_{10}+3X_{11})+26(X_{00}X_{10}+3X_{01}X_{11})+\right.$$
$$\left.+31(X_{00}^2X_{10}+3X_{01}^2X_{11})+31(X_{00}X_{10}^2+3X_{01}X_{11}^2)+25(X_{00}^2X_{10}^2+3X_{01}^2X_{11}^2)\right).$$

Из примера 3.1 видно, что кратные вычисления в арифметической форме дают незначительный выигрыш по времени вычисления по сравнению с повторением вычислений при одинарной кратности. Этот выигрыш определяется тем, что умножение на коэффициенты A_i выполняются один раз для всех наборов независимых переменных $X^{(r)}$, но объединение наборов переменных связано с выполнением умножения на значности алгебры k в степенях $1, 2, 3, \ldots, (r-1)$.

§ 14. Арифметико-логическая форма кратных вычислений

Исследуем кратные вычисления при представлении функций в арифметико-логической форме. Для ортогонального объединения функций $f_j(X)$ исходной системы (3.4) представим их в виде арифметико-логических полиномов:

$$P_j(X)=\sum_{i=0}^{k^n-1} a_{ij}\left(C_i \circ X_{n-1}^{i_{n-1}} \circ \ldots \circ X_0^{i_0}\right), \quad j=\overline{0,\ q-1}, \qquad (3.12)$$

где a_{ij} – i-й коэффициент полинома для j-й функции; C_i – некоторые произвольные константы, $C_i \in D$; $\circ$ – логические операции; X_i^j – переменная X_i в степени j; $i=(i_{n-1} \ldots i_1\ i_0)_k$ – n разрядное k-ичное представление числа i. Представление логической функции в форме (3.12) осуществляется на основе дискретного ортогонального преобразования. Подробно методика синтеза аналитической конструкции таких форм изложена в главе 4.

Для ортогонального объединения функций и наборов переменных запишем (3.12) для всех m наборов переменных $X^{(r)}$, $r=\overline{0,\ m-1}$:

$$P_j(X^{(r)})=\sum_{i=0}^{k^n-1} a_{ij}\left(C_i \circ X_{(n-1)r}^{i_{n-1}} \circ \ldots \circ X_{0r}^{i_0}\right)=\sum_{i=0}^{k^n-1} a_{ij}\theta_i(X^{(r)}),$$

где θ_i – логическая часть формы, $\theta_i(X^{(r)})=C_i \circ X_{(n-1)r}^{i_{n-1}} \circ \ldots \circ X_{0r}^{i_0}$.

Объединим полученные выражения в виде взвешенной суммы с коэффициентами $k^{\varphi(r,j)}$, порожденными весовой функцией φ:

$$P(\tilde{X})=\sum_{r=0}^{m-1}\sum_{j=0}^{q-1} k^{\varphi(r,j)}\left\{\sum_{i=0}^{k^n-1} a_{ij}\theta_i(X^{(r)})\right\}.$$

После разделения переменных φ в соответствии с (3.10) имеем

$$P(\tilde{X})=\sum_{r=0}^{m-1} k^{\mu(r)}\sum_{j=0}^{q-1} k^{\upsilon(j)}\left\{\sum_{i=0}^{k^n-1} a_{ij}\theta_i(X^{(r)})\right\},$$

и в результате тождественных преобразований получим:

$$P(\tilde{X})=\sum_{i=0}^{k^n-1}\left(\sum_{j=0}^{q-1}k^{\upsilon(j)}a_{ij}\right)\times\left\{\sum_{r=0}^{m-1}k^{\mu(r)}\theta_i(X^{(r)})\right\},$$

или

$$P(\tilde{X})=\sum_{i=0}^{k^n-1}\tilde{A}_i(\upsilon)\tilde{\theta}_i(\tilde{X},\ \mu), \qquad (3.13)$$

где $\tilde{A}_i(\upsilon)=\sum_{j=0}^{q-1}k^{\upsilon(j)}a_{ij}$, а $\tilde{\theta}_i(\tilde{X},\ \mu)=\sum_{r=0}^{m-1}k^{\mu(r)}\theta_j(X^{(r)})$.

Выражение (3.13) описывает широкий класс полиномиальных форм кратных вычислений, отличающихся различным выбором весовых функций μ и υ, а также различным заданием логических и степенных операций. При использовании в качестве логических операций арифметического умножения, в качестве степенных – операции возведения в арифметическую степень и установив $C_i=1$, получаем арифметическую форму кратных вычислений (3.11), а задав при $k=2$ степенные операции в виде

$$X_i^j=\begin{cases}-1, & \text{если } j=1 \text{ и } X_i=1;\\ 1, & \text{если } j=0 \text{ или } X_i=0,\end{cases} \quad \text{или} \quad X_i^j=\begin{bmatrix}1 & 1\\ 1 & -\end{bmatrix},$$

получаем форму кратных вычислений в базисе Уолша (см. табл. 1.2).

Оценим комплексную эффективность вычисления выражения (3.13). Для однократных вычислений (3.13) может быть записано как

$$P(X)=\sum_{i=0}^{k^n-1}A_i\theta_i(X), \qquad (3.14)$$

что потребует использования АЛУ с разрядностью не менее q, а время вычисления составит $s(\tau(n)+\tau_\times+\tau_+)$, где s – сложность полинома или количество коэффициентов A_i, не равных нулю; $\tau(n)$ – время вычисления логической части $\theta_i(X)$, τ_+ ($\tau_\times$) – время выполнения операций сложения (умножения). С учетом этого, эффективность вычисления (3.13) по времени по сравнению с параллельным в пространстве кратным вычислением, когда происходит m-кратное дублирование вычислительного средства, составит

$$\xi=\frac{s(\tau(n)+\tau_{\times}+\tau_{+})}{s(\tilde{\tau}(n)+\tau_{\times}+\tau_{+})}\approx\frac{\tau(n)}{\tilde{\tau}(n)},\quad \eta=\frac{qm}{qm}=1,\quad \vartheta=\xi\eta\approx\frac{\tau(n)}{\tilde{\tau}(n)},\qquad (3.15)$$

где $\tilde{\tau}(n)$ – среднее время вычисления $\tilde{\theta}_i$.

Анализ последнего выражения показывает, что для реализации параллельных во времени кратных вычислений необходимо обеспечить выполнение условия $\tau(n)>\tilde{\tau}(n)$, что представляется технически неосуществимым, т.к. требуется реализовать логической части полинома при кратных вычислениях за время, меньшее чем время однократных вычислений. Основой для реализации параллельных во времени кратных вычисления может служить только уменьшение сложности вычислений ν_m (см. § 5).

Для эффективной реализации параллельных в пространстве кратных вычислений необходимо обеспечить равенство времен вычисления $\tau(n)$ и $\tilde{\tau}(n)$. Заметим, что при однократных вычислениях используется m АЛУ с разрядностью q, в исследуемом случае – одно АЛУ с разрядностью qm. Поставим целью распараллелить вычисление $\tilde{\theta}_i$ по всем наборам переменных, что обеспечит комплексную эффективность, равную единице.

Поразрядные логические операции. Для дальнейшего упрощения полученной формы (3.13) используем порязрядные логические операции, определенные в алгебре $B_{k'}$ что обеспечит параллельные в пространстве вычисления логической части полинома на наборах переменных. Если операция $\odot$ является поразрядной, т.е. $\forall X\in D$, $\forall Y\in D$,

$$X\odot Y=\left(\sum_{i=0}^{n-1}X_i k^i\right)\odot\left(\sum_{i=0}^{n-1}Y_i k^i\right)=\sum_{i=0}^{n-1}(X_i\odot Y_i)k^i,$$

где $\odot$ – логическая операция $\circ$ или операция возведения в степень из (3.12), то выражение (3.13) может быть представлено как:

$$P(\tilde{X})=\sum_{i=0}^{k^n-1}\tilde{A}_i(\upsilon)\left(\tilde{C}_i\circ\tilde{X}_{n-1}^{\tilde{i}_{n-1}}\circ\ldots\circ\tilde{X}_0^{\tilde{i}_0}\right),\qquad (3.16)$$

где

$$\left.\begin{array}{l} \tilde{A}_i = \sum\limits_{j=0}^{q-1} k^{\upsilon(j)} a_{ij}, \quad \tilde{A}_i \in Z_k, \quad a_{ij} \in Z \\ \tilde{C}_i = \sum\limits_{r=0}^{m-1} k^{\mu(r)} C_{ir}, \quad \tilde{C}_i \in N_k, \quad C_i \in D \end{array}\right\} \quad i=\overline{0,\ 2^n-1},$$

$$\left.\begin{array}{l} \tilde{X}_j = \sum\limits_{r=0}^{m-1} k^{\mu(r)} X_{jr}, \quad \tilde{X}_j \in N_k, \quad X_{jr} \in D \\ \tilde{i}_j = \sum\limits_{r=0}^{m-1} k^{\mu(r)} i_{jr}, \quad \tilde{i}_j \in N_k, \quad i_j \in D \end{array}\right\} \quad j=\overline{0,\ n-1},$$

Выражение (3.16) описывает m-кратные вычисления системы функций (3.4) и представляет собой арифметико-логический полином, заданный относительно поразрядных логических и степенных операции. Таким образом показана справедливость следующего утверждения.

Утверждение 3.1. Кратные вычисления системы логических функций могут быть эффективно реализованы путем вычисления арифметико-логического полинома в базисе поразрядных логических и степенных операций.

Из (3.16) при $k=2$, $\mu=r$ и $\upsilon=mj$ получаем арифметико-логический полином в алгебре Φ_m (см. главу 2), где в качестве логических операций используются поразрядная булевая операция конъюнкции, а степенные операции определены как в (1.10).

Выражение (3.16) позволяет выполнить структурную минимизацию формы кратных вычислений путем выбора весовых функций, а также реализовать логические вычисления с переменной кратностью (изменяемой от вычисления к вычислению). Коэффициенты $\tilde{A}_i$ в явном виде не зависят от кратности m и путем соответствующего выбора весовой функции υ можно получить форму, которая не зависит от кратности вычислений. Для вычислений с переменной кратностью достаточно по другому "конструировать" $\tilde{X}_j$, $\tilde{i}_j$ и $\tilde{C}_i$, оставив без измененения коэффициенты полинома $\tilde{A}_i$.

Пример 3.2. Пусть задана система логических функций из примера 3.1. Выполнить трехкратные вычисления системы на наборах переменных из таблицы:

Набор	X_1	X_0
0	2	0
1	0	1
2	1	2

Представим функции в арифмметико–логической форме. Для этого используем базис поразрядных операций, состоящий из обобщенной конъюнкция $X\&Y=\min(X,Y)$ и обобщенная неэквиваленция $X\oplus Y=\max(X,Y)-\min(X,Y)$, используемой в качестве степенной операции. Матрица прямого дискретного преобразования D_9, характеристическая матрица F и матрица коэффициентов A выглядят следующим образом (здесь и далее коэффициенты приводятся в десятичной системе счисления):

$$D_9=\frac{1}{8}\left[\begin{array}{ccc|ccc|ccc} 1&-2&-1&-2&2&2&-1&2&1\\ -2&4&-2&2&-2&2&2&-4&2\\ -1&-2&1&2&2&-2&1&2&-1\\ \hline -2&2&2&4&-2&-4&-2&2&2\\ 2&-2&2&-2&0&-2&2&-2&2\\ 2&2&-2&-4&-2&4&2&2&-2\\ \hline -1&2&1&-2&2&2&1&-2&-1\\ 2&-4&2&2&-2&2&-2&4&-2\\ 1&2&-1&2&2&-2&-1&-2&1 \end{array}\right],\quad F=\left[\begin{array}{c} 0\ 0\\ 1\ 0\\ 2\ 0\\ \hline 1\ 0\\ 1\ 1\\ 2\ 2\\ \hline 2\ 0\\ 2\ 2\\ 2\ 1 \end{array}\right],\quad A=D_9\times F=\frac{1}{8}\left[\begin{array}{cc} 4&11\\ 4&-4\\ 4&1\\ \hline 4&-4\\ 0&-6\\ 4&8\\ \hline 4&1\\ 4&8\\ -4&-5 \end{array}\right].$$

Выберем весовые функции $\mu(r)=q^r$, $\upsilon(j)=j$, что обеспечит смежное расположение значений функций f_j при последовательном расположение значений системы F(X) на различных наборах переменных $X^{(r)}$ в разрядной сетке:

$$P(\tilde{X})=\left(f_1(X^{(2)})\ f_0(X^{(2)})\ \ldots\ f_1(X^{(0)})\ f_0(X^{(0)})\right)_{3'}$$

а также независимость коэффициентов $\tilde{A}_i$ от кратности вычисления m. Конструируя полином в соответствии с (3.16) получим:

$$P(\tilde{X})=37\tilde{X}_0^0\tilde{X}_1^0-9\tilde{X}_0^1\tilde{X}_1^0+7\tilde{X}_0^2\tilde{X}_1^0-9\tilde{X}_0^0\tilde{X}_1^1-18\tilde{X}_0^1\tilde{X}_1^1+28\tilde{X}_0^2\tilde{X}_1^1+7\tilde{X}_0^0\tilde{X}_1^2+28\tilde{X}_0^1\tilde{X}_1^2-19\tilde{X}_0^2\tilde{X}_1^2.$$

Для кратных вычислений системы необходимо вычислить $P(\tilde{X})$ при

$$\tilde{X}_1=\left(02|00|01\right)_{3'},\qquad \tilde{X}_0=\left(00|01|02\right)_{3'}$$

т.е. значения переменной $X_i^{(r)}$ на каждом наборе r дополняются нулями до двух троичных цифр и записываются последовательно в разрядной сетке переменной $\tilde{X}_i$. В результате вычисления полинома имеем:

$$P(\tilde{X})=\left(20\,|\,10\,|\,22\right)_3.$$

Выделяя последовательно по два троичных разряда результата получаем значения функций на заданных наборах переменных.

§ 15. Абсолютная эффективность кратных вычислений

Как было показано в § 14, вычисления арифметико-логического полинома в базисе поразрядных операций реализует параллельные в пространстве кратные вычисления. Понятно, что комплексная эффективность в этом случае всегда равна единице, т.к. для сравнения выбирается однократное вычисление той же формы.

Поставим целью получить абсолютную оценку эффективности кратных вычислений. Для сравнения выберем табличную форму вычисления системы функций. Табличный способ вычислений, по видимому, является самым быстродействующим, т.к. требует всего $\tau_т=q$ операций для выборки одного значения системы, а его вычислительная сложность определяется затратами $v_т=qk^n$, что соответствует объему памяти для хранения таблицы. С другой стороны (см. § 3), асимтотическая оценка количества операций при реализации системы функций в некотором фиксированном базисе стремится к величине k^n при достаточно большом количестве функций q и переменных n. К тому же, до того как будет решена задача представления системы функций в некоторой форме, эта система, как правило, задается в виде таблицы истинности. Все это обосновывает возможность использования табличного способа вычислений в качестве базового для абсолютной оценки эффективности.

Процедура кратных вычислений. Для абсолютной оценки эффективности кратных вычислений зададим процедуру вычисления полинома (3.16). Известны несколько путей вычисления полинома: непосредственное вычисление по формуле [8, 9] и вычисления по преобразованному полиному, когда последний предварительно сведен в композицию линейных полиномов [10, 11].

Рассмотрим вычисление полинома (3.15) непосредственно по формуле. Для этого представим весовую функцию φ как сумму двух весовых функций $\mu(r)=qr$ и $\nu(j)=j$. Так как в произвольном полиноме некоторые из коэффициентов нулевые, а X_i^0 могут быть исключены из аналитической конструкции путем соответствующего задания степенных операций, то полином (3.16) представим в виде:

$$P(\tilde{X})=\sum_{i=0}^{s}\tilde{A}_i\left(\tilde{C}_i\circ\tilde{X}_{i_1}^{\tilde{j}_1}\circ\ldots\circ\tilde{X}_{i_t}^{\tilde{j}_t}\right)=\sum_{i=0}^{s}\tilde{A}_i\tilde{\theta}_i(\tilde{X}), \qquad (3.17)$$

где s – вычислительная сложность полинома, $s<k^n$; $\tilde{\theta}_i(\tilde{X})$ – логическая часть полинома, $\tilde{\theta}_i(\tilde{X})=\tilde{C}_i\circ\tilde{X}_{i_1}^{\tilde{j}_1}\circ\ldots\circ\tilde{X}_{i_t}^{\tilde{j}_t}$.

Вычисление полинома (3.17) можно выполнить по следующей рекуррентной процедуре:

$$\begin{cases} P_0(\tilde{X})=\tilde{A}_0; \\ P_i(\tilde{X})=P_{i-1}(\tilde{X})+\tilde{A}_i\times\tilde{\theta}_i, \quad (i=\overline{1,s}); \\ P(\tilde{X})=P_s(\tilde{X}). \end{cases} \qquad (3.18)$$

Из (3.18) видно, что на одну операцию арифметического сложения и умножения приходится $2t$ поразрядных операций. В работе [12] даны процедуры однократных вычислений в булевой алгебре при параллельном (за один такт) вычислении θ_i. В работе [13] рассмотрен замкнутый класс функций k-значной логики при $k=2^m-1$ и показана возможность реализации m-кратных вычислений в булевой алгебре путем вычисления арифметико-логического полинома в базисе поразрядных операций, а в работе [14] приведены соответствующие процедуры кратных вычислений.

Информационные характеристики. Для оценки эффективности кратных вычислений в соответствии с (3.18) нам потребуется оценка среднего значение модуля коэффициентов полинома $|\tilde{A}|_{cp}$ и среднее количество k-ичных разрядов H_p, необходимых для хранения одного такого коэффициента или энтропия полинома (см. § 10).

Связь характеристического вектора F системы функций и вектора коэффициентов формы A определяется дискретным преобразованием (1.12). По методике, описанной в § 9 оценим среднее значение модуля коэффициентов A_i:

$$|\tilde{A}_i|_{cp}=\left|\sum_{j=0}^{k^n-1} d_{ij}F(j)\right|_{cp}=F_{cp}\max(d_i^+,\ d_i^-),$$

где d_{ij} – (i,j)-й элемент матрицы D_{k^n}; $|x|$ – модуль числа x; d_i^+ (d_i^-) – сумма абсолютных значений положительных (отрицательных) элементов i-й строки D_{k^n}; F_{cp} – усредненное значение системы функций, $F_{cp}=k^q/2$. Выполнив усреднение по всем $|\tilde{A}_i|_{cp}$ получим оценку для $|\tilde{A}|_{cp}$:

$$|\tilde{A}|_{cp}=\frac{1}{k^n}\sum_{i=0}^{k^n-1}|\tilde{A}_i|_{cp}=\frac{k^{q-n}}{2}\max(d^+,\ d^-),$$

где d^+ (d^-) – сумма абсолютных значений положительных (отрицательных) элементов матрицы D_{k^n}. Энтропию полинома H_p и его информационную емкость I_p определим так:

$$H_p=\log_k\left(2|\tilde{A}|_{cp}+1\right),\qquad I_p=sH_p. \tag{3.19}$$

Время и сложность кратных вычислений. Вычисление всех $\tilde{\theta}_i$ потребует в среднем $s\cdot\tilde{\tau}(n)$ тактов работы R разрядного k-ичного АЛУ ($R\geq mq$), где $\tilde{\tau}(n)$ – среднее количество операций (тактов), необходимых для вычисления $\tilde{\theta}_i$,

$$\tilde{\tau}(n)=\frac{1}{k^n}\sum_{t=1}^{n}(2t-1)(k-1)^{t-1}C_n^t,$$

здесь C_n^t – число сочетаний из n элементов по t, а $(k-1)^{t-1}C_n^t$ –

количество различных θ_i, имеющих t логических операций и t–1 операцию возведения в степень. Умножение $\tilde{A}_i$ на $\tilde{\theta}_i$ выполняется в среднем за H_p тактов, а сложение $\tilde{A}_i \cdot \tilde{\theta}_i$ с текущим значением полинома – за s тактов. Таким образом, среднее время кратных вычислений составляет:

$$\tau_p = s\left(1+H_p+\tilde{\tau}(n)\right), \quad [\text{такт}],$$

и не зависит от кратности вычислений m. Если разрядность АЛУ R меньше чем $m \cdot q$, введем в выражение для τ_p поправочный коэффициент:

$$\tau_p = s\left\rceil \frac{mq}{R} \right\lceil \left(1+H_p+\tilde{\tau}(n)\right), \quad [\text{такт}],$$

где $\rceil x \lceil$ – наименьшее целое, равное или большее x.

