ИНФОРМАТИКА

2004 **октябрь-декабрь** **№4**

УДК 681.32

П.Н. Бибило, Л.Д. Черемисинова

АВТОМАТИЗАЦИЯ ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ

Дается краткий обзор теоретических результатов, полученных в лаборатории логического проектирования ОИПИ НАН Беларуси в области автоматизации логического проектирования дискретных устройств и сверхбольших интегральных схем. Сообщается также о разработанных в лаборатории комплексах программ и системах логического проектирования.

Введение

Лаборатория системного программирования и логического синтеза (ныне логического синтеза) была создана в 1971 г. чл.-корр. АН БССР А.Д. Закревским. Сотрудниками лаборатории проводились исследования в областях создания логико-комбинаторного базиса логического проектирования; автоматизации программирования логических задач; разработки языков описания дискретных устройств управления и средств автоматизации отладки описаний; программирования микропроцессоров и логических контроллеров; моделирования и синтеза тестов; логического синтеза дискретных управляющих устройств и интегральных микросхем. В качестве базиса синтеза использовались микросхемы от малой до сверхбольшой степени интеграции На основе теоретических исследований, важнейшие результаты которых представлены в 17 монографиях и более чем в тысяче статей, был разработан ряд программных комплексов автоматизации алгоритмического и логического проектирования управляющих устройств и сверхбольших интегральных схем (СБИС). Ниже приводится краткий обзор наиболее важных из полученных теоретических и практических результатов.

1. Языки программирования и проектирования

Языки программирования ЛЯПАС, ЛЯПАС-М [1, 2] по набору операций, типам данных и структуре программных модулей были ориентированы на эффективное представление алгоритмов решения комбинаторных задач логического проектирования. На базе этих языков был создан ряд систем программирования для вычислительных машин Минск-22, -32, БЭСМ-6 [3], ЕС [1, 2], IBM PC. Обзор работ лаборатории по системному программированию приведен в [4].

Язык SF [5] был разработан для описания сложных дискретных систем в виде иерархии функционально-структурных описаний составляющих их блоков. Функциональные описания представляются системами логических уравнений (скобочных форм в булевом базисе – И, ИЛИ, НЕ) или дизъюнктивных нормальных форм (ДНФ) булевых функций.

Языки АЛУ и ПРАЛУ описания параллельных алгоритмов управления. В 1980-х гг. в лаборатории были разработаны языки АЛУ и ПРАЛУ [6, 7], позволяющие описывать широкий класс алгоритмов логического управления (последовательных, параллельных, циклических) и допускающие представление алгоритмов в виде иерархии взаимосвязанных подалгоритмов. Алгоритм на этих языках представляется в виде причинно-временных зависимостей между событиями, происходящими в технической системе. Основными операциями являются *ожидание* и *действие*. Первая операция сводится к ожиданию наступления некоторого события, вторая приводит к наступлению события в объекте управления. Событие представляется предикатом (в языке АЛУ) или конъюнкцией булевых переменных (в языке ПРАЛУ). Алгоритм управления представляется неупорядоченной совокупностью предложений, состоящих из одинаково помеченных цепочек «μ_i: $l_i \rightarrow \nu_i$», где l_i – линейный алгоритм, составленный из операций языка, μ_i и ν_i – подмножества натуральных чисел, служащих метками предложений. В процессе реализации алгоритма цепочки запускаются и выполняются независимо друг от друга. В расширенные версии языков введены арифметические переменные и операции над ними. Для прерывания некоторых параллельно выполняемых ветвей алгоритма в языки введена операция гашения.

В основе базовых ПРАЛУ-описаний, содержащих только операции ожидания и действия, лежит известный формализм сетей Петри. Такие описания прямо отображаются в элементы автоматного описания, для них развит мощный математический аппарат [7] анализа и реализации. В силу этого базовое подмножество языка ПРАЛУ используется в качестве внутреннего языка САПР систем управления [8 – 10].

Языки VHDL и Verilog, самые известные в мире языки описания проектов цифровых систем, используются в качестве входных языков программных комплексов, разработанных в лаборатории. Эта возможность обеспечивается с помощью конверторов описаний на данных языках (а также EDIF) на внутренний язык комплексов SF и обратно [11]. Проблемы моделирования и синтеза логических схем по спецификациям на языке VHDL рассмотрены в монографиях [12, 13].