Аппаратные затраты v_p или объем памяти, необходимый для хранения коэффициентов $\tilde{A}_i$ можно оценить величиной $v_p = sH_p = I_p$. Объем памяти в предложенной схеме вычислений является минимальным по отношению к различному выбору весовых функций, а сами коэффициенты не зависят от кратности и совпадают с коэффициентами арифметико-логического полинома для однократных вычислений.

Комплексная эффективность. Оценим абсолютную комплексную эффективность кратных вычислений. Абсолютная эффективность по сложности вычисления η_a и по времени вычисления ξ_a определим следующим образом:

$$\eta_a = \frac{v_т}{v_p} = \frac{qk^n}{I_p} = \frac{qk^n}{sH_p},$$

$$\xi_a = \frac{m\tau_т}{\tau_p} = \frac{mq}{s\left\rceil \frac{mq}{R} \right\lceil \left(1+H_p+\tilde{\tau}(n)\right)} \approx \frac{R}{s\left(1+H_p+\tilde{\tau}(n)\right)},$$

а абсолютную комплексную эффективность выразим так:

$$\vartheta_a = \eta_a \xi_a = \frac{qk^n}{sH_p} \cdot \frac{R}{s\left(1+H_p+\tilde{\tau}(n)\right)},$$

и после тождественных преобразований с учетом (3.19) получим

$$\vartheta_a = \frac{qRk^n}{s^2 H_p \left(1+H_p+\tilde{\tau}(n)\right)}. \qquad (3.20)$$

где q – количество функций в системе; R – k-ичная разрядность АЛУ; s – вычислительная сложность полинома; n – количество независимых переменных; I_p (H_p) – информационная емкость (энтропия); $\tilde{\tau}(n)$ – среднее время вычисления логической части.

Для эффективной реализации кратных вычислений необходимо обеспечить выполнение условия $\vartheta_a > 1$, откуда, в соответствии с (3.20), получаем оценку для эффективной вычислительной сложности:

$$s_{эф} = \sqrt{\frac{qRk^n}{H_p\left(1+H_p+\tilde{\tau}(n)\right)}}, \qquad (3.21)$$

задающую граничное значение сложности, при которой комплексная эффективность еще равна единице.

Пути повышения эффективности. Из выражения (3.21) видно, что эффективная вычислительная сложность $s_{эф}$ не зависит от кратности вычислений m и слабо зависит от количества функций в системе q. Другой важный вывод можно сделать относительно разрядности АЛУ: увеличение разрядности АЛУ в четыре раза позволяет эффективно вычислять полиномы с удвоенной вычислительной сложностью.

Оценка (3.21) также показывает, что повышение комплексной эффективности может быть достигнуто за счет уменьшения энтропии полинома: уменьшение энтропии H_p в четыре раза позволяет эффективно реализовать полиномы с более чем удвоенной вычислительной сложностью. Уменьшение энтропии H_p возможно как путем совместной минимизация функций, так и путем подбора весовых функций φ при синтезе аналитической конструкции в соответствии с (3.16). Наименьшее значение энтропии H_p достигается при независимости коэффициентов полинома от кратности вычислений, например, когда $\upsilon(j)=j$.

На рис. 3.1 приведена зависимость эффективной вычислительной сложности $s_{эф}$ от количества переменных и от значности логики. В качестве логических операций использовалась обобщенная конъюнкция:

X&Y=min(X, Y), степенная операция задана как

$$X^{Y}=\begin{cases} 1, & \text{при } X>Y; \\ 0, & \text{при } X\le Y, \end{cases}$$

а разрядность R пересчитывалась для каждого значения k таким образом, чтобы она была эквивалентна 64 битовому АЛУ:

$$R=64/(\log_2 k).$$

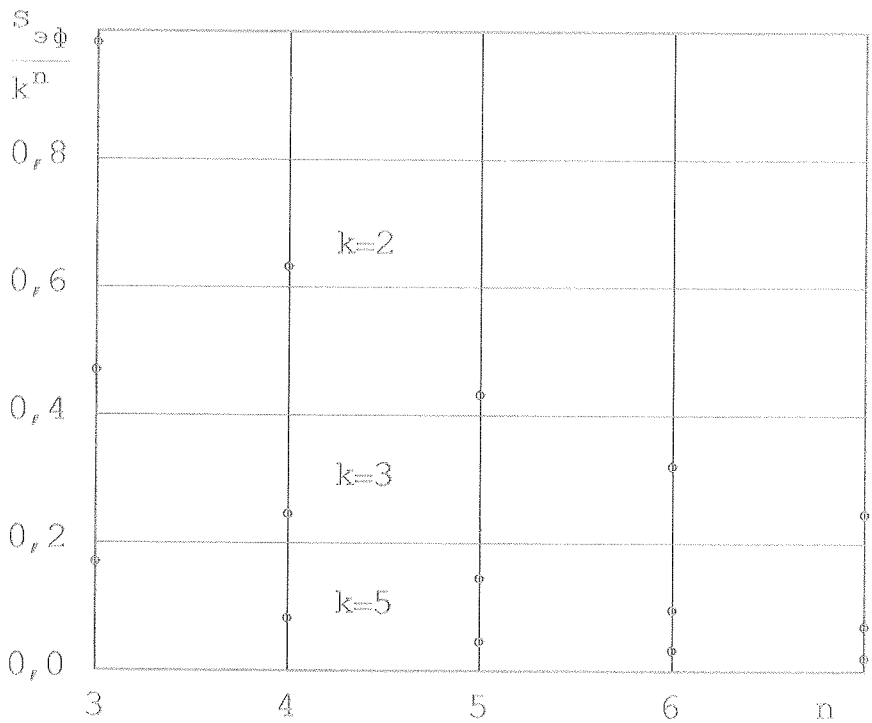


Рис. 3.1. Эффективная вычислительная сложность полинома

Анализ рис. 3.1 показывает, что количество арифметико-логических полиномов, имеющих эффективную реализацию, значительно превосходит количество линейных полиномов, для которых вычислительная сложность равна количеству независимых переменных n, а количество функций, допускающих такое представление – k+2n. Это позволяет, например, строить более эффективные процедуры логических вычисления путем композиции полиномов, имеющих некоторую заданную эффективность при последовательных в пространстве кратных вычислениях. Открытым, правда, остается вопрос оценки количества логических функций, имеющих заданную вычислительную сложность, а также вопросы синтеза и минимизации арифметико-логических форм.

Выводы

1). Реализация кратных вычислений в арифметико–логической форме в базисе поразрядных операций позволяет более интенсивно использовать арифметико–логическое устройство, а сами вычисления могут быть отнесены к параллельным в пространстве кратным вычислениям.

2). Для реализации параллельных во времени кратных вычислений необходимо обеспечить комплексную эффективность больше единицы. Основой для этого может служить только уменьшение сложности вычислений, определяемой аппаратными затратами v_m.

3). Выбор весовой функции, используемой при ортогональном объединении логических функций и наборов переменных позволяет получать различные формы кратных вычислений, в том числе и не зависящие от кратности.

4). Абсолютная комплексная эффективность кратных вычислений не зависит от кратности, а ее повышение возможно как путем увеличения разрядности АЛУ, так и при уменьшении энтропии арифметико–логического полинома.

5). Количество арифметико–логических полиномов, имеющих эффективную реализацию значительно превосходит количество линейных полиномов, что позволяет выполнить последовательные в пространстве кратные вычисления путем композиции полиномов, имеющих заданную эффективность реализации.

Глава 4. Аналитические методы синтеза полиномиальных форм

Как было показано в главе 3, вычисление арифметико-логического полинома в базисе поразрядных операций реализует параллельные в пространстве кратные вычисления. Понятно, что комплексная эффективность, определенная по отношению к однократному вычислению той же арифметико-логической формы всегда равна единице. На практике возникает потребность в оценке времени кратных вычислений для сравнения с другими формами представления той же системы функций. В этом случае абсолютная эффективность зависит только от эффективности однократных вычислений и определяется минимальностью представления логической функции (системы логических функций) в некотором базисе операций.

Известные методы реализации логических вычислений своей структурой повторяют формульное представление, а основой повышения эффективности логического вычисления служит расширение базиса операций. Традиционная архитектура ЭВМ не предусматривает специализированных устройств для логических вычислений, а современное состояние информационной технологии отражает неадекватность используемых методов обработки данных структуре вычислительных средств [1, 2]. В связи с этим наблюдается закономерный интерес к разработке новых форм представления логических данных. В последнее время находит все большее применение метод расширения форм, основанный на преобразовании Фурье в дискретных базисах [3, 4]. Практический интерес представляют полиномиальные формы, которые имеют однородную алгебраическую структуру и хорошо реализуются средствами современной микроэлектроники.

Наименее изученными остаются вопросы, связанные с расширением форм представления логических функций с учетом особенностей

используемого вычислительного средства, а также синтез форм в расширенном базисе операций. Основываясь на ранее полученных результатах [5, 6], изложим обобщение метода синтеза полиномиальных форм. Суть подхода состоит в том, что логические функции представляется в форме, учитывающей особенности логических данных, отражающей функциональные возможности вычислительного средства и синтезируемой в расширенном базисе операций.

§ 16. Информационные оценки при синтезе

Исходными данными для представления логических вычислений в некотором базисе операций служит таблица логических данных, которая интерпретируется как таблица истинности системы функций. Поставим задачу перейти от таблицы истинности к такой форме совместного описания системы, в которой используются операции δ_t из расширенного множества операций $\Lambda=\{\delta_0, \delta_1, \ldots, \delta_p\}$. Например, это множество может совпадать с системой команд вычислительного средства, сами операции могут иметь наименьшее время выполнения или быть предпочтительными по иным соображениям.

Под логическим вычислением будем понимать процесс нахождения значения произвольной системы q логических функции от n независимых переменных в алгебре k-значной логики B_k:

$$\begin{cases} F(X)=(f_{q-1}(X), \ldots, f_0(X))=\#_{j=0}^{q-1} f_j(X); \\ X=(X_{n-1}, \ldots, X_0)=\#_{j=0}^{n-1} X_i, \end{cases} \qquad (4.1)$$

где $f_j(X)$ – логические функции; X_i – независимые переменные.

Система функций (4.1) однозначно определяется таблицей истинности $T=[t_{ij}]$ – матрицей размерности $k^n \times q$. Элементы матрицы T обладают тем свойством, что каждый из них может принимать любое значение в диапазоне от 0 до I, где I=k–1. Таким образом,

информационная емкость (см. § 10) системы функций (4.1) в табличном представлении может быть определена как $I_т=qk^n\log_2 k$, [бит]. Информационная емкость $I_т$ задает количество степеней свободы, которые можно использовать для задания произвольной системы (4.1).

Очевидно, что метод синтеза формы P(X) совместного описания системы функций должен обеспечивать сохранение информационной емкости табличного способа, т.е.

$$I_p \geq I_т = qk^n \log_2 k. \qquad (4.2)$$

Только в этом случае преобразование форм может быть взаимно однозначным и количество степеней свободы при задании F(X) будет соответствовать количеству степеней свободы при задании P(X).

Произвольную форму P(X) можно определить некоторым характеристическим вектором, определяющим ее вид (коэффициенты формы). Например, для табличного задания характеристический вектор представляет собой столбец таблицы истинности, для арифметических форм – вектор коэффициентов. Выражение (4.2) задает требования для метода синтеза аналитической конструкции формы P(X) и соответствует использованию некоторого ортогонального преобразования коэффициентов.

Для представления конкретной системы функций необходимо среди всех возможных форм выбрать те формы, которые имеют наименьшую информационную емкость I, т.е.

$$I=\min_{(p)}(I_p) \leq I_т. \qquad (4.3)$$

Это связано с тем, что абсолютная эффективность кратных вычислений по времени ξ всегда меньше или равна единице (см. § 15). Следовательно, для обеспечения комплексной эффективности $\vartheta>1$ необходимо увеличить эффективность по сложности вычисления η, которая определяется информационной емкостью формы I.

Ограничимся рассмотрением линейных ортогональных преобразований, когда элементы характеристических векторов связаны

линейными соотношениями в классе некоторых операций сложения и умножения. Используемые операции сложения и умножения не обязательно должны совпадать с традиционным их определением в арифметике натуральных (целых) чисел. Достаточно, чтобы эти операции образовывали кольцо на некотором подмножестве натуральных (целых) чисел, а преобразование характеристических векторов не уменьшало информационную емкость табличной формы $I_т$.

§ 17. Методика синтеза полиномиальных форм

Пара дискретного ортогонального преобразования в базисе D удовлетворяет требованию (4.2) и определяется матричными соотношениями вида:

$$\begin{cases} A=D\times F; \\ F=D^{-1}\times A, \end{cases} \qquad (4.4)$$

где F – характеристический вектор функции длины k^n, A – вектор коэффициентов той же длины, D (D^{-1}) – квадратная матрица прямого (обратного) преобразования с размерностью $k^n\times k^n$, × – знак операции умножения матриц.

Вектор исходных данных F является столбцом таблицы истинности функции. По спектру A с точностью до нормирующего множителя синтезируется аналитическое выражение:

$$p(X)=\sum_{i=0}^{k^n-1} a_i\theta_i(X), \qquad p(X)\in(\overline{0,\ I}), \qquad (4.5)$$

где a_i – i-й коэффициент формы, θ_i – арифметико-логическое выражение, определяемое правилом формирования базиса, X – логические переменные.

Задача конструирования формы (4.5) соответствует разложению некоторой логической функции f(X) в ряд по заданным ортогональным функциям $\theta_i(X)$ и сводится к вычислению спектра A вектора логических

данных F в базисе D. При этом значения функции f(X) и значения арифметико-логической формы p(X) совпадают во всех k^n точках области определения.

Разнообразие форм p(X) для одной и той же функции f(X) определяется возможностью получения различных полных систем ортогональных функций $\{\theta_i(X)\}$. По соотношению (4.5) синтезируются логические и арифметические формы. При синтезе логических форм суммирование выполняется по модулю k, где k – простое число, а операция умножения заменяется на некоторую логическую операцию. При синтезе арифметических форм используются операции арифметического сложения и умножения, в связи с чем снимаются ограничения на значение k.

Выбор различных систем функций $\{\theta_i(X)\}$ позволяет получать полиномиальные и неполиномиальные формы [1]. Если аналитическое выражение (4.5) представляется в виде полинома от переменных X_i, то такую форму относят к классу полиномиальных, в противном случае форма относится к классу неполиномиальных. Наибольшее распространение получила неполиномиальная форма Хаара [7, 8]. Для неполиномиальных форм характерна высокая сложность и нерегулярность аналитической конструкции, что ограничивает возможности ее модификации применительно к конкретным условиям синтеза. Полиномиальные формы обладают регулярной структурой и обеспечивают широкие возможности по учету особенностей исходной функции. Приведем обобщение метода синтеза аналитической конструкции полиномиальных форм из работы [5].

Постоение базиса дискретного преобразования. Не ограничивая общности, выражение (4.5) можно представить в полиномиальном виде:

$$p(X)=\sum_{i=0}^{k^n-1} a_i\left(C_i\delta_n X_{n-1}^{i_{n-1}}\delta_{n-1}\ \dots\ \delta_0 X_0^{i_0}\right) \qquad (4.6)$$

где a_i – коэффициенты формы; C_i – некоторые произвольные константы, $C_i\in\overline{(0,\ k-1)}$; δ_t – арифметические или логические операции (возможно

и совпадающие при различных t), входящие в некоторую систему двуместных операции $\Lambda=\{\delta_0, \delta_1, \ldots, \delta_p\}$, относительно которых ищется аналитическое выражение; $X_i^{i_t}$ – независимая переменная X_i в степени i_t; $i=(i_{n-1}, \ldots, i_0)_k$ – представление числа i в n разрядной k–ичной позиционной системе счисления:

$$i=\sum_{j=0}^{n-1} i_j k^j, \quad i_j \in \overline{(0, k-1)},$$

В выражении (4.6) предполагается, что порядок выполнения операций справа налево при приоритетном выполнении операции возведения в степень, а задание логических и степенных операций осуществляется таким образом, чтобы область значения предыдущей операции совпадала или включалась в область определения следующей.

В этом случае матрицы дискретного преобразования D и D^{-1} могут быть получены следующим образом.

1). Задается ядро дискретного преобразования в виде определений степенных операций для каждой переменной X_t и представляет собой совокупность матриц W_t, $(t \in \overline{0, n-1})$, с размерностью k×k и элементами

$$w_t(i, j)=i^j, \quad (i, j \in \overline{0, k-1}), \qquad (4.7)$$

где i – номер строки, j – номер столбца. При задании матриц W_t требуется, чтобы строки (столбцы) были линейно независимы относительно используемых операций сложения и умножения.

2). Строится матрица обратного дискретного преобразования D^{-1} по рекуррентному правилу:

$$G_0=W_0, \quad G_{t+1}=W_t \otimes_t G_t, \quad D^{-1}=C_n \delta_n G_{n-1}, \quad (t \in \overline{0, n-1}), \qquad (4.8)$$

где $\otimes_t$ – обобщенная операция кронекеровского произведения матриц, выполняющаяся относительно операции δ_t; C_n – матрица констант, состоящая из k^n одинаковых строк k^n произвольных констант из области определения операции δ_n. Обобщенную операцию кронекеровского произведенияи $\otimes_t$ определим следующим образом:

$$W_t \otimes_t G_t = \left[\begin{array}{c|c|c|c} w_t(0,0)\delta_t G_t & w_t(0,1)\delta_t G_t & \cdots & w_t(0,I)\delta_t G_t \\ \hline w_t(1,0)\delta_t G_t & w_t(1,1)\delta_t G_t & \cdots & w_t(1,I)\delta_t G_t \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline w_t(I,0)\delta_t G_t & w_t(I,1)\delta_t G_t & \cdots & w_t(I,I)\delta_t G_t \end{array}\right], \qquad (4.9)$$

В соответствии с (4.8) и (4.9) элементы матрицы D^{-1} имеют вид:

$$d_{ij}^{-1} = c_i \delta_n j_{n-1}^{i_{n-1}} \delta_{n-1} \cdots \delta_0 j_0^{i_0},$$

где $i=(i_{n-1} \cdots i_0)_k$ и $j=(j_{n-1} \cdots j_0)_k$ соответственно k–ичное представления чисел i и j.

При построении G_{t+1} следят, чтобы строки (столбцы) были линейно независимы. Выбор логических констант c_i производится таким образом, чтобы определитель D^{-1} был отличен от нуля (матрица G_{n-1}, например, может содержать нулевую строку или столбец).

3). Матрица D прямого дискретного преобразования должна удовлетворять условию ортогональности (4.2) и получается обращением матрицы D^{-1}, т.е.

$$D \times D^{-1} = E_{k^n},$$

где E_{k^n} – единичная матрица размерности $k^n \times k^n$. Вычисление определителей при обращении D^{-1}, а также проверка линейной зависимости строк (столбцов) выполняется в классе используемых в выражении (4.6) операций сложения и умножения.

Пример 4.1. Представим функцию $F=[0\ 2\ 2\ 1\ 0\ 2\ 2\ 1\ 1]^T$ трехзначной логики от двух переменных в полиномиальной форме. Искомый полином имеет вид:

$$p(X) = \sum_{i=0}^{3^2-1} a_i \left(c_i \delta_1 X_1^{i_1} \delta_0 X_0^{i_0} \right).$$

Определим степенные и логическую операции в виде матриц:

$$X_0^j = (X_0 + j)_{\mathrm{mod}\ 3} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}, \qquad X_1^j = (X_0 - j)_{\mathrm{mod}\ 3} = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{bmatrix},$$

$$X \delta_0 Y = XVY = \max(X,Y) = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 2 \\ 2 & 2 & 2 \end{bmatrix}.$$

В соответствии с (4.8) получим матрицу обратного преобразования:

$$D^{-1}=\begin{bmatrix}0&1&2\\1&2&0\\2&0&1\end{bmatrix}\otimes_0\begin{bmatrix}0&2&1\\1&0&2\\2&1&0\end{bmatrix}=\left[\begin{array}{ccc|ccc|ccc}0&1&2&1&1&2&2&2&2\\1&2&0&1&2&1&2&2&2\\2&0&1&2&1&1&2&2&2\\\hline1&1&2&0&1&2&1&1&2\\1&2&1&1&2&0&1&2&1\\2&1&1&2&0&1&2&1&1\\\hline2&2&2&1&1&2&0&1&2\\2&2&2&1&2&1&1&2&0\\2&2&2&2&1&1&2&0&1\end{array}\right].$$

Так как матрица D^{-1} не содержит нулевых строк (столбцов), определим операцию δ_1 и константы C_i таким образом, чтобы они отсутствовали в аналитической конструкции, например: $C_i=0$, $X\delta_1 Y=\max(X,Y)$. После обращения D^{-1} имеем:

$$D=\frac{1}{315}\begin{bmatrix}-169&-4&26&144&9&99&-29&31&-44\\-4&26&-169&9&99&144&31&-44&-29\\26&-169&-4&99&144&9&-44&-29&31\\144&9&99&-324&-99&-144&144&9&99\\9&99&144&-99&-144&-324&9&99&144\\99&144&9&-144&-324&-99&99&144&9\\-29&31&-44&144&9&99&-169&-4&26\\31&-44&-29&9&99&144&-4&26&-169\\-44&-29&31&99&144&9&26&-169&-4\end{bmatrix}.$$

Умножив характеристический вектор F на матрицу D получим вектор коэффициентов $A=[1\ 0\ -1\ 0\ 0\ 1\ 0\ 0\ 0]^T$, откуда искомый арифметико-логический полином представим так:

$$p(X)=\sum_{i=1}^{8}a_i\left(X_1^{i_1}\vee X_0^{i_0}\right)=\left(X_1^0\vee X_0^0\right)-\left(X_1^0\vee X_0^2\right)+\left(X_1^1\vee X_0^2\right),$$

а с учетом определений операций можно записать:

$$p(X)=X_1\vee X_0-X_1\vee(X_0+2)_{\mathrm{mod}\ 3}+(X_1-1)_{\mathrm{mod}\ 3}\vee(X_0+2)_{\mathrm{mod}\ 3}.$$

Для вычисления логической части полученного полинома требуется три операции ($\tilde{\tau}(n)=3$), вычислительная сложность полинома s равна трем, а информационная энтропия H_p – единице (в троичной системе). В итоге, абсолютная коплексная эффективность полученной формы в соответствии с (3.20) будет равна:

$$\vartheta_a=\frac{qRk^n}{s^2H_p\left(1+H_p+\tilde{\tau}(n)\right)}=\frac{1\cdot9\cdot3^2}{3^3(1+3)}=\frac{9}{4}>1,$$

где разрядность $R=9$ и выбрана достаточной для полнократных

вычислений, а также учтено, что при вычислении полинома операция умножения не выполняется.

Дополнительные ограничения при синтезе. Логические вычисления в форме (4.5) сводятся к вычислению выражения (4.6). С целью повышения эффективности вычислений на выбор базиса накладывают дополнительные ограничения, которые возможны по причине определенной произвольности в выборе матриц ядра W_t, операций δ_t и логических констант c_i.

Среди таких ограничений – возможность факторизации базиса, т.е. представления матриц прямого (обратного) преобразования в виде произведения слабозаполненных матриц, что позволяет построить быстрый алгоритм дискретного преобразования [9, 10].

Иногда важным является уменьшение сложности формирования базиса, поскольку это отражается на программных и аппаратных средствах. Варьируя базис при неизменных входных данных можно получить множество спектров, некоторые из которых являются более предпочтительными по условию решаемой задачи или содержать много нулевых компонентов [11]. В последнем случае решается задача минимизации функции в заданном классе операций Λ.

Можно также ввести ограничение на выбор базиса, связанное с минимизацией коэффициентов формы, что приведет к уменьшению памяти, необходимой для их хранения, или включить в множество Λ операции, реализованные с наибольшей эффективностью.

На практике получили развитие и имеют широкое распростанение методы вычисления с помощью арифметических полиномов [14, 15]. Выражение (4.6) дает возможность свести вычисления в k-значной логике к вычислению арифметико-логического полинома в булевой алгебре путем соответствующего выбора степенных функций. В этом случае ядро дискретного преобразования задается в виде булевых матриц, а логические вычисления сводятся к выполнению логических

операций над совокупностью промежуточных булевых переменных, полученных в результате возведения в степень исходных переменных.

§ 18. Мультипликативные формы

Формирование базиса дискретного ортогонального преобразования сопряжено со значительными вычислительными трудностями, которые связаны с необходимостью обращения матриц большой размерности (см. пример 4.1). В связи с чем на практике в качестве логических операций δ_t используется некоторая мультипликативная операция, совпадающая по определению с операцией арифметического умножения. В булевой алгебре такой операцией является конъюнкция [12], в многозначной логике – обычное арифметическое умножение [13].

Использование мультипликативных операций позволяет значительно упростить формирование базиса дискретного преобразования. Это связано с интересным свойством операции кронекеровского произведения матриц относительно арифметического умножения. Для произвольных матриц $A_1, A_2, \ldots, A_n$, для которых существуют обратные матрицы $A_1^{-1}, A_2^{-1}, \ldots, A_n^{-1}$ справедливо выражение:

$$(A_1 \otimes A_2 \otimes \ldots \otimes A_n)^{-1} = A_1^{-1} \otimes A_2^{-1} \otimes \ldots \otimes A_n^{-1}, \qquad (4.10)$$

т.е. обращение кронекеровского произведения матриц может быть получено путем кронекеровского произведения обратных матриц. В этом случае соотношения для формирование базиса (4.8) могут быть представлены в следующем виде:

$$\left.\begin{array}{ll} D_0^{-1} = W_0, & D_{t+1}^{-1} = W_t \otimes D_t^{-1}; \\ D_0 = W_0^{-1}, & D_{t+1} = W_t^{-1} \otimes D_t, \end{array}\right\} \quad t = \overline{0, n-2}. \qquad (4.11)$$

Булева мультипликативная форма. Представление логических функций в булевом мультипликативном базисе основано на использовании в качестве логических операций булевой конъюнкции &, совпадающей с операцией арифметического умножения: $X\&Y = X \cdot Y$ при

X, $Y \in B_2$. Полином (4.6) в этом случае будет иметь вид:

$$p(X)=\sum_{i=0}^{k^n-1} a_i \left(X_{n-1}^{i_{n-1}} \& X_{n-2}^{i_{n-2}} \& \ldots \& X_0^{i_0} \right). \qquad (4.12)$$

Для согласования области определения булевой конъюнкции с областями значений степенных операций, последние необходимо задавать в виде булевых матриц, т.е. матриц, элементы которых принимают значения в B_2: $w(i,j) \in B_2$. При таком определении операций построение базиса для (4.12) может быть выполнено в соответствии с (4.11).

Пример 4.2. Представим функцию $F=[0\ 1\ 0\ 2\ 2\ 2\ 0\ 1\ 0]^T$ трехзначной логики от двух переменных в виде полинома в булевом мультипликативном базисе (4.12):

$$p(X)=\sum_{i=0}^{3^3-1} a_i \left(X_1^{i_1} \& X_0^{i_0} \right),$$

для чего степенные операции зададим в виде булевых матриц:

$$W_0=\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \qquad W_1=\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Для формирования матрицы прямого ортогонального преобразования D найдем матрицы, обратные W_0 и W_1:

$$W_0^{-1}=\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \qquad W_1^{-1}=\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix},$$

в соответствии с (4.11) получим:

$$D=\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & -1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \end{array}\right].$$

Вычислим вектор коэффициентов $A=D \times F=[0\ 1\ 0\ 2\ -1\ 0\ 0\ 0\ 0]^T$ и учитывая, что $X^0=1$ получим в итоге аналитическое выражение для искомой функции:

$$p(X)=X_0^1+2X_1^1-X_1^1\&X_0^1.$$

Заметим, что определения степенных операций для переменных X_0 и X_1 различаются и X_i^1 никогда не равно X_i $(\forall k>2)$.

Использование булевой мультипликативной формы позволяет эффективно реализовать вычисления в k–значной логике на вычислительных средствах, использующих двоичное кодирование данных. При реализации (4.12) операция арифметического умножения не выполняется, а вычисление сводится к суммированию коэффициентов, для которых логическая часть не равна нулю. Время вычисления логической части $\tilde{\tau}(n)$ и время вычисления всей аналитической конструкции τ_p в булевой мультипликативной форме имеют следующие значения:

$$\tilde{\tau}(n)=2n-1, \qquad \tau_p=2ns,$$

где s – вычислительная сложность полинома. Если определить степенные операции таким образом, чтобы переменные в степени 0 исключались из аналитического выражения (4.12), то полученные оценки могут быть уменьшены.

Минимизация булевой мультипликативной формы возможна путем подбора степенных операций для каждой из переменных. Количество обращаемых булевых матриц $N_B(k)$ размерности $k\times k$ может быть подсчитано по формуле

$$N_B(k)=k!\left(2\beta(k)-1\right)=k!\left(2^{\frac{k(k-1)}{2}+1}-1\right), \qquad (4.13)$$

где $k!$ – факториал числа k, $\beta(k)$ – количество базовых булевых матриц. Все базовые матрицы обращаемы и различимы относительно операций транспонирования и перестановки строк (столбцов).

Вывод (4.13) может быть осуществлен следующим образом. В качестве исходной матрицы выберем единичную матрицу E_k размерности $k\times k$, все строки которой по определению линейно независимы. От матрицы E_k породим базовые матрицы B_k путем линейной комбинации

строк (столбцов) по следующей процедуре: к первой строке можно прибавить от одной до (k–1) строк матрицы E_k, несовпадающих с первой; ко второй строке – от одной до (k–2) строк с номерами, большими двух, и т.д., к предпоследней строке можно прибавить только одну последнюю строку, что в итоге дает

$$\beta(k)=2^{((k-1)+(k-2)+\ldots+1)}=2^{\frac{k(k-1)}{2}} \qquad (4.14)$$

различных базовых матриц. Понятно, что определитель базовых матриц отличен от нуля. Каждую базовую матрицу можно транспонировать, что увеличивает количество обращаемых матриц вдвое. Для учета, того что транспонирование матрицы E_k дает ту же матрицу, уменьшим количество матриц после транспонирования на единицу. И, наконец, от каждой полученной матрицы можно породить k! новых матриц путем перестановки строк или столбцов. В итоге имеем количество обращаемых булевых матриц, задаваемое формулой (4.13).

Процедура, описанная при выводе формулы (4.13) может быть использована для порождения булевых матриц по их номеру, что необходимо при минимизации булевых мультипликативных форм. Вычислительный эксперимент, проведенный по минимизации логических функций путем подбора степенных операций показал, что вычислительная сложность s произвольной логической функции от n переменных в булевой мультипликативной форме (4.12) удовлетворяет оценке Шеннона–Лупанова: $s\sim k^n/n$ (см. § 2).

Мультипликативная форма Уолша. Определим в мультипликативном базисе степенные операции в виде матриц, состоящих из элементов множества {1, –1}. В результате получим форму, заданную относительно операции арифметического умножения и являющуюся своеобразным расширением полиномиальной формы Уолша (см. § 3) в алгебре k–значной логики:

$$p(X)=\sum_{i=0}^{k^n-1} a_i \cdot X_{n-1}^{i_{n-1}} \cdot X_{n-2}^{i_{n-2}} \cdot \ldots \cdot X_0^{i_0}. \qquad (4.15)$$

Вычисление (4.15) сводится к подсчету $b_i(X)$ – количества отрицательных значений степенных операций для каждого числа i на заданном наборе переменных X и суммированию коэффициентов со знаками, равными знакам выражений $(-1)^{b_i(x)}$.

Минимизация мультипликативной формы Уолша возможна путем подбора степенных операций для каждой из переменных. Количество обращаемых матриц Уолша $N_W(k)$ размерности $k\times k$, полученные в результате вычислительного эксперимента, представлено в табл. 4.1. Там же для сравнения приведены значения для $N_B(k)$.

Таблица 4.1

k	2	3	4
$N_W(k)$	8	192	22 272
$N_B(k)$	6	90	3 048

Из табл. 4.1 видно, что возможности минимизации логической функции в мультипликативной форме Уолша превосходят аналогичные возможности в булевой форме при $k<5$.

Для уменьшения времени вычисления логической части (4.15) определим степенные операции по аналогии с полиномиальной формой Уолша в булевой алгебре

$$X_i^j=(-1)^{(x_i \cdot j)_{\mathrm{mod}\ k}},$$

откуда находим, что форма (4.15) существует только при $k=2$ и $k=3$, что вызвано линейной зависимостью строк матриц ядра для всех $k>3$. При $k=2$ получаем классический базис Уолша. Матрицы ядра дискретного ортогонального преобразования при $k=3$ имеют вид:

$$W=\begin{bmatrix}1 & 1 & 1\\ 1 & -1 & 1\\ 1 & 1 & -1\end{bmatrix}, \qquad W^{-1}=\frac{1}{2}\begin{bmatrix}0 & 1 & 1\\ 1 & -1 & 0\\ 1 & 0 & -1\end{bmatrix},$$

а соответствующие ортогональные функции приведены на рис. 4.1.

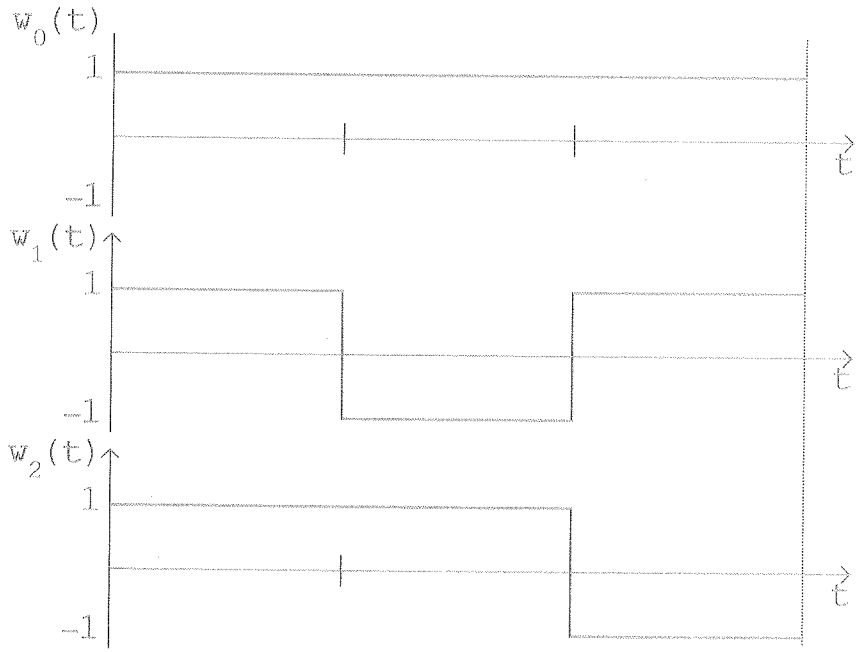


Рис. 4.1. Ортогональные функции Уолша в трехзначной логике

Задавая степенные операции как

$$X_i^j=(-1)^{(x_i+j)_{\mathrm{mod}\ k}},$$

где k – простое число, получаем аналитическую конструкцию, существующую при всех k и в которой мультипликативная операция заменяется на операцию сложения:

$$p(X)=\sum_{i=0}^{k^n-1} a_i(-1)^{\theta(X)}, \qquad \theta(X)=\sum_{j=0}^{n-1}(X_j+i_j)_{\mathrm{mod}\ k}. \tag{4.17}$$

К достоинствам формы (4.17) можно отнести регулярность структуры аналитической конструкции и малое время вычисления, которое незначительно превосходит время вычисления булевой мультипликативной формы:

$$\tilde{\tau}(n)=2n, \qquad \tau_p=s(1+2n).$$

Матрицы ядра дискретного ортогонального преобразования размерности k×k для (4.17) имеют вид:

$$W=\begin{bmatrix} 1 & -1 & 1 & -1 & \cdot & 1 \\ -1 & 1 & -1 & \cdot & 1 & \cdot \\ 1 & -1 & \cdot & 1 & \cdot & 1 \\ -1 & \cdot & 1 & \cdot & 1 & -1 \\ \cdot & 1 & \cdot & 1 & -1 & 1 \\ 1 & \cdot & 1 & -1 & 1 & -1 \end{bmatrix}, \qquad W^{-1}=\frac{1}{2^{k-2}}\begin{bmatrix} 1 & 0 & 0 & \cdot & 0 & 1 \\ 0 & 0 & \cdot & 0 & 1 & 1 \\ 0 & \cdot & 0 & 1 & 1 & 0 \\ \cdot & 0 & 1 & 1 & 0 & \cdot \\ 0 & 1 & 1 & 0 & \cdot & 0 \\ 1 & 1 & 0 & \cdot & 0 & 0 \end{bmatrix},$$

а система ортогональных функций при k=5 представлена на рис. 4.2.

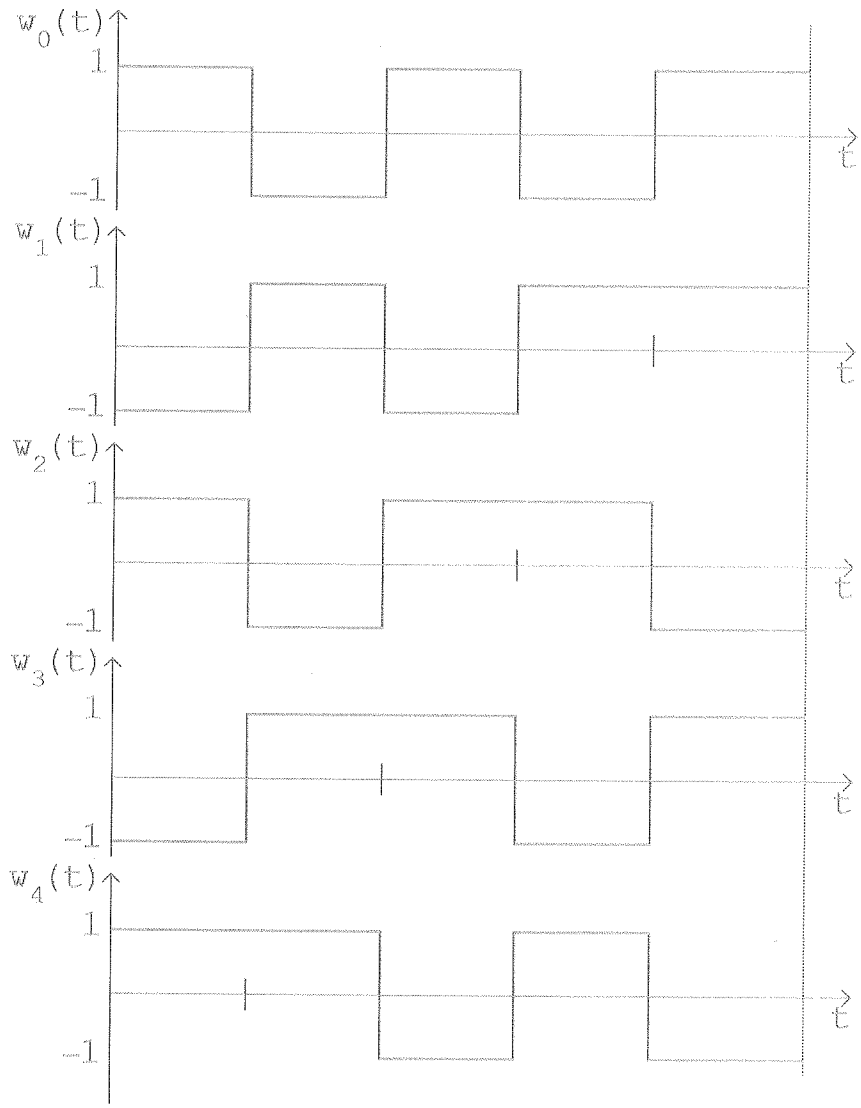


Рис. 4.2. Ортогональные функции Уолша (для k=5)

Обобщенная мультипликативная форма. Использование значений степенных операций из множества {-1, 0, 1} позволяет обобщить формы представления логических функций в мультипликативных базисах. Разложение произвольной функции в этом случае производится по трехуровневым ортогональным функциям.