2. Реализация параллельных алгоритмов логического управления

Разработана технология проектирования иерархических систем управления сложными техническими комплексами. Основное внимание уделено построению математических моделей, описывающих устройство управления на разных уровнях его проектирования: иерархическое ПРАЛУ-описание и *α-сеть* – на этапе разработки алгоритма функционирования; параллельный, секвенциальный и булев автоматы – на этапе автоматной реализации; программа на языке контроллера – на этапе программной реализации. Предложены комплексы методов и программ, решающих задачи верификации, моделирования и визуализации поведения системы управления.

2.1. Разработка алгоритмов логического управления

Проектирование управляющей системы начинается с разработки алгоритма управления и его верификации, традиционно разделяемой на *синтаксическую* и *семантическую*. Первая, кроме проверки описания на соответствие синтаксису языка, включает в себя также и проверку формальных синтаксических условий, образующих понятие *корректности* [7]. Семантический анализ (отладка алгоритма [6]) требует участия проектировщика, владеющего информацией о планируемом поведении системы управления. Алгоритм может быть корректным, но не реализовать задуманное поведение. В работе [14] приведен обзор работ сотрудников лаборатории по интерактивной разработке алгоритмов управления: проверке корректности, диагностике, моделированию, композиции и декомпозиции алгоритмов, генерации корректных ПРАЛУ-описаний, поиску отношения параллельности на множестве мест алгоритма управления.

Проверка корректности алгоритма управления. *Корректными* названы алгоритмы, обладающие свойствами непротиворечивости, устойчивости, безызбыточности, восстанавливаемости и самосогласованности [7]. Проверка каждого из этих свойств представляет нетривиальную комбинаторную задачу. Традиционно рассматриваемые в лаборатории подходы к проверке корректности алгоритма управления заключаются в построении его «скелета» – α-сети, являющейся подклассом ординарных сетей Петри (расширенной сетью свободного выбора). Показано [7], что проверка корректности алгоритма управления сводится в значительной степени к проверке *живости* и *безопасности* α-сети: если она безопасна, то алгоритм самосогласован; если алгоритм безызбыточен и восстанавливаем, то α-сеть жива. Проверка живости α-сети выполняется непосредственно на ее графе достижимости путем перебора пар «разметка, переход». Снижение трудоемкости метода достигается за счет использования понятий ловушки и тупика и исследования отношений между ними. Эта задача сводится к решению логических уравнений.

Трудоемкость решения задачи безопасности α-сети, связанной с построением множества всех достижимых разметок, существенно понижается при использовании операций редукции (доказано [15], что редукция живых и безопасных α-сетей посредством этих операций всегда идет до конца). Предложен ряд редукционных методов проверки алгоритма управления на корректность, позволяющих эффективно осуществлять частичную отладку алгоритмов (без учета информационных связей между ветвями алгоритма управления). Полная проверка корректности алгоритма управления, требующая проверки его самосогласованности и устойчивости (т. е.

анализа информационного взаимодействия между цепочками), решается на α-сети, расширенной за счет информации, извлекаемой из операций ожидания и действия.

Диагностика алгоритма управления заключается в локализации некорректности алгоритма. Систематические методы решения этой задачи в силу ее сложности развиты слабее, рассматриваются либо разные частные случаи, либо эвристические методы [14].

Отладка алгоритмов логического управления путем моделирования используется для их оптимизации или верификации. Программа моделирования предоставляет разработчику системы управления средства для воспроизведения ее работы в различных ситуациях. В лаборатории разработано несколько программ моделирования алгоритмов управления на языке ПРАЛУ, различающихся диалоговым языком управления, предположениями о длительности выполнения операций действия, отображением на экране динамики точек активности в тексте алгоритма, а также платформой реализации (ЕС ЭВМ [6, 10], БЭСМ-6 [10], ПЭВМ).

Структурные преобразования алгоритмов логического управления включают в себя равносильные преобразования на уровне ПРАЛУ-описаний. Самые простые из них сокращают числа меток, цепочек, операций ожидания и действия [7, 16]. *Декомпозиция* параллельного алгоритма заменяет его множеством подалгоритмов, связь между которыми имеет чисто информационный характер (через общий набор переменных). Разработаны методы декомпозиции алгоритма на ПРАЛУ (по заданному разбиению на множестве предложений) в сеть не связанных между собой [17], последовательных