Количество степенных операций для обобщенной мультипликативной формы значительно превосходит количество степенных операций для мультипликативной формы Уолша и булевой мультипликативной форму. Понятно, что матрицами степенных операций для обобщенной

мультипликативной формы могут быть все $N_B(k)$ булевы матрицы и все $N_W(k)$ матрицы Уолша, а также все обращаемые матрицы, получаемые заменой произвольного числа единиц в матрицах Уолша на ноль. Это определяет широкие возможностями минимизации в классе обобщенных мультипликативных форм.

§ 19. Формы со смешанной значностью переменных

При решении практических задач некоторые из переменных, на которых задается логическая функция, могут иметь диапазон значений, отличный от диапазона значений переменной k-значной логики, или при логическом вычислении принимать не все возможные значения по условию решаемой задачи. Функция с введенными ограничениями называется неоднородной и может рассматриваться как не полностью заданная.

Предположим, что переменные X_j принимают значения на множестве $B_{k_j}=\{0, \ldots, k_j-1\}$. При смешанной (различной) значности переменных полином (4.6) представим в следующем виде:

$$p(X)=\sum_{i=0}^{(k_{n-1}\cdot\ldots\cdot k_0)-1} a_i\left(c_i \delta_n X_{n-1}^{i_{n-1}} \delta_{n-1} \ldots \delta_0 X_0^{i_0}\right), \qquad (4.16)$$

где переменные X_j и соответствующие показатели степени i_j принимают значения из множества B_{k_j}.

Использование формы (4.16) позволяет сократить как время логических вычислений, так и объем памяти, необходимый для представления коэффициентов формы. При этом также упрощается и процедура формирования базиса. При синтезе полиномиальной формы (4.16) ядро дискретного ортогонального преобразования W_j строится с учетом значности переменных X_j в виде матарицц размерностью $k_j \times k_j$, а логические операции δ_j выбираются так, чтобы они были определены при различной значности операндов и задаются в виде матриц

размерностью $k_{j+1} \times k_j$, $j=\overline{0, n-1}$.

Пример 4.3. Получим полиномиальную форму форму в дизъюнктивном базисе для не полностью заданной функции f(X):

X_1	X_0	f(X)
0	0	1
0	1	0
0	2	1
1	0	0
1	1	1
1	2	2
2	0	1
2	1	2
2	2	3
3	0	2
3	1	1
3	2	0

Из таблицы видно, что переменная X_0 принимает значения от 0 до 2, а переменная X_1 – от 0 до 3. Полиномиальная форма функции в соответствии с (4.16) имеет вид:

$$p(X)=\sum_{i=0}^{4\cdot 3-1} a_i \left(c_i \vee X_1^{i_1} \vee X_0^{i_0} \right),$$

определения степенных операций для переменных X_1 и X_0 зададим как:

$$W_0=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}, \quad W_1=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{bmatrix},$$

и в соответствии с (4.8) построим матрицу G:

$$G=\left[\begin{array}{ccc|ccc|ccc|ccc} 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&1&0&0&1&0&0&1&0 \\ 0&2&1&0&2&1&0&2&1&0&2&1 \\ \hline 0&0&0&1&1&1&0&0&0&0&0&0 \\ 0&1&0&1&1&1&0&1&0&0&1&0 \\ 0&2&1&1&2&1&0&2&1&0&2&1 \\ \hline 0&0&0&2&2&2&1&1&1&0&0&0 \\ 0&1&0&2&2&2&1&1&1&0&1&0 \\ 0&2&1&2&2&2&1&2&1&0&2&1 \\ \hline 0&0&0&0&0&0&2&2&2&1&1&1 \\ 0&1&0&0&1&0&2&2&2&1&1&1 \\ 0&2&1&0&2&1&2&2&2&1&2&1 \end{array}\right].$$

Учитывая, что первый столбец матрицы G нулевой, задаем вектор констант C, например, $C=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$, преобразующий в соответствии с (4.8) нулевой столбец матрицы G в столбец матрицы

D^{-1}, отличный от нулевого. Матрицу прямого дискретного преобразования D получаем путем обращения матрицы D^{-1}:

$$D=\left[\begin{array}{ccc|ccc|ccc|ccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
1 & 0 & -1 & -1 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 1 \\
\hline
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 \\
\hline
-1 & -1 & -1 & 1 & 2 & 1 & 0 & -1 & -1 & 0 & 0 & 1 \\
-1 & 1 & 0 & 1 & -1 & 0 & -1 & 1 & 0 & 1 & -1 & 0 \\
-1 & 0 & 1 & 2 & -1 & -1 & -1 & 0 & 1 & 0 & 1 & -1
\end{array}\right]$$

Вектор $F=\left[1\ 0\ 1\ 0\ 1\ 2\ 1\ 2\ 3\ 2\ 1\ 0\right]^T$ извлекаем из таблицы истинности функции f(X). В результате умножения матрицы D на вектор F вычисляем вектор коэффициентов $A=\left[1\ 0\ -1\ 1\ -2\ 0\ 0\ 0\ 2\ -3\ 0\ 2\right]^T$. В итоге искомый полином имеет вид:

$$p(X)=1 \vee X_1^0 \vee X_0^0 - X_1^0 \vee X_0^2 + X_1^1 \vee X_0^0 - 2(X_1^1 \vee X_0^1) + 2(X_1^2 \vee X_0^2) - 3(X_1^3 \vee X_0^0) + 2(X_1^3 \vee X_0^2),$$

а учитывая, что $X_0^0=X_1^0=0$ и $X_0^1=X_0$ окончательно получим:

$$p(X)=1-X_0^2+X_1^2-2(X_1^2 \vee X_0)+2(X_1^2 \vee X_0^2)-3X_1^3+2(X_1^3 \vee X_0^2).$$

При синтезе и вычислении полиномиальных форм со смешанной значностью переменных необходимо учитывать, что разложение произвольного числа i на k_j-ичные цифры выполняется в системе счисления с различными основаниями для каждой из цифр: k_{n-1}, k_{n-2}, ..., k_0.

§ 20. Полиномиальные формы систем функций

Одним из достоинств полиномиальных форм является возможность обобщения их на систему логических функций. При этом сохраняется не только структура аналитической конструкции, но и способ ее синтеза, что имеет принциальное значение на практике.

Действительно, для произвольного базиса дискретного преобразования, при построении которого в соответствии с (4.4) и

(4.6) использованы операции сложения и умножения, вычисление вектора коэффициентов формы A осуществляется путем умножения матрицы D на вектор исходных данных F. Если представить значения системы в виде ортогонального объединения функций $f_j(X)$ по значностям k_j,

$$\left.\begin{aligned} F(X) &= \sum_{j=0}^{q-1} f_j(X) \prod_{p=0}^{j-1} k_p, \\ f_j(X) &\in B_{k_j},\ B_{k_j}=\{0, \ldots, k_j-1\}, \\ X &\in \{0, 1, \ldots, (k_{n-1}\cdot\ldots\cdot k_0)-1\}, \end{aligned}\right\} \qquad (4.17)$$

то вследствие дистрибутивности сложения и умножения матрицы на число и дистрибутивности сложения и умножения сцепленных матриц получим полиномиальную форму P(X) для системы функций при смешанной значности (при различной значности как самих функций, так и переменных):

$$\left.\begin{aligned} P(X) &= \sum_{i=0}^{(k_{n-1}\cdot\ldots\cdot k_0)-1} A_i \left(C_i \delta_n X_{n-1}^{i_{n-1}} \delta_{n-1} \ldots \delta_0 X_0^{i_0} \right), \\ A_i &= \sum_{j=0}^{q-1} a_{ij} \prod_{p=0}^{j-1} k_p, \end{aligned}\right\} \qquad (4.18)$$

где A_i – i-й коэффициент формы системы функций F(X), a_{ij} – i-й коэффициент формы для функции $f_j(X)$.

Пример 4.4. Представим в полиномиальной форме (4.18) систему логических функций, заданную таблицей истинности:

X_1	X_0	f_1	f_0
0	0	2	2
0	1	3	0
0	2	3	1
1	0	3	1
1	1	1	1
1	2	3	1
2	0	3	1
2	1	1	1
2	2	3	1
3	0	2	2
3	1	3	0
3	2	1	2

Из таблицы видно, что переменная X_0 имеет значность $k_0=3$, а X_1 – значность $k_1=4$, значность функции f_0 равна 3, а функции f_1 – 4, т.е. $k_{f_0}=3$, $k_{f_1}=4$. Вычислим в соответствии с (4.17) характеристический вектор системы

$$F=\begin{bmatrix}8 & 9 & 10 & 10 & 4 & 10 & 10 & 4 & 10 & 8 & 9 & 5\end{bmatrix}^T.$$

Ядро дискретного преобразования зададим в виде булевых матриц:

$$W_0=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 1 & 1\end{bmatrix}, \quad W_1=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 1\end{bmatrix},$$

а матрицы обратного D^{-1} и прямого D дискретного преобразования в дизъюнктивном базисе при векторе констант

$$C=\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

могут быть представлены следующим образом:

$$D^{-1}=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0\\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1\\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0\\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1\\
1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0\\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1\\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix},$$

$$D=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0\\
0 & 1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 1\\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
-1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0\\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1\\
-1 & 1 & 0 & 1 & -1 & 0 & -1 & 1 & 0 & 1 & -1 & 0\\
0 & -1 & 1 & 0 & 1 & -1 & 0 & -1 & 1 & 0 & 1 & -1
\end{bmatrix}.$$

Умножение матрицы D на вектор F дает вектор коэффициентов

$$A=\begin{bmatrix}8 & -6 & 1 & 0 & 7 & -5 & 0 & 0 & 0 & -5 & 0 & 5\end{bmatrix},$$

по которому строим полином для заданной системы с учетом определений степенных операций:

$$P(X)=8-6X_0^1+X_0^2+7(X_1^1 \vee X_0^1)-5(X_1^1 \vee X_0^2)-5X_1^3+5(X_1^3 \vee X_0^2).$$

Вычислим полином $P(X)$ в точке $X=(3, 2)$ и получим $P(3, 2)=5$. После преобразования, обратного (4.17), имеем значения функций в этой точке: $f_1=1$ и $f_0=2$. Заметим, что после возведения исходных переменных в соответствующие степени, последующие вычисления производятся в булевой алгебре, а значение полинома вычисляется путем суммирования коэффициентов, для которых логическая часть не равна нулю.

Полиномиальная форма (4.18) позволяет при ограниченной вычислительной сложности описать произвольное количество функций, а использование смешанной значности как функций, так и переменных упрощает формирование базиса и сокращает разрядность исходных, промежуточных и результирующих данных при логическом вычислении.

§ 21. Эффективность аналитической конструкции

Эффективность аналитической конструкции можно понимать в различных смыслах. Для представления произвольной логической функции целесообразно синтезировать такие формы, которые содержат много нулевых коэффициентов. В этом случае эффективность аналитической конструкции определяется вычислительной сложностью формы и может достигать величины k^n/n. Минимизация полиномиальных форм путем выбора оптимального или близкого к оптимальному базиса позволяет получить более низкую вычислительную сложность, но это сопряжено со значительными вычислительными затратами.

Для произвольной системы функций надежды на повышение эффективности аналитической конструкции путем оптимизации базиса не оправдываются из-за трудности минимального представления всех функций системы в одном базисе операций. Подобрать базис, в котором все функции имеют минимальное или близкое к минимальному представ-

ление, как правило, удается для очень небольшого множества функций. Путем увеличения количества операций в используемом базисе возможно расширение класса функций, имеющих эффективное представление рамках синтезируемой аналитической конструкции. Понятно, что вычислительная сложность таких форм должна быть меньше эффективной вычислительной сложности $s_{эф}$ (см. § 15), что необходимо обеспечения для абсолютной эффективности логических вычислений.

Актуальной является не только задача минимизации вычислительной сложности формы, но и задача уменьшения ее информационной емкости (энтропии). В этом случае в качестве оценки эффективности аналитической конструкции целесообразно использовать информационную энтропию. Путем уменьшения энтропии формы удается увеличить эффективную вычислительную сложность $s_{эф}$ и, тем самым, повысить эффективность логических вычислений. В итоге, эффективность аналитической конструкции можно определить двумя характеристиками: вычислительной сложностью s и информационной энтропией H_p.

Помимо минимизации энтропии и вычислительной сложности существенным при логических вычислениях является уменьшение времени вычисления. Для кратных вычислений особенно важно уменьшение времени вычисления логической части формы. Понятно, что времена вычисления одной и той же системы функций, представленных в различных формах будет различаться.

Поставим задачу оценки эффективности аналитической конструкции для представления произвольной таблицы логических данных. Для чего исследуем зависимость времени вычисления полиномиальной формы произвольной системы функций от вида аналитической конструкции. Систему функций зададим в виде характеристического вектора F длиной L_F элементы вектора F представим в соответствии с (4.17), а эффективность аналитической конструкции определим путем сравнения

времени вычислений в различных формах.

На практике получили развитие и имеют широкое распространение логические вычисления в арифметических и арифметико–логических формах в булевой алгебре [1, 14, 15, 16]. В связи с чем выберем полиномиальную форму (4.6) в булевой алгебре в качестве базовой при оценке эффективности различных аналитических конструкций.

Для синтеза формы используем дискретное ортогональное преобразование (4.4). Количество переменных n, участвующих в формировании аналитической конструкции (4.6) в булевой алгебре вычислим по формуле:

$$n=\left]\log_2 L\right[,$$

где $]x[$ – наименьшее целое, превосходящее или равное x. Количество операций (тактов) τ, необходимых для вычисления этой формы может быть представлено как

$$\tau=s+s\frac{1}{2^n}\sum_{t=0}^{n} tC_n^t=s(1+\tilde{\tau}_2), \qquad \tilde{\tau}_2=\frac{1}{2^n}\sum_{t=0}^{n} tC_n^t,$$

где s – вычислительная сложность полинома; C_n^t – число сочетаний из n элементов по t (равно количеству $\theta_i(X)$, содержащих t переменных); $\tilde{\tau}_2$ – среднее время вычисления $\theta_i(X)$. Здесь предполагается, что разрядность АЛУ достаточна для выполнения вычислений, арифметические и логические операции выполняются за один такт и при синтезе аналитической конструкции переменные в степени 0 исключены из $\theta_i(X)$, а в степени 1 совпадают со значением переменной. Ядро дискретного преобразования в этом случае задается матрицами:

$$W_j=\begin{bmatrix} e_{j_1} & 0 \\ e_{j_2} & 1 \end{bmatrix}, \qquad j=\overline{0,\ n-1},$$

где константы e_{j_1} и e_{j_2} задаются исходя из вида логической операции δ_j. Заметим, что такое построение ядра определяет минимальное время вычисления логической части синтезируемого полинома в булевой алгебре.

Смешанная значность переменных. Проведем синтез формы для той же системы логических функций, но при смешанной значности переменных. Для чего представим число L или близкое к нему число $\hat{L}$ в виде произведения произвольных $\hat{n}$ натуральных чисел:

$$L < \hat{L}=k_0^{n_0}\cdot k_1^{n_1}\cdot\ldots\cdot k_p^{n_p}=\prod_{j=0}^{p} k_j^{n_j} \qquad \hat{n}=\sum_{j=0}^{p} n_j.$$

Здесь n_j – количество переменных значности k_j; $k_0=2$, $k_j>2$ при $j>0$. Для исключения операции арифметического умножения используем дискретное преобразование в мультипликативном базисе (см. § 15). Умножение характеристического вектора функции на матрицу прямого преобразования даст искомые коэффициенты синтезируемой формы.

Количество тактов $\hat{\tau}$, необходимых для вычисления полинома (4.16), имеющего вычислительную сложность $\hat{s}$, может быть выражено формулой

$$\hat{\tau}=\hat{s}(1+\tilde{\tau}_k).$$

Как и ранее, переменные в нулевой степени исключаем из аналитической конструкции. Общее количество операций, которые необходимо выполнить для вычисления всех θ_i без учета исключения переменных будет равно

$$\hat{L}\left(k_0^{n_0}+2\sum_{t=1}^{p} k_t^{n_t}\right).$$

Учетом исключение переменных в нулевой степени из аналитической конструкции. Имеется $n_t\cdot\hat{L}/k_t$ различных θ_i, в которых отсутствует вхождение переменной со значностью k_t. С учетом этого, общее количество операций уменьшается: для булевых переменных на $n_t\cdot\hat{L}/k_t$, для переменных с иной значностью – на $2n_t\cdot\hat{L}/k_t$ (исключается одна степенная и одна логическая операция). Таким образом

$$\tilde{\tau}_k=\frac{1}{\hat{L}}\left(\hat{L}(n_0+2\sum_{t=1}^{p} n_t)-\frac{\hat{L}}{k_0}-2\sum_{t=1}^{p}\frac{\hat{L}}{k_t}\right)=n_0\frac{k_0-1}{k_0}+2\sum_{t=1}^{p} n_t\frac{k_t-1}{k_t}.$$

В итоге эффективность аналитической конструкции γ определим

выражением:

$$\gamma=\frac{\tau}{\hat{\tau}}=\frac{s(1+\tilde{\tau}_2)}{\hat{s}(1+\tilde{\tau}_k)}.$$

Полагаем, что вычислительные сложности полиномов максимальны, т.е. $s=2^n$, $\hat{s}=\hat{L}$. Тогда

$$\gamma=\frac{2^n+\sum_{t=0}^{n} tC_n^t}{\hat{L}\left(1+n_0\frac{k_0-1}{k_0}+2\sum_{t=1}^{p} n_t\frac{k_t-1}{k_t}\right)}. \qquad (4.19)$$

Результаты вычисления выражения для γ при различных L приведены в табл. 4.2, откуда видно, что для любой таблицы логических данных можно подобрать некоторую оптимальную аналитическую конструкцию.

Таблица 4.2

L	n	2^n	k_j	$\hat{L}$	γ
9	4	16	(3, 3)	9	1.454
10	4	16	(2, 5)	10	1.548
11	4	16	(2, 2, 3)	12	1.200
12	4	16	(2, 2, 3)	12	1.200
13	4	16	(2, 7)	14	1.066
14	4	16	(2, 7)	14	1.066
15	4	16	(2, 2, 2, 2)	16	1.000
16	4	16	(2, 2, 2, 2)	16	1.000
17	4	32	(2, 3, 3)	18	1.493
18	4	32	(2, 3, 3)	18	1.493
19	4	32	(2, 2, 5)	20	1.555
20	4	32	(2, 2, 5)	20	1.555
21	4	32	(3, 7)	21	1.317
22	4	32	(2, 2, 2, 3)	24	1.217
23	4	32	(2, 2, 2, 3)	24	1.217
24	4	32	(2, 2, 2, 3)	24	1.217
25	4	32	(5, 5)	25	1.066
26	4	32	(2, 13)	26	1.287
27	4	32	(2, 2, 7)	28	1.076
28	4	32	(2, 2, 7)	28	1.076
29	4	32	(2, 2, 2, 2, 2)	32	1.000
30	4	32	(2, 2, 2, 2, 2)	32	1.000
31	4	32	(2, 2, 2, 2, 2)	32	1.000
32	4	32	(2, 2, 2, 2, 2)	32	1.000
33	5	64	(3, 11)	33	1.868
34	5	64	(5, 7)	35	1.695
35	5	64	(5, 7)	35	1.695
36	5	64	(2, 2, 3, 3)	36	1.523
37	5	64	(2, 2, 2, 5)	40	1.560

Заметим, что при синтезе полиномиальной формы при смешанной значности переменных не использовались дополнительные возможности по минимизации вычислительной сложности $\hat{S}$, связанные с введением степенных операций. Следовательно, выражение (4.19) дает заниженную оценку эффективности аналитической конструкции. Минимизация вычислительной сложности и/или информационной энтропии арифметико-логической формы в расширенном базисе степенных и логических операций в пределах выбранной аналитической конструкции приводит, в конечном итоге, к большей абсолютной эффективности логических вычислений.

Смешанная значность функций. Аналогично построению формы при смешанной значности переменных может быть выполнена процедура синтеза с учетом диапазона изменения логических данных путем использования смешанной значности логических функций.

Предположим, в результате анализа таблицы установлено, что каждая из функций принимает значение в некотором диапазоне значений: $[f_{j,\min} \div f_{j,\max}]$, $j=\overline{0, q-1}$. Значность каждой функции установим равную

$$k_{f_j} = f_{j,\max} - f_{j,\min},$$

а аналитическую конструкцию систезируем в соответствии с (4.18), скорректировав характеристический вектор F_j на величину $f_{j,\min}$. После выполнения вычислений значения функций подвергнем обратной коррекции, сложив со тем же значением $f_{j,\min}$.

Информационная энтропия синтезированного полинома при смешанной значности функций уменьшается из-за уменьшения диапазона изменения коэффициентов формы и приводит к более эффективному использования разрадной сетки АЛУ. Единственным недостатком этого подхода является необходимость выполнения операции арифметического деления для извлечения значений функций из результата вычислений.

Выводы

1). Обобщенная методика синтеза полиномиальных форм позволяет получить формы логических функций с различной сложностью аналитической конструкции и для произвольных систем операций как в алгебре k-значной логики, так и в булевой алгебре.

2). Использование мультипликативных форм позволяет значительно упростить формирование базиса дискретного ортогонального преобразования и эффективно реализовать логическое вычисление в многозначной логике на традиционных вычислительных средствах, использующих двоичное кодирование данных.

3). Существуют три мультипликативные формы (булева, Уолша и обобщенная), различающиеся возможностями минимизации, причем наибольшие возможности предоставляет обобщенная мультипликативная форма.

4). Синтез полиномиальных форм при различной значности как переменных, так и логических функций позволяет упростить аналитическую конструкцию, сократить разрядность представления данных и уменьшить объем памяти при логических вычислениях.

5). Абсолютная эффективность логических вычислений в полиномиальной форме определяется тремя характеристиками: вычислительной сложностью полинома, его информационной энтропией и эффективностью аналитической конструкции.

6). Эффективность аналитической конструкции определяется временем вычисления логической части полинома и для произвольной таблицы логических данных существует оптимальная аналитическая конструкция, синтезируемая при смешанной значности переменных.

Глава 5. Реализация кратных вычислений

В настоящее время возможности дальнейшего наращивания производительности средств вычислительной техники в рамках последовательного принципа обработки данных считаются практически исчерпанными, что обусловлено в основном конечной скоростью распространения сигналов [1, с. 3]. Поиск решений проблемы повышения производительности идет в направлении развития структурных методов кратных вычислений.

Кратные вычисления возникают в виде повторного вызова подпрограмм и распараллеливания вычислительного процесса – на крупноблочном уровне, в виде конвейерного и параллельного выполнения команд – на среднеблочном уровне, в виде распараллеливания и конвейеризации операций – на мелкоблочном уровне.

Наименее изученными остаются вопросы реализации кратных вычислений в многозначной логике. Содержательная постановка задач логической обработки часто использует многозначное представление, но многозначные данные повсеместно кодируются в двоичном алфавите еще на этапе постановки задачи, а логическая обработка реализуется вычислением систем булевых функций. Многие исследователи заметили качественные отличия многозначной логики от двузначной [2]. Прямой перенос методов булевой алгебры в k-значную логику не позволяет использовать ее широкие выразительные возможности.

Рассмотрим круг вопросов, связанных с реализацией кратных вычислений на разных уровнях конвейерных и параллельных вычислительных средств, по возможности сохранив выразительные возможности многозначной логики, что достигается переходом к двоичному кодированию только на заключительном этапе реализации.

§ 22. Принципы построения программных средств

Рассмотрим программную реализацию логических вычислений на крупноблочном уровне. Основными задачами программной реализации кратных вычислений являются: представление системы логических функций в табличной форме; выбор типа аналитической конструкции с учетом функциональных возможностей вычислительного средства; синтез полиномиальной формы и ее минимизация; представление полученной минимальной формы в виде вычислительной процедуры.

Особенностью кратных вычислений в полиномиальной форме является использование АЛУ большой разрядности, определяемой как количеством вычисляемых функций, так и кратностью вычислений (см. § 15):

$$R \geq qm]\log_2 k[,$$

где R – разрядность АЛУ, q – количество логических функций, k – значность алгебры логики, m – кратность вычислений. Современные вычислительные средства используют двоичное кодирование данных ограниченной разрядности представления не превосходящей, как правило, 64. Это ограничивает количество реализуемых функций и кратность вычислений.

Решением указанной проблемы может служить использование вычислений с двойной, тройной и т.д. точностью [3]. По оценке (3.20), использование этого метода уменьшает комплексную эффективность кратных вычислений. Помимо этого возрастает трудность программирования и увеличивается сложность программы, т.к. каждая операция трансформируется в вызов подпрограммы с передачей ей в качестве операндов чисел большой разрядности. Виртуализация вычислительного средства в рамках объектно-ориентированной методологии проектирования программных средств [4] позволяет избавиться от трудностей программирования и реализовать кратные

вычисления универсальным, гибким, наглядным и эффективным способом.

Для обеспечения потребности практики оформим программные средства кратных вычислений в виде библиотеки классов. Это позволит встраивать реализацию кратных вычислений в прикладные программы традиционным способом, на этапе компоновки загружаемого модуля или в виде динамически загружаемой библиотеки. При реализации библиотеки классов используем следующие принципы, которые непосредственно вытекают из предыдущего изложения: виртуализация вычислительного средства путем использования абстрактного типа данных – целых чисел с управляемой разрядностью представления; отражение полиномиальной формы в структуру вычислительного средства; простота формирования базиса дискретного преобразования, достигаемая путем использования мультипликативных форм; минимизация вычислительной сложности и информационной энтропии формы путем подбора степенных операций; оптимизация аналитической конструкции формы путем использования смешанной значности как переменных, так и функций; инвариантность формы при изменении кратности вычисления, достигаемая путем соответствующего выбора весовых функции.

Реализацию перечисленных принципов выполним на разных этапах проектирования программного обеспечения. Вирутализацию вычислительного средства осуществим на этапе проектирования стандартного программного обеспечения; отражение полиномиальной формы в структуру вычислительного средства и ее инвариантность при изменении кратности вычислений реализуем на этапе создания бибилиотеки классов, а синтез и минимизацию осуществим на этапе проектирования прикладной программы.

§ 23. Библиотека классов для кратных вычислений

Иерархия разработанных абстрактных классов приведена на рис. 5.1. В основе иерархии лежит класс ARRAY, обеспечивающий работу с динамически выделяемыми областями оперативной памяти. Для выполнения кратных вычислений необходимо также обеспечить возможность оперирования объектами двух других типов: объектами класса INTEGER, реализующего целые числа с управляемой разрядность представления и объектами класса MATRIX – реализущего целочисленные матрицы с изменяемой размерностью. Завершает иерархию класс POLINOM, выполняющий кратные вычисления. Класс COEFICIENT является базовым для класса POLINOM и реализует однонаправленный список объектов класса INTEGER. Двойной линией на рис. 5.1 показаны связи между базовыми и производными классами, одинарной линией – связи между взаимодействующими классами, а пунктирной линией – возможные связи с производными классами прикладной программы. Заголовочные файлы для указанных классов приведены в приложении 1.

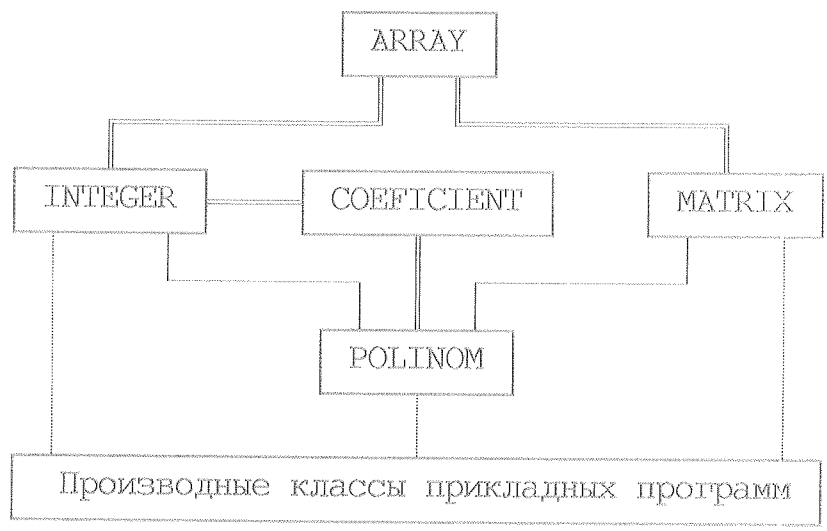


Рис. 5.1. Иерархия классов кратных логических вычислений

Проблема фрагментации памяти. Особенности объектно-ориентированной технологии программирования накладывают определен-

ные правила на последовательность вызова конструкторов и деструкторов объектов абстрактных типов [5]: при создании объекта производится вызов конструктора, а при выходе объекта из зоны видимости – вызов деструктора. При возврате функцией значения в виде объекта или ссылки на объект абстрактного типа вызывается конструктор копирования, а затем, в связи с выходом исходного объекта из области видимости – соответствующий деструктор. Деструктор освобождает выделенную под объект область динамической памяти и, тем самым, делает возможным потерю значений передаваемых данных.

Стандартным решением указанной проблемы является выделение конструктором копирования новой динамической области и пересылка в нее возвращаемых данных. Это приводит к неоправданному выделению и освобождению памяти, следствием чего является фрагментация памяти, которая снижает эффективность реализации и требует наличия развитых средств "сборки мусора" [6].

Класс ARRAY. Для более эффективного решения проблемы фрагментации памяти при передаче данных большого объема разработан класс ARRAY. Он предназначен для гибкого выделения и освобождения динамической памяти при возврате функцией объектов абстрактных типов. Динамическая область памяти выделяется один раз и при возврате объекта абстрактного класса не переназначается. Для контроля за последовательностью вызовов конструкторов и деструкторов область снабжается дескриптором, содержащим счетчик копирования: каждый конструктор копирования увеличивает счетчик на 1, каждый деструктор – уменьшает на 1, а при равенстве счетчика нулю – деструктор освобождает выделенную область динамической памяти.

Класс INTEGER реализует абстрактный тип данных – целое число с управляемой разрядностью представления. Синтаксис и семантика перегружаемых операций полностью совпадает с синтаксисом и семантикой использования встроенного типа данных – целого числа. Это поз-

воляет создавать легко читаемый программный текст, уменьшить количество ошибок при программировании приложений, улучшить возможность модификации программ.

Класс MATRIX реализует абстрактный тип данных – целочисленную матрицу с изменяемой размерностью. Синтаксис и семантика использования методов класса MATRIX совпадает с традиционным определением матриц в математике. Для синтеза аналитической конструкции полиномиальных форм введен дополнительный метод – обобщенное кронекеровское произведение матриц. Для ускорения матричных операций элементами матриц являются данные встроенного целого типа.

Класс POLINOM реализует арифметико-логический полином в булевом мультипликативном базисе, что связано с двоичным кодированием данных в ЭВМ. Для кратных вычислений с переменной (изменяемой) кратностью выбор весовой функции осуществляется таким образом, что коэффициенты формы не зависят от кратности вычислений.

Конструктор класса POLINOM по задаваемой при его вызове матрице коэффициентов каждой из функций и матрицам определений степенных операций вычисляет коэффициенты формы для кратных вычислений и размещает их в динамически выделяемой области памяти, куда также копируются заданные определения степенных операций.

Кратные вычисления осуществляются путем вызова метода VALUE, которому в качестве параметра передается матрица наборов переменных, а результатом вычисления является матрица значений системы логических функций на заданных наборах переменных.

Методика использования библиотеки классов. Библиотека классов для кратных вычисление является встраиваемым программным средством и может быть использована в любой прикладной программе, созданной в объектно-ориентированной среде программирования.

Для реализации кратных вычислений предварительно получают представление вычисляемой системы функций в некоторой мультиплика-

тивной форме в виде матрицы коэффициентов и матриц определений степенных операций. Размещение этих матрицы может осуществляться как в загружаемом модуле, так и во внешнем файле и подгружаться в процессе выполнения прикладной программы.

Конструктор класса POLINOM формирует в динамической памяти описание аналитической конструкции, позволяющее реализовать вычисления с переменной кратностью. В случае необходимости, объект класса полином может быть представлен и в статической памяти на этапе компиляции текста прикладной программы. Возможно использование смешанной значности как переменных, так и функций. В этом случае вместо значности алгебры логики при вызове конструктора задается вектор значностей переменных и вектор значностей функций.

Кратные логические вычисления реализуются путем вызова метода VALUE для сконструированного объекта класса POLINOM. Требуемая кратность вычислений определяется по размерности матриц исходных данных. Результат вычисления возвращается в виде матрицы, размерность которой определяется количеством функций и заданной кратности вычислений. Пример использования бибилиотеки классов для кратных вычислений приведен в приложении 2.

§ 24. Комплекс программ для логических исследований

Решение задач проектирования дискретных устройств и обработки логических данных основывается на результатах логических исследований. Здесь под логическим исследованием будем понимать поиск форм представления логических данных и эффективных процедур логических вычислений.

При проведении логических исследований наметились два подхода. Первый подход назовем символьным. Он основан на тождественных преобразованиях формульного представления логической функции, уже

заданной в некотором базисе операций [7]. Как правило, исходной формой является представление булевой функции в СДНФ или СКНФ (см. § 3). Последующие тождественные преобразования формульного представления преследуют две цели: минимизация формы в исходном базисе операций или переход в другой базис и минимизация формы в новом базисе. Сложность оперирования выражениями в алгебре многозначной логики приводит к необходимости кодирования значений логических данных в двоичной системе счисления и выполнения тождественных преобразований в булевой алгебре. Несмотря на кажущуюся простоту метода, его реализация сопряжена со значительными вычислительными трудностями. В более общей постановке этот метод эквивалентен решению задачи декомпозиции логической функции (см. § 2).

Второй подход назовем матричным. Он основан на использовании дискретного ортогонального преобразования и эквивалентен поиску оптимального базиса для характеристического вектора функции [8]. Реализация этого подхода также сопряжена со значительными вычислительными затратами, но регулярность аналитической конструкции обеспечивает простоту реализации метода.

Разработанная библиотека классов послужила основой для создания комплекса программ для логических исследований, где реализован второй подход. Комплекс программ состоит из следующих программных средств: LOGIC – вычисление характеристических векторов (см. § 25); BACK – обращение матриц; CRON – обобщенное кронекеровское произведение матриц; DET – вычисление определителей матриц; MUL – произведение матриц; RANDOM – генерация псевдослучайных матриц; GEN – генерация определений степенных операций; MEMORY – подсчет информационной емкости матрицы; SPECTR1 – минимизация информационнной энтропии полинома; SPECTR2 – минимизация вычислительной сложности полинома.

Входными и выходными данными для всех перечисленных программ

являются целочисленные матрицы, представленные в формате текстовых файлов. Использование чисел с фиксированной запятой позволило значительно повысить скорость выполнения матричных операций по сравнению с наиболее распространенной реализацией матричной алгебры для чисел с плавающей запятой.

Реализация программ BACK, CRON, DET, MUL, RANDOM и MEMORY не вызывает каких-либо трудностей. Все требуемые для этого функции реализованы в библиотеке классов. Программа GEN порождает обращаемые булевые матрицы по их номеру. Алгоритм построения таких матриц и перечисляющая процедура описаны при рассмотрении булевой мультипликативной формы в § 18. Здесь под перечисляющей процедурой понимается реализация алгоритма, устанавливающего взаимно однозначное соответствие между булевой матрицей и последовательностью натуральных чисел.

Программы минимизации SPECTR1 и SPECTR2 различаются только определениями целевой функции и реализуют, фактически, полный перебор возможных вариантов. Вычислительный эксперимент показал, что на современных персональных ЭВМ за приемлимое время могут быть минимизированы полиномиальные формы, содержащих до 4 переменных в степенях до 5 включительно. Для подавляющего числа применений такое отраничение не вызывает особых осложнений.

Методика логических исследований. Методика использования комплекса программ для логических исследований основана на обобщенной методике синтеза полиномиальных форм посредстеом дискретного ортогонального преобразования (см. главу 4).

Система логических функций, подлежащая исследованию, представляется в виде таблицы истинности. В случае необходимости, характеристические вектора могут быть стенерированы "случайным" образом – с помощью программного средства RANDOM.

Синтез аналитической конструкции формы возможен двумя

способами: путем использования программного средства SPECTR1 (SPECTR2), выполняющего минимизацию вычислительной сложности (информационной энтропии) мультипликативной формы или путем непосредственного использования программных средств CRON, BACK, DET.

При втором способе необходимо предварительно определить матрицы используемых логических и степенных операций. Определения степенных операций могут быть получены также с помощью программного средства GEN. Оценка эффективности формы в этом случае осуществляется программным средством MEMORY.

§ 25. Язык логического программирования

На практике логические функции помимо таблицы истинности часто задаются в виде формул в некотором функционально полном базисе операций. При матричном подходе исследуемая система должна быть представлена в виде характеристических векторов. Преобразование формульного описания в табличное осуществим путем компиляции описания системы логических функций в некоторый исполняемый или интерпретируемый код, выполнение которого для всевозможных значений независимых переменных даст искомый столбец таблицы истинности (характеристический вектор).

В приложении 3 приведена формальная грамматика языка программирования LOGIC, позволяющая задать систему логических функций в виде формул в произвольной системе унарных и бинарных операций k-значной алгебры. Реализация компилирующей и исполняющей системы выполнена в виде единого программного средства. Входными данными программы служит текстовый файл, в котором задано описание используемых унарных и бинарных операций, декларированы независимые переменные и приведены выражения для логических функций. Результатом работы программного средства является текстовый файл, содержа-

щий характеристические вектора каждой функции, заданные в виде матрицы размерности $k^n \times q$, где k – значность алгебры, n – количество независимых переменных, q – количество логических функций. Пример входных и выходных данных программы приведены в приложении 3.

Структура программного средства и организация стековой памяти представлена на рис. 5.3. Программа на языке LOGIC поступает на вход синтаксического анализатора, который проверяет правильность представления программы и, в случае ошибок, выдает сообщение о месте и типе ошибки. Так как формальная грамматика, порождающая язык LOGIC, является контекстно-свободной LL(1)-грамматикой [9], то реализация фазы синтаксического анализа выполнена в виде программно-реализованного детерминированного магазинного автомата с двумя промежуточными стеками (стек 1 и стек 2). Наличие двух стеков не является принципиальным, а необходимо только из соображений простоты реализации. В фазе синтаксического анализа строятся таблицы операций, переменных, имен и функций, а также запоминаются определения логических операций.

В фазе компиляции строится интерпретируемый машинно-независимый промежуточный код. Система команд включает команды: засылка в стек непосредственных данных, засылка в стек значения переменной, логическая операция (унарная или бинарная) и вызов функции. Номера (адреса) переменных, функций и операций задаются в командах в виде непосредственных данных.

В фазе интерпретации полученный код исполняется для каждой функции в отдельности. Свободное адресное пространство стеков используется для организации стека 3, необходимого для выполнения вычислений. Элементы характеристических векторов получаются путем занесения в ячейки памяти значений переменных и вызова вычисляемой функции. Описанная последовательность повторяется для всех функций при всех значениях переменных.

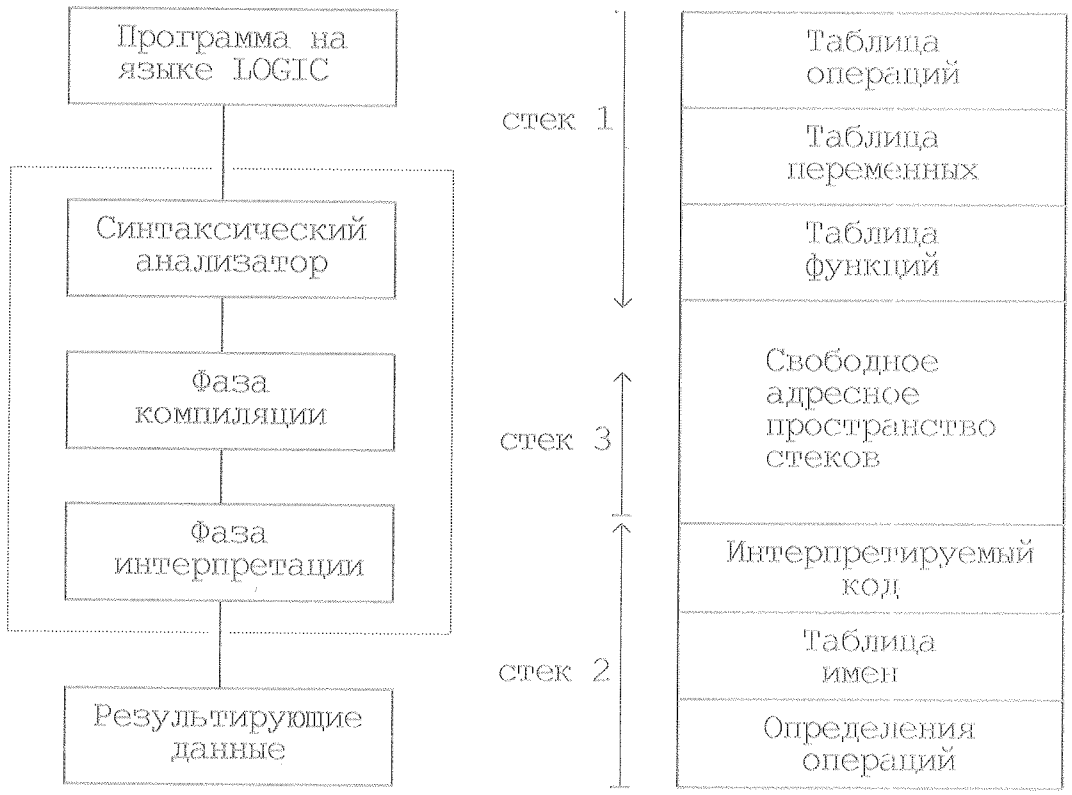


Рис. 5.3. Структура программного средства LOGIC и организация стековой памяти

§ 26. Аппаратная реализация кратных вычислений

Известна аппаратная реализация арифметических полиномов в булевой алгебре в виде комбинационной схемы [10], где система булевых функций представляется в виде композиции двух линейных полиномов (рис. 5.4). Блок Q_r первый сумматор и умножитель реализуют линейный полином

$$Q(X)=\sum_{j=1}^{n} X_j Q_j,$$

в результате вычисления которого из отдельных разрядов извлекаются требуемые конъюнкции входных переменных $\theta_i(X)$, $i=\overline{1,s}$. Блок A_r второй сумматор и умножитель реализуют линейный полином

$$P(X)=P(\theta_1(X), \ldots, \theta_s(X))=A_0+\sum_{i=1}^{s} A_i\theta_i(X).$$

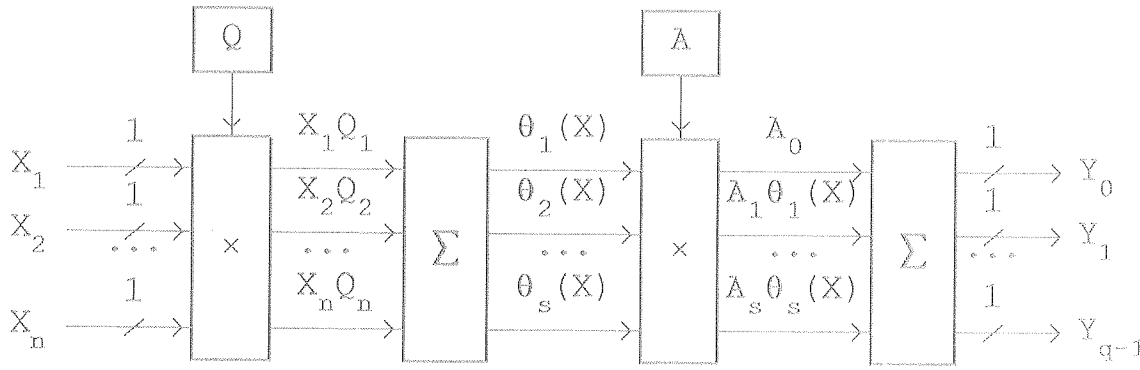


Рис. 5.4. Вычисление системы булевых функций

Описанная декомпозиция произвольной системы булевых функций на два линейных полинома может служить основой для реализации последовательных в пространстве кратных вычислений. Для этого введем в состав устройства на рис. 5.4. буферный регистр после первого сумматора и получаем конвейерное устройство. Количество ступеней конвейера равно 2_r что обеспечивает комплексную эффективность ϑ_r равную 2 при достаточно длинной последовательности данных (см. § 5).

В [11] логические вычисления реализованы в виде логического процессорного элемента при последовательном поступлении переменных и параллельном вычислении конъюнкций и при параллельном поступлении переменных и последовательном вычислении конъюнкций. Модификация этих процедур для кратных логических вычислений в алгебре Φ_m рассмотрена в § 11. Недостатком перечисленных реализаций является неэффективность обработки многозначных данных.

На рис. 5.5 приведена структура логического процессорного элемента, реализующего логическое вычисление при смешанной значности переменных и функций при последовательном вычислении полинома. Структура включает накапливающий сумматор, состоящий из комбинационного сумматора $\sum$ и регистра RG_r блок номеров коэффициентов I_r блок определений степенных операций P_r блок коэффициентов A_r блок вычисления степенней переменных X^I_r мультипликатор MUL и мультиплексор MUX. Разрядности данных имеют следующие значения:

$$r_j=\lceil \log_2 k_j \rceil, \quad r=\sum_{j=0}^{n-1} r_j=\sum_{j=0}^{n-1} \lceil \log_2 k_j \rceil,$$

$$q_p=\lceil \log_2 k_{f_p} \rceil, \quad q=\sum_{p=0}^{q-1} q_p=\sum_{p=0}^{q-1} \lceil \log_2 k_{f_p} \rceil,$$

где r_j (k_j) – разрядность (значность) j-ой переменной, q_p (k_{f_p}) – разрядность (значность) p-ой функции, n – количество переменных, q – количество функций, r (g) – разрядность входных (выходных) данных.

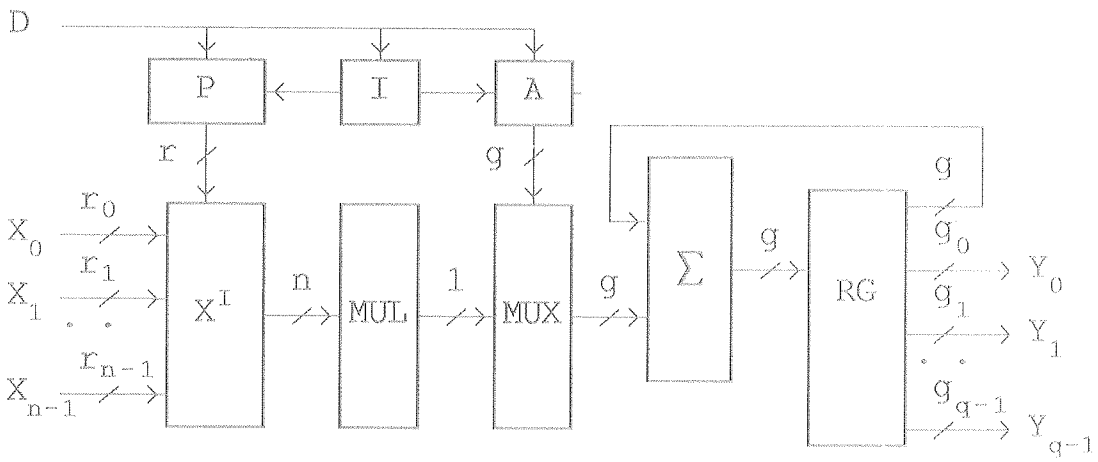


Рис. 5.5. Последовательное вычисление полинома

На каждом такте блок I выдает номер ненулевого коэффициента мультипликативной формы. Этот номер используется блокам P для возведения входных переменных в соответствующие степени, а блоком A – для выборки требуемого (очередного) коэффициента. На выход блока MUX подается коэффициент из блока A только при неравенстве нулю результата вычисления в блоке MUL, на вход которого подаются значения степеней входных переменных. Блоки памяти P, I и A организованы в виде очереди, что позволяет по шине данных D параллельно с обработкой загружать очередные данные. Пропускная способность шины H должна удовлетворять следующему условию:

$$H \geq H_p+(n+1)\sum_{j=0}^{n-1} \log_2 k_j,$$

где H_p – энтропия реализуемого полинома.

Для устройства на рис. 5.5 можно отказаться от параллельной подгрузки данных в процессе вычислений. Это позволит получить минимизированые конструкции блоков P, I и A для жестко заданной системы логических функций. Блоки P, I и A в этом случае эквивалентны блокам памяти с объемом (в битах):

$$M_P=\sum_{j=0}^{n-1}\left(\log_2 k_j\right)^2,\quad M_I=s\sum_{j=0}^{n-1}\log_2 k_j,\quad M_A=I_p,$$

где I_p – информационная емкость реализуемого полинома.

На рис. 5.6 представлена структура процессорного элемента при параллельном вычислении полинома. В отличие от предыдущей структуры, здесь отсутствует блок I, вычисление производится за один такт и подгрузка содержимого блоков P и A осуществляться заблаговременно. Расположив между блоками X^I, MUL и MUX буферные регистры получаем логический процессорный элемент с конвейерной обработкой, что реализует последовательные в пространстве кратные вычисления. Комплексная эффективность 9 для этого случая равна 4.

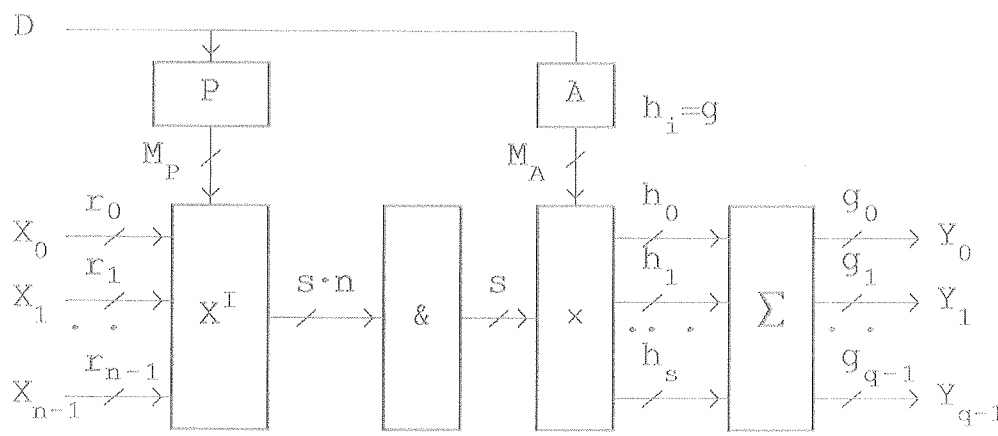


Рис. 5.6. Параллельное вычисление полинома

Для пространственного распараллеливания вычислений устройства на рис. 5.5 и рис. 5.6 дублируют. Если блоки P, I и A сделать общими для всех каналов, то распараллеливание осуществляется во времени и комплексная эффективность больше единицы (аппаратные затраты растут медленнее кратности). Для структуры на рис. 5.6

реализовано совмещение двух методов кратных вычислений: параллельного во времени и последовательного в пространстве, что обеспечивает комплексную эффективность, равную произведению эффективностей каждого из методов. Если предположить, что аппаратные затраты на блоки А и Р составляют 30% от всего оборудования, а каждый из двух реализуемых каналов разделен на четыре стадии, то получаем комплексную эффективность, равную 5,6.

Приведенные выше способы аппаратной реализации кратных вычислений относятся к мелкоблочному уровню вычислительных средств. В работе [12] предложены архитектурные принципы построения процессора с сокращенным набором команд и рассмотрены вопросы взаимодействия операционных устройств, выполняющих параллельные и последовательные вычисления в пространстве и времени на среднеблочном уровне.

В состав процессора вводится несколько операционных устройств, в том числе и требующих более одного такта для обработки данных. Занятость устройства, разделяемого между процессами или командами вызывает контекстное переключение процессора. Каждое разделяемое устройство, в том числе и участвующее в образовании мультиплексного конвейера или/и построенное по конвейерному принципу, по завершении обработки вызывает контекстное переключение на процесс, ожидающий результат. При этом у компилятора появляется возможность генерации кода параллельных вычислений как на уровне процессов, так и на уровне команд. Для увеличения быстродействия путем параллельной в пространстве обработки данных предлагается использование нескольких однотипных операционных устройств. Если при этом дублировании существуют общие блоки, то реализуются параллельные во времени кратные вычисления.

Выводы

1). Реализация логического вычисления предполагает проведение исследований по поиску форм представления логических данных. Методика проведения этих исследований основывается на обобщенном методе синтеза полиномиальных форм посредством дискретного ортогонального преобразования.

2). Язык логического программирования позволяет связать два подхода к выполнению логических исследований: символьный, основанный на тождественных преобразованиях формул и матричный, основанный на дискретном ортогональном преобразовании.

3). Принципами построения программных средств для кратных вычислений являются: виртуализация вычислительного средства, отражение полиномиальной формы в вычислительную структуру, простота формирования базиса, минимизация вычислительной сложности (энтропии), оптимизация аналитической конструкции, инвариантность формы при изменении кратности вычислений.

4). Аппаратная реализация логического вычисления на мелкоблочном и среднеблочном уровне позволяет использовать все известные методы структурной организации кратных вычислений. Возможно также совмещение двух методов в одном устройстве, что обеспечивает наивысшую итоговую комплексную эффективность.

Заключение

Настоящая диссертация является самостоятельной работой, в которой на основании выполненных автором теоретических исследований, экспериментальных работ и опытно-конструкторских разработок сформулировано и обосновано решение научной задачи, имеющей важное прикладное значение.

I. Разработаны теоретические основы кратных вычислений и, в частности, установлено:

1) основными методами структурной организации кратных вычислений является последовательные во времени, параллельные в пространстве, последовательные в пространстве и параллельные во времени кратные вычисления;

2) кратные логические вычисления являются особым случаем параллельных логических вычислений, могут быть описаны в полиномиальной форме и реализованы путем вычисления арифметико-логического полинома в базисе поразрядных операций;

3) различный выбор весовой функции, используемой при ортогональном объединении логических функций и наборов переменных в единую полиномиальную форму позволяет получать различные формы кратных вычислений, в том числе и не зависящие от кратности;

II. Исследовано применение кратных вычислений для логической обработки данных и показано:

1) возникающие на практике задачи логической обработки разделяются на 4 типа: логический синтез, логическое моделирование, логический анализ и логическое вычисление;

2) логическая обработка данных сводится к логическому синтезу и кратным логическим вычислениям, а при логическом моделировании дополнительно решается задача получения автоматной таблицы с наименьшими затратами.

III. Проанализирована эффективность реализации кратных вычислений и найдено:

1) для оценки эффективности кратных вычислений необходимо использование абсолютных и относительных критериев: эффективности по времени вычисления, эффективности по сложности вычислений и комплексной эффективности;

2) относительная комплексная эффективность больше единицы только для последовательных в пространстве и параллельных во времени кратных вычислений, а остальные методы структурной организации кратных вычислений имеют комплексную эффективность, равную единице;

3) абсолютная комплексная эффективность кратных вычислений в полиномиальной форме не зависит от кратности, а ее повышение возможно как путем увеличения разрядности АЛУ, так и при уменьшении энтропии полинома;

4) уменьшение энтропии полиномиальной формы достигается путем синтеза аналитической конструкции в расширенной базисе операций и при смешанной значности переменных и функций.

IV. Разработаны принципы реализации кратных вычислений и показано:

1) на крупноблочном уровне кратные вычисления позволяют до определенного предела наращивать интенсивность использования вычислительного средства;

2) на среднеблочном и мелкоблочном уровнях возможна совместная реализация нескольких методов структурной организации кратных вычислений, что обеспечивает наибольшую комплексную эффективность.

Литература

К введению

1. Цетлин М.Л. О непримитивных схемах // Проблемы кибернетики. – 1997. – Вып. 1. – С. 23–45.

2. Поспелов Д.А. Логические методы анализа и синтеза схем. – М.: Энергия, 1974.

3. Горбатов В.А. Основы дискретной математики. – М.: Высшая школа, 1986.

4. Захаров В.Н. Автоматы с распределенной памятью. М.: Энергия, 1975.

5. Малюгин В.Д. Надежность переключательных схем // Автоматика и телемеханики. – 1964. – № 3. – С. 1375–1383.

6. Яблонский С.В. Функциональные построения в k–значной логике // Труды Мат. ин–та АН СССР им. В.А. Стеклова. – 1958. – С. 5–142.

7. Иваськов Ю.Л. Принципы построения многозначных физических схем. – Киев: Наукова думка, 1971.

8. Лабунец В.Г., Ситников О.П. Обобщение понятия пороговой функции k–значной логики над конечным полем Галуа // Изв. АН СССР. Сер. Техн. киберн. – 1975. – № 5. – С. 137–141.

9. Закревский А.Д. Логические уравнения. – Минск: Наука и техника, 1975.

10. Пухальский Г.И. Логическое проектирование цифровых устройств радиотехнических систем. – Л.: Изд–во ЛГУ, 1976.

11. Карповский М.Г., Москалев Э.С. Спектральные методы анализа и синтеза дискретных устройств. – Л.: Энергия, 1973.

12. Миллер Р. Теория переключательных схем. – М.: Наука, 1970. – Т. 1.

13. Рвачев В.Л. Теория R-функций и некоторые ее приложения. – Киев: Наукова думка, 1982.

14. Томич Ж. Арифметическое представление логических функций // Дискретные автоматы и сети связи: Сб. науч. тр. – М.: Наука, 1970. – С. 131–136.

15. Малюгин В.Д., Кухарев Г.А., Шмерко В.П. Преобразования полиномиальных форм булевых функций. – М.: Институт проблем управления. Препринт, 1986.

16. Малюгин В.Д. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. – 1982. – № 4. – С. 84 – 93.

17. Малюгин В.Д. Реализация кортежей булевых функций посредством линейных арифметических полиномов // Автоматика и телемеханика. – 1984. – № 2. – С. 114 – 122.

18. Малюгин В.Д. Параллельные логические вычисления // Всесоюзное совещание по распределенной обработке информации: Сб. тр. – Львов, 1985.

19. Malyugin V.D., Veits A.V. Intensive calculation in parallel logic // Proceedings of 5th intern. workshop on spectral techniques. China. – Beijina, 1994. – P. 63–64.

20. Кузнецов О.П. О программной реализации логических автоматов // Автоматика и телемеханика. – 1977. – № 7. – С. 163–174; № 9. – С. 137–149.

21. Мак–Нотон Р. Теорема о бесконечнозначной логике высказываний // Кибернетический сборник. – М.: Изд–во иностр. лит., 1961. – Вып. 3. – С. 59–78.

22. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976.

23. Левин В.И. Структурно–логические методы исследования сложных систем. – М.: Наука, 1987.

К главе 1

1. Кухарев Г.А., Шмерко В.П., Янушкевич С.Н. Техника параллельной обработки бинарных данных на СБИС. – Минск: Вышэйшая школа, 1991.

2. Поспелов Д.А. Логические методы анализа и синтеза схем. – М.: Энергия, 1974.

3. Пухальский Г.И. Логическое проектирование цифровых устройств радиотехнических систем. – Л.: Изд-во ЛГУ, 1976.

4. Кудрявцев В.Б., Алешин С.В., Подколзин А.С. Введение в теорию автоматов. – М.: Наука. Гл. ред. физ.-мат. лит., 1985.

5. Закревский А.Д. Логические уравнения. – Минск: Наука и техника, 1975.

6. Захаров В.Н., Поспелов Д.А., Хазацкий В.Е. Системы управления. – М.: Энергия, 1977.

7. Коробчинский Н.Е., Трахтенброт Б.А. Введение в теорию конечных автоматов. – М.: Наука, 1962.

8. Глушков В.М. Синтез цифровых автоматов. – М.: Наука, 1962.

9. Бибило П.Н. Декомпозиция булевых функций (обзор) // В кн. Проектирование устройств логического управления. – М.: Наука, 1985. – С. 106–126.

10. Яблонский С.В. Об алгоритмических трудностях синтеза минимальных контактных схем //Проблемы кибернетики.– 1959.– Вып. 2.

11. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – М.: Энергоатомиздат, 1988.

12. Лупанов О.Б. О синтезе некоторых классов управляющих систем // Проблемы кибернетики. – 1963. – Вып. 10. – С. 63–97.

13. Кузнецов О.П. О программной реализации логических функций и автоматов // Автоматика и телемеханика. – 1977. – № 7. – С. 163–174; № 9. – С. 137–149.

14. Яблонский С.В. Введение в дискретную математику. – М.: Наука. Гл. ред. физ.-мат. лит., 1986.

15. Кузьмин В.А. Оценка сложности реализации функций алгебры логики простейшими видами бинарных программ // Дискретный анализ. – Новосибирск: Институт математики СО АН СССР, 1976. – Вып. 29. – С. 11–39.

16. Гильберт Д., Аккерман Б. Основы теоретической логики. – М.: Иностранная литература, 1947.

17. Авсаркисян Г.С., Брайловский Г.С. Представление логических функций в виде полиномов Жегалкина // Автоматика и вычислительная техника. – 1975. – № 6. – С. 6–8.

18. Авсаркисян Г.С. Обобщенные полиномиальные формы булевых функций и синтез многовыходных логических схем // Автоматика и телемеханики. – 1983. – № 11. – С. 111–119.

19. Menger K.S. A Transform for Logic Networks // IEEE Trans. on computers. – 1969. – Vol. C–18, № 3. – P. 241–250.

20. Benjauthrit B., Reed S. Galois Switching Function and their Application // IEEE Trans. on computers. – 1976. – Vol. C–25, № 1. – P. 78–86.

21. Александров И.Н., Котов В.М., Никитюк Н.М. Применение переключательных функций в поле Галуа $GF(2^m)$ для синтеза универсальных динамически программируемых модулей // Автоматика и телемеханика. – 1995. – № 5. – С. 137–148.

22. Малюгин В.Д. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. – 1982. – № 4. – С. 84–93.

23. Антоненко В.М., Иванов А.А., Шмерко В.П. Линейные арифметические формы k–значных логик и их реализация на систолических массивах // Автоматика и телемеханика. – 1995. – № 4. – С. 139–155.

24. Карповский М.Г., Москалев Э.С. Реализация системы логических функций при помощи разложения в ортогональные ряды // Автоматика и телемеханика. – 1967. – № 12.

25. Карповский М.Г., Москалев Э.С. Использование автокорреляционных характеристик для реализации систем функций алгебры логики // Автоматика и телемеханика. – 1970. – № 2.

26. Садыхов Р.Х., Чеголин П.М., Шмерко В.П. Методы и средства обработки сигналов в дискретных базисах. – Минск: Наука и техника, 1987.

27. Карповский М.Г., Москалев Э.С. Спектральные методы анализа и синтеза дискретных устройств. – Л.: Энергия, 1973.

28. Залмазон Л.А. Преобразование Фурье, Уолша, Хаара и их применение в управлении, связи и других областях. – М.: Наука, 1989.

29. Кухарев Г.А., Шмерко В.П., Янушкевич С.Н. Новые возможности дискретного преобразования Фурье для аналитического описания бинарных и многозначных данных // Распознавание, классификация, прогноз. Математические методы и их применение / ВЦ АН СССР. – М.: Наука, 1991. – Вып. 3. – С. 112–147.

30. Кочкарев Ю.А. Ортогональные сигналы в вычислительной технике. – Ростов: Из–во Ростовского университета, 1980.

31. Крот А.М. Дискретные модели динамических систем на основе полиномиальной алгебры. – Минск: Наука и техника, 1990.

32. Малюгин В.Д. Параллельные логические вычисления // Всесоюзное совещание по распределенной обработке информации: Сб. тр. – Львов, 1985.

33. Артюхов В.Л., Кондратьев В.Н., Шалыто А.А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. – 1988. – № 4. – С.138 – 147.

34. Малюгин В.Д. Реализация кортежей булевых функций посредством линейных арифметических полиномов // Автоматика и телемеханика. – 1984. – № 2. – С. 114–122.

35. Шмерко В.П. Синтез арифметических форм булевых функций посредством преобразования Фурье // Автоматика и телемеханика. – 1989. – № 5. – С. 134–142.

36. Malyugin V.D., Veits A.V. Intensive calculation in parallel logic // Proceedings of 5th intern. workshop on spectral techniques. China. – Beijina, 1994. – P. 63–64.

37. Дагман Э.Е., Кухарев Г.А. Быстрые дискретные ортогональные преобразования. – Новосибирск: Наука. Сиб. отделение, 1983.

38. Малюгин В.Д. Реализация систем логических функций посредством арифметических полиномов // Computer and Artificial Intellegence. – Bratislava, 1987. – № 6. – P. 541–552.

39. Пухальский Г.И., Новосельцева Т.Я. Проектирование дискретных устройств на интегральных микросхемах: Справочник. – М.: Радио и связь, 1990.

40. Амамия М., Танака Ю. Архитектура ЭВМ и искусственный интеллект. – М.: Мир, 1993.

41. Водяхо А.И., Горнец Н.Н., Пузанков Д.В. Высокопроизводительные системы обработки данных: Учеб. пособие для вузов. – М.: Высш. шк., 1997.

42. Burcs A.W., Goldstine H.H., von Neumann J. Preliminary Discussion of the Logig Design of an Electronic Computing Instrument. – Prinseton: Institute for Advanced Study, 1946. – Pt. 1, vol. 1.

43. Кухарчук А.Г., Луцкий Г.М. Конвейерный принцип обработки информации // Кибернетика. – 1968. – № 6. – С. 43–49.

44. Самофалов К.Г., Луцкий Г.М. Основы теории многоуровневых конвейерных вычислительных систем. – М.: Радио и связь, 1989.

45. Головкин Б.А. Параллельные вычислительные системы. – М.: Наука, 1980.

46. Системы параллельной обработки / Под ред. Д.Ивенса. – М.: Мир, 1985.

47. Лапкин Л.Я. О векторной программной реализации логических функций // Автоматика и телемеханика. – 1983. – № 3.

48. Степаненко С.А. Параллельное вычисление булевых функций // Абстрактная и структурная теория релейных устройств. – М.: Наука, 1982. – С. 64–76.

К главе 2

1. Выхованец В.С. Многократные параллельные логические вычисления // Информационно–управляющие системы и специализированные вычислительные устройства для обработки и передачи данных: Материалы Всероссийской конференции. – Махачкала, 1996. – С. 105–106.

2. Выхованец В.С. Многократные параллельные логические вычисления // Вестник Приднестровского университета. – Тирасполь, 1997. – № 2. – С. 64–74.

3. Малюгин В.Д., Выхованец В.С. Кратные логические вычисления // Автоматика и телемеханика. – 1998. – № 4.

4. Яблонский С.В. Введение в дискретную математику. – М.: Наука. Гл. ред. физ.–мат. лит., 1986.

5. Малюгин В.Д. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. – 1982. – № 4. – С. 84–93.

6. Малюгин В.Д. Реализация кортежей булевых функций посредством линейных арифметических полиномов // Автоматика и телемеханика. – 1984. – № 2. – С. 114–122.

7. Малюгин В.Д., Кухарев Г.А., Шмерко В.П. Преобразования полиномиальных форм булевых функций. – М.: Институт проблем управления. Препринт, 1986.

8. Горяшко А.П., Немировский А.С. Оценки информационной стоимости вычисления булевых функций в комбинационных схемах // Проблемы передачи информации. – 1978. – Т. XIV, вып. 1. – С. 90–100.

9. Малюгин В.Д., Соколов В.В. Интенсивные логические вычисления // Автоматика и телемеханика. – 1993. – № 4. – С. 160–167.

10. Малюгин В.Д. Реализация систем логических функций посредством арифметических полиномов // Computer and Artificial Intellegence. – Bratislava, 1987. – № 6. – P. 541–552.

11. Шалыто А.А., Кондратьев В.Н. Методы программной реализации алгоритмов логического управления для судовых микропроцессорных систем. – Л., Институт повышения квалификации. Судпром, 1990.

12. Артюхов В.Л., Кондратьев В.Н., Шалыто А.А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. – 1988. – № 4. – С.138–147.

К главе 3

1. Малюгин В.Д., Выхованец В.С. Кратные логические вычисления // Автоматика и телемеханика. – 1998. – № 4.

2. Выхованец В.С. Кратные логические вычисления и их применение в управлении, обработке информации и других областях / Приднестровский государственно-корпоративный университет им. Т.Г. Шевченко. – Тирасполь, 1997. – 18 с. – Деп. в ВИНИТИ 05.06.97, № 1851-В97.

3. Антоненко В.М., Иванов А.А., Шмерко В.П. Линейные арифметические формы k-значных логик и их реализация на систолических массивах // Автоматика и телемеханика. – 1995. – № 4. – С. 139–155.

4. Strazdins I.J The polinomial Algebra of Multivalued Logic // Algebra, Combinatorics and Computer Science. – 1983. – P. 777–785.

5. Кухарев Г.А., Шмерко В.П., Зайцева Е.Н. Алгоритмы и систолические процессоры для обработки многозначных данных. – Минск: Вышэйшая школа, 1991.

6. Малюгин В.Д. О полиномиальной реализации кортежа булевых функций // Доклады АН СССР. – 1982. – Т. 265, № 6. – С. 1338–1341.

7. Малюгин В.Д. Параллельные логические вычисления // Всесоюзное совещание по распределенной обработке информации: Сб. науч. тр. – Львов, 1985.

8. Малюгин В.Д. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. – 1982. – № 4. – С. 84–93.

9. Кухарев Г.А., Шмерко В.П., Янушкевич С.Н. Техника параллельной обработки бинарных данных на СБИС. – Минск: Вышэйшая школа, 1991.

10. Малюгин В.Д. Реализация кортежей булевых функций посредством линейных арифметических полиномов // Автоматика и телемеханика. – 1984. – № 2. – С. 114–122.

11. Кондратьев В.Н., Шалыто А.А. Реализация систем булевых функций с использованием линейных арифметических полиномов // Автоматика и телемеханика. – 1993. – № 3. – С. 135–151.

12. Малюгин В.Д., Соколов В.В. Интенсивные логические вычисления // Автоматика и телемеханика. – 1993. – № 4. – С. 160–167.

13. Выхованец В.С. Многократные параллельные логические вычисления // Информационно-управляющие системы и специализированные вычислительные устройства для обработки и передачи данных: Материалы Всероссийской конференции. – Махачкала, 1996. – С. 105–106.

14. Выхованец В.С. Многократные параллельные логические вычисления // Вестник Приднестровского университета. – 1997. – № 2. – С. 64–74.

К главе 4

1. Кухарев Г.А., Шмерко В.П., Янушкевич С.Н. Техника параллельной обработки бинарных данных на СБИС. – Минск: Вышэйшая школа, 1991.

2. СБИС для распознавания образов и обработки изображений / К. Фу, С. Кунг, Р. Пикард и др. – М.: Мир, 1988.

3. Дагман Э.Е., Кухарев Г.А. Быстрые дискретные ортогональные преобразования. – Новосибирск: Наука. Сиб. отделение, 1983.

4. Кухарев Г.А., Шмерко В.П., Янушкевич С.Н. Новые возможности дискретного преобразования Фурье для аналитического описания бинарных и многозначных данных // Распознавание, классификация, прогноз. Математические методы и их применение / ВЦ АН СССР. – М: Наука, 1991. – Вып. 3. – С. 112–147.

5. Выхованец В.С. Дискретное преобразование Фурье и его применение при логических вычислениях / Приднестровский государственно-корпоративный университет им. Т. Г. Шевченко. – Тирасполь, 1997. – 15 с. – Деп. в ВИНИТИ 05.06.97, № 1852–В97.

6. Малюгин В.Д., Выхованец В.С. Кратные логические вычисления // Автоматика и телемеханика. – 1998. – № 4.

13. Выхованец В.С. Многократные параллельные логические вычисления // Информационно-управляющие системы и специализированные вычислительные устройства для обработки и передачи данных: Материалы Всероссийской конференции. – Махачкала, 1996. – С. 105–106.

14. Выхованец В.С. Многократные параллельные логические вычисления // Вестник Приднестровского университета. – 1997. – № 2. – С. 64–74.

К главе 4

1. Кухарев Г.А., Шмерко В.П., Янушкевич С.Н. Техника параллельной обработки бинарных данных на СБИС. – Минск: Вышэйшая школа, 1991.

2. СБИС для распознавания образов и обработки изображений / К. Фу, С. Кунг, Р. Пикард и др. – М.: Мир, 1988.

3. Дагман Э.Е., Кухарев Г.А. Быстрые дискретные ортогональные преобразования. – Новосибирск: Наука. Сиб. отделение, 1983.

4. Кухарев Г.А., Шмерко В.П., Янушкевич С.Н. Новые возможности дискретного преобразования Фурье для аналитического описания бинарных и многозначных данных // Распознавание, классификация, прогноз. Математические методы и их применение / ВЦ АН СССР. – М: Наука, 1991. – Вып. 3. – С. 112–147.

5. Выхованец В.С. Дискретное преобразование Фурье и его применение при логических вычислениях / Приднестровский государственно-корпоративный университет им. Т. Г. Шевченко. – Тирасполь, 1997. – 15 с. – Деп. в ВИНИТИ 05.06.97, № 1852-В97.

6. Малюгин В.Д., Выхованец В.С. Кратные логические вычисления // Автоматика и телемеханика. – 1998. – № 4.

7. Menger K.S. A Transform for Logic Networks // IEEE Trans. on computers. – 1969. Vol. C–18, № 3. – P. 241–250.

8. Залмазон Л.А. Перобразование Фурье, Уолша, Хаара и их применение в управлении, связи и других областях. – М.: Наука, 1989.

9. Блейхут Р. Быстрые алгоритмы цифровой обработки сигналов. – М.: Мир, 1989.

10. Быстрые алгоритмы в цифровой обработке изображений / Т.С. Хуанг, Дж. О. Эклунд, Г.Дж. Нуссбаумер и др. – М.: Радио и связь, 1984.

11. Шмерко В.П. Синтез арифметических форм булевых функций посредством преобразования Фурье // Автоматика и телемеханика. – 1989. – № 5. – С. 134–142.

12. Малюгин В.Д., Кухарев Г.А., Шмерко В.П. Преобразования полиномиальных форм булевых функций. – М.: Институт проблем управления. Препринт, 1986.

13. Антоненко В.М., Иванов А.А., Шмерко В.П. Линейные арифметические формы k–значных логик и их реализация на систолических массивах // Автоматика и телемеханика. – 1995. – № 4. – С. 139–155.

14. Артюхов В.Л., Кондратьев В.Н., Шалыто А.А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. – 1988. – № 4. – С. 138–147.

15. Малюгин В.Д. Реализация кортежей булевых функций посредством линейных арифметических полиномов // Автоматика и телемеханика. – 1984. – № 2. – С. 114–122.

16. Малюгин В.Д., Соколов В.В. Интенсивные логические вычисления // Автоматика и телемеханика. – 1993. – № 4. – С. 160–167.

1. Самофалов К.Г., Луцкий Г.М. Основы теории многоуровневых конвейерных вычислительных систем. – М.: Радио и связь, 1989.

2. Гиндикин С.Г. Алгебра логики в задачах. – М.: Наука, 1972.

3. Лю Ю-Чжен, Гибсон Г. Микропроцессоры семейства 8086/8088. Архитектура, программирование, проектирование микрокомпьютерных систем. – М.: Радио и связь, 1987.

4. Буч Г. Объектно-ориентированное проектирование с примерами применения. – М.: Конкорд, 1992.

5. Цимбал А.А., Майоров А.Г., Козодаев М.А. Turbo C++: язык и его применение. – М.: Джен Ай Лтд., 1993.

6. Шоу А. Логическое проектирование операционных систем. – М.: Мир, 1981.

7. Миллер Р. Теория переключательных схем. – М.: Наука, 1970.

8. Карповский М.Г., Москалев Э.С. Спектральные методы анализа и синтеза дискретных устройств. – Л.: Энергия, 1973.

9. Рейуорд-Смит В.Дж. Теория формальных языков. – М.: Радио и связь, 1988.

10. Малюгин В.Д. Реализация систем логических функций посредством арифметических полиномов // Computer and Artificial Intellegence. – Bratislava, 1987. – № 6. – P. 541-552.

11. Малюгин В.Д., Соколов В.В. Интенсивные логические вычисления //Автоматика и телемеханика. – 1993. – № 4. – С. 160-167.

12. Выхованец В.С., Гордиенко К.П. Процессор с сокращенным набором команд // Вестник Приднестровского университета. – Тирасполь: 1994. – № 1. – С. 124-128.

1. Самофалов К.Г., Луцкий Г.М. Основы теории многоуровневых конвейерных вычислительных систем. – М.: Радио и связь, 1989.

2. Гиндикин С.Г. Алгебра логики в задачах. – М.: Наука, 1972.

3. Лю Ю-Чжен, Гибсон Г. Микропроцессоры семейства 8086/8088. Архитектура, программирование, проектирование микрокомпьютерных систем. – М.: Радио и связь, 1987.

4. Буч Г. Объектно-ориентированное проектирование с примерами применения. – М.: Конкорд, 1992.

5. Цимбал А.А., Майоров А.Г., Козодаев М.А. Turbo C++: язык и его применение. – М.: Джен Ай Лтд., 1993.

6. Шоу А. Логическое проектирование операционных систем. – М.: Мир, 1981.

7. Миллер Р. Теория переключательных схем. – М.: Наука, 1970.

8. Карповский М.Г., Москалев Э.С. Спектральные методы анализа и синтеза дискретных устройств. – Л.: Энергия, 1973.

9. Рейуорд-Смит В.Дж. Теория формальных языков. – М.: Радио и связь, 1988.

10. Малюгин В.Д. Реализация систем логических функций посредством арифметических полиномов // Computer and Artificial Intellegence. – Bratislava, 1987. – № 6. – P. 541-552.

11. Малюгин В.Д., Соколов В.В. Интенсивные логические вычисления //Автоматика и телемеханика. – 1993. – № 4. – С. 160-167.

12. Выхованец В.С., Гордиенко К.П. Процессор с сокращенным набором команд // Вестник Приднестровского университета. – Тирасполь: 1994. – № 1. – С. 124-128.

Заголовочные файлы для кратных логических вычислений

П.1. Заголовочный файл для встроенных типов данных TYPES.H

```
#if !defined( _TYPES_ )
#define _TYPES_
#include <limits.h>
//-------------------------------------------------------------------------//
typedef unsigned char  byte;
typedef unsigned int   word;
typedef unsigned long  dword;
typedef unsigned int   index;
typedef unsigned int   size;
typedef int            boolean;
//-------------------------------------------------------------------------//
#define SHORT_BIT          ((CHAR_BIT)*sizeof(short))
#define INT_BIT            ((CHAR_BIT)*sizeof(int))
#define LONG_BIT           ((CHAR_BIT)*sizeof(long))
#define BYTE_BIT           ((CHAR_BIT)*sizeof(byte))
#define WORD_BIT           ((CHAR_BIT)*sizeof(word))
#define DWORD_BIT          ((CHAR_BIT)*sizeof(dword))
#define POINTER_BIT        ((CHAR_BIT)*sizeof(void*))
#define BYTE_MAX           (UCHAR_MAX)
#define WORD_MAX           (UINT_MAX)
#define DWORD_MAX          (ULONG_MAX)
#define INDEX_MAX          (UINT_MAX)
#define SIZE_MAX           (UINT_MAX)
#define HEX_BIT             4
#define HEX_BASE            16
#define BYTE_HEX           ((BYTE_BIT)/(HEX_BIT))
#define WORD_HEX           ((WORD_BIT)/(HEX_BIT))
#define DWORD_HEX          ((DWORD_BIT)/(HEX_BIT))
//-------------------------------------------------------------------------//
#define TYPES_FRAME_TYPE  long
#define TYPES_FRAME_SIZE  (sizeof(TYPES_FRAME_TYPE))
typedef union {
   TYPES_FRAME_TYPE  t;
   long              l  [ TYPES_FRAME_SIZE/sizeof( long ) ];
```

```
    int             i  [ TYPES_FRAME_SIZE/sizeof( int  ) ];
    short           s  [ TYPES_FRAME_SIZE/sizeof( short) ];
    char            c  [ TYPES_FRAME_SIZE/sizeof( char ) ];
    unsigned long   ul [ TYPES_FRAME_SIZE/sizeof( unsigned long ) ];
    unsigned int    ui [ TYPES_FRAME_SIZE/sizeof( unsigned int ) ];
    unsigned short us [ TYPES_FRAME_SIZE/sizeof( unsigned short ) ];
    unsigned char   uc [ TYPES_FRAME_SIZE/sizeof( unsigned char ) ];
    signed long     sl [ TYPES_FRAME_SIZE/sizeof( signed long ) ];
    signed int      si [ TYPES_FRAME_SIZE/sizeof( signed int ) ];
    signed short    ss [ TYPES_FRAME_SIZE/sizeof( signed short ) ];
    signed char     sc [ TYPES_FRAME_SIZE/sizeof( signed char ) ];
    dword           d  [ TYPES_FRAME_SIZE/sizeof( dword ) ];
    word            w  [ TYPES_FRAME_SIZE/sizeof( word ) ];
    byte            b  [ TYPES_FRAME_SIZE/sizeof( byte ) ];
    void*           p  [ TYPES_FRAME_SIZE/sizeof( void* ) ];
} types_frame;
//-------------------------------------------------------------------------//
#endif
```

Ш.2. Заголовочный файл ARRAY.H

```
#if !defined( _ARRAY_ )
#define _ARRAY_
#if !defined( _TYPES_ )
#include <types.h>
#endif
//-------------------------------------------------------------------------//
typedef struct {
   int   use;                  // счетчик копирования
   size  len;                  // длина области
   char  buf[ 1 ];             // начало области
} array_struct, *memory;
//-------------------------------------------------------------------------//
class array {
private:
   memory val;                 // указатель на область
protected:
   size resize  ( size );      // изменение размера области
   size unique  ( void );      // создание уникальной области
   int  inuse   ( void ) const;// счетчик копирования
```

```
public:
   ~array( void );                        // деструктор
   array( void );                         // конструктор
   array( const array& );                 // конструктор копирования
   array& operator=( const array& );      // оператор копирования
   void*  value   ( void ) const;         // указатель на область данных
   size   length  ( void ) const;         // длина области данных
};
#endif
```

П1.3. Заголовочный файл INTEGER.H

```
#if !defined( _INTEGER_ )
#define _INTEGER_
#include <string.h>
#include <iostream.h>
#if !defined( _ARRAY_ )
#include <array.h>
#endif
//-------------------------------------------------------------------------//
#if (sizeof(dword)!=2*sizeof(word))||sizeof(word)!=sizeof(int)
#   error Error defined base types in integer.h
#endif
#define  INTEGER_SLICE_MAX    ((INDEX_MAX/sizeof( word ))*WORD_BIT)
#define  INTEGER_SCAN_START   (ULONG_MAX)
//-------------------------------------------------------------------------//
typedef word (*integer_binary) ( word, word );
typedef word (*integer_unary ) ( word );
typedef unsigned long slice;
//-------------------------------------------------------------------------//
class integer : public array {
private:
   integer& not     ( void  );
   integer& neg     ( void  );
   integer& modul   ( void  );
   integer& add     ( int, slice = 0 );
   integer& sub     ( int, slice = 0 );
   integer& add     ( const integer&, slice = 0 );
   integer& sub     ( const integer&, slice = 0 );
   integer& mul     ( const integer&, word, index );
```

```
    slice    bitsum ( slice = INTEGER_SLICE_MAX ) const;
    index    atoi   ( const char*, index );
    slice    bits   ( void ) const;
protected:
    word*    value  ( void ) const;
    index    length ( void ) const;
    int      sign   ( void ) const;
    index    resize ( index );
    index    output ( ostream& ) const;
    index    input  ( istream& );
public:
    static index  stoi( slice );
    static word   carrier;
    static slice  bit;
    ~integer( void );
    integer( void );
    integer( int  );
    integer( slice );
    integer( const integer& );
    integer( const char* );
    integer( const string& );
    integer( istream& );
    integer& operator ++ ( void );
    integer& operator -- ( void );
    integer& operator ++ ( int );
    integer& operator -- ( int );
    integer  operator -  ( void ) const;
    integer  operator ~  ( void ) const;
    boolean  operator !  ( void ) const;
    integer  operator *  ( const integer& ) const;
    integer  operator /  ( const integer& ) const;
    integer  operator %  ( const integer& ) const;
    integer  operator +  ( const integer& ) const;
    integer  operator -  ( const integer& ) const;
    integer  operator *  ( int    ) const;
    integer  operator /  ( int    ) const;
    integer  operator %  ( int    ) const;
    integer  operator +  ( int    ) const;
    integer  operator -  ( int    ) const;
    integer  operator << ( slice ) const;
```

```
integer  operator >> ( slice ) const;
boolean  operator >= ( const integer& ) const;
boolean  operator <= ( const integer& ) const;
boolean  operator >  ( const integer& ) const;
boolean  operator <  ( const integer& ) const;
boolean  operator == ( const integer& ) const;
boolean  operator != ( const integer& ) const;
integer  operator &  ( const integer& ) const;
integer  operator ^  ( const integer& ) const;
integer  operator |  ( const integer& ) const;
integer& operator =  ( int    );
integer& operator =  ( const char*    );
integer& operator =  ( const string&  );
integer& operator =  ( const integer& );
integer& operator += ( const integer& );
integer& operator -= ( const integer& );
integer& operator &= ( const integer& );
integer& operator |= ( const integer& );
integer& operator ^= ( const integer& );
integer& operator *= ( const integer& );
integer& operator /= ( const integer& );
integer& operator %= ( const integer& );
integer& operator <<=( slice );
integer& operator >>=( slice );
integer& operator += ( int    );
integer& operator -= ( int    );
integer& operator *= ( int    );
integer& operator /= ( int    );
integer& operator %= ( int    );
integer& operation ( integer_unary );
integer& operation ( integer_binary, const integer& );
integer& resize( slice );
integer& set   ( slice );
integer& clear ( slice );
boolean  test  ( slice ) const;
slice    scan  ( slice = INTEGER_SCAN_START ) const;
friend slice    length ( const integer& );
friend slice    bits   ( const integer& );
friend int      bitsum ( const integer& );
friend int      sign   ( const integer& );
```

```
    friend boolean  compare( const integer&, int );
    friend boolean  compare( const integer&, const integer& );
    friend integer  min    ( const integer&, const integer& );
    friend integer  max    ( const integer&, const integer& );
    friend ostream& operator << ( ostream&, integer& );
    friend istream& operator >> ( istream&, integer& );
};
//-------------------------------------------------------------------------//
#endif
```

П1.4. Заголовочный файл MATRIX.H

```
#if !defined( _MATRIX_ )
#define _MATRIX_
#include <iostream.h>
#if !defined( _ARRAY_ )
#include <array.h>
#endif
//-------------------------------------------------------------------------//
#define MATRIX_LENGTH_MAX INDEX_MAX
typedef int type;
typedef type (*matrix_binary)( type, type );
//-------------------------------------------------------------------------//
class matrix : public array {
private:
    index       row;
    index       col;
protected:
    static boolean set_range   ( const matrix& );
    static type    minor_range ( index, index );
    static void    reset_range ( void );
    index  input  ( istream& );
    index  output ( ostream& ) const;
public:
    ~matrix( void );
    matrix( void );
    matrix( index, index );
    matrix( const matrix& );
    matrix( const matrix&, index, index );
    matrix  operator *  ( const type    ) const;
```

```
    matrix  operator *  ( const matrix& ) const;
    matrix  operator /  ( const type    ) const;
    matrix  operator %  ( const type    ) const;
    matrix  operator +  ( const type    ) const;
    matrix  operator +  ( const matrix& ) const;
    matrix  operator -  ( const type    ) const;
    matrix  operator -  ( const matrix& ) const;
    matrix& operator *= ( const type    );
    matrix& operator /= ( const type    );
    matrix& operator %= ( const type    );
    matrix& operator += ( const type    );
    matrix& operator += ( const matrix& );
    matrix& operator -= ( const type    );
    matrix& operator -= ( const matrix& );
    matrix& operator =  ( const type    );
    matrix& operator =  ( const matrix& );
    index   rows    ( void ) const;
    index   columns ( void ) const;
    type    det     ( void ) const;
    type    minor   ( index, index ) const;
    type&   element ( index, index ) const;
    matrix& resize  ( index, index );
    index   length  ( void ) const;
    type*   value   ( void ) const;
    friend  matrix   back      ( const matrix& );
    friend  ostream& operator<<( ostream&, const matrix& );
    friend  istream& operator>>( istream&, matrix& );
    friend  matrix cron(matrix&, matrix_binary, matrix&);
};
#endif
```

П1.5. Заголовочный файл POLINOM.H

```
#if !defined( _POLINOM_ )
#define _POLINOM_
#include "integer.h"
#include "matrix.h"
//-------------------------------------------------------------------------//
class coeficient : public integer {
private:
```

```
    slice       item;           // Разрядность коэффициента
protected:
    coeficient* link;           // Поле для связи в список
    coeficient& add    ( coeficient& );
public:
    coeficient  ( slice );
    coeficient  ( integer&, slice );
    ~coeficient ( void );
    integer&    value  ( void );
    slice       number ( void );
    coeficient* next   ( void );
};

//-------------------------------------------------------------------------//
class polinom : public coeficient {
protected:
    word        base;           // Значность логики
    word        functions;      // Количество функций
    word        arguments;      // Количество переменных
    matrix**    power;          // Определения степеней
    matrix&     baseX;          // Значности переменных
    matrix&     baseF;          // Значности функций
public:
    ~polinom( void );
    polinom ( void );
    polinom( int base, matrix power[],
             int func, matrix& coef );
    polinom( matrix& basex, matrix power[],
             matrix& basef, matrix& coef );
    polinom& operator = ( const polinom& );
    matrix&  value ( matrix& );
};
//-------------------------------------------------------------------------//
#endif
```

Пример реализации кратных вычислений

П2.1. Результаты логических исследований

Пусть задана система логических функций в табличной форме:

X_1	X_0	f_1	f_0
0	0	2	0
0	1	1	0
1	0	1	1
1	1	0	1
2	0	2	0
2	1	1	1

Синтезируем полиномиальную форму исходной системы. Переменная X_0 имеет значность 2, переменная X_1 – значность 3. Искомые полиномы представим в мультипликативном базисе:

$$p_j(X)=\sum_{i=0}^{2\cdot 3-1} a_{ij}\cdot\left(X_1^{i_1}\&X_0^{i_0}\right),\quad j\in\{0,\ 1\}.$$

Зададим ядро дискретного ортогонального преобразования (определения степенных операций) в виде булевых матриц:

$$W_1=\begin{bmatrix}1&0\\1&1\end{bmatrix},\qquad W_2=\begin{bmatrix}1&0&0\\1&1&0\\1&0&1\end{bmatrix},$$

и вычислим обратные им матрицы:

$$W_1^{-1}=\begin{bmatrix}1&0\\-1&1\end{bmatrix},\qquad W_2^{-1}=\begin{bmatrix}1&0&0\\-1&1&0\\-1&0&1\end{bmatrix}$$

Построим матрицу прямого ортогонального преобразования D и вычислим коэффициенты полиномов A для каждой из функции системы:

$$D=\begin{bmatrix}1&0&0\\-1&1&0\\-1&0&1\end{bmatrix}\otimes\begin{bmatrix}1&0\\-1&1\end{bmatrix}=\left[\begin{array}{cc|cc|cc}1&0&0&0&0&0\\-1&1&0&0&-1&1\\\hline-1&0&1&0&0&0\\1&-1&-1&1&0&0\\\hline-1&0&0&0&1&0\\1&-1&0&0&-1&1\end{array}\right],\quad A=D\times F=D\times\begin{bmatrix}2&0\\1&0\\1&1\\0&1\\2&0\\1&1\end{bmatrix}=\begin{bmatrix}2&0\\-1&0\\-1&1\\0&0\\0&0\\0&1\end{bmatrix}.$$

В итоге, с учетом определений степенных операций, получим искомое полиномиальное представление функций исходной системы:

$$\begin{cases} p_0(X)=X_1^1+X_1^2X_0; \\ p_1(X)=2-X_0-X_1^1. \end{cases}$$

П2.2. Программа для кратных вычислений

```
#include "integer.h"
#include "matrix.h"
#include "polinom.h"
#include <iostream.h>
#define POWER_MAX 10
//-------------------------------------------------------------------------//
void main example( void )
   {
//------ Задание аналитической конструкции ------//
   matrix basex;                        // Вектор значностей переменных
   matrix power[ POWER_MAX ];           // Массив степенных операций
   cin >> basex;                        // Ввод значностей переменных
   for( int i=0; i < basex.column(); i++ )
      cin >> power[i];                  // Ввод степенных операций
//------ Задание функций вычисляемой системы ------//
   matrix basef;                        // Вектор значностей функций
   matrix coef;                         // Матрица коэффициентов
   cin >> basef;                        // Ввод значностей функций
   cin >> coef;                         // Ввод матрицы коэффициентов
//------ Конструирование полинома ------//
   polinom P( basex, power, basef, coef );
//------ Кратные вычисления ------//
   matrix sets;                         // Матрица наборов переменных
   matrix results;                      // Матрица результатов
   cin >> sets;                         // Ввод наборов переменных
   result = P.value( sets );            // Кратные вычисления
   cout << result;                      // Вывод результата
   }
```

П2.3. Входной поток данных

```
1×2                 ← Матрица значностей переменных
2 3

2×2                 ← Матрица 1-й степенной операции
1 0
1 1

3×3                 ← Матрица 2-й степенной операции
1 0 0
1 1 0
1 0 1

1×2                 ← Матрица значностей функций
2 3

1×6                 ← Матрица коэффициентов 1-й функции
0 0 1 0 0 1

1×6                 ← Матрица коэффициентов 2-й функции
2 -1 -1 0 0 0

2×3                 ← Матрица 3-х наборов 2-х переменных
0 1 2
1 1 1
```

П2.4. Выходной поток данных

```
2×3                 ← Матрица значений 2-х функций на 3-х наборах
0 1 1
1 0 1
```

Язык программирования LOGIC

П3.1. Формальная грамматика языка LOGIC

```
<программа> ::=
        <значность> <операции> <объявления> <определения>
<значность> ::=
        base <константа> ;
<операции> ::=
        <унарные_операций> <бинарные_операций>
<унарные_операций> ::=
        unary <определения_операций> ;
<бинарные_операций> ::=
        binary <определения_операций> ;
<определения_операций> ::=
        <знак_операции> ( <список_констант> )
        <знак_операции> ( <список_констант> ) , <определения_операций>
<список_констант> ::=
        <константа>
        <константа> <список_констант>
<объявления> ::=
        variable <список_переменных> ;
<список_переменных> ::=
        <идентификатор>
        <идентификатор> , <список_переменных>
<определения> ::=
        function <список_выражений> ;
<список_выражений> ::=
        <выражение>
        <выражение> , <список_выражений>
<выражение> ::=
        <терм>
        <терм> <бинарная_операция> <выражение>
<терм> ::=
        <унарная_операция> <терм>
        <первичное>
<первичное> ::=
        <константа>
```

```
        <идентификатор>
        ( <выражение> )
<константа> ::=
        <цифра>
        <цифра><константа>
<идентификатор> ::=
        <буква>
        <идентификатор><буква>
        <идентификатор><цифра>
<цифра> ::=
        0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<буква> ::=
        A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P
        Q | R | S | T | U | V | W | X | Y | Z | _
        a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p
        q | r | s | t | u | v | w | x | y | z
<лексема> ::=
        <пробельный_знак>
        <знак_пунктуации>
        <знак_унарной_операции>
        <знак_бинарной_операции>
        <константа>
        <идентификатор>
        <ключевое_слово>
<пробельный_знак> ::=
        <пробел>
        <новая_строка>
        <перевод_строки>
        <перевод_формата>
        <горизонтальная_табуляция>
        <вертикальная_табуляция>
        <комментарий>
<знак_пунктуации> ::=
        <конец_файла> | , | ; | ( | )
<ключевое_слово> ::=
        base | unary | binary | variable | function
<комментарий> ::=
        /* <последовательность_знаков> */
```

П3.2. Пример программы на языке LOGIC

```
base 3;                 /* значность алгебры логики, k=3 */
unary                   /* определение унарных операций */
     ~ ( 1 2 0 ),       /* циклическое отрицание z = (x+1) mod 3 */
     - ( 2 1 0 );       /* "зеркальное" отрицание z = (2 - x)    */
binary                  /* определение бинарных операций */
     & ( 0 0 0          /* конъюнкция z = min(x, y) */
         0 1 1
         0 1 2 ),
     | ( 0 1 2          /* дизъюнкция z = max(x, y) */
         1 1 2
         2 2 2 );
variable                /* объявление независимых переменных */
     x0, x1;
function               /* определение логических функций */
     -~x0 & ~x1,
     ~x1 | -x0;
```

П3.3. Пример результата вычислений

```
9x2 logic.prg
2 1
1 0
1 1
2 1
2 0
2 2
2 0
1 0
0 0
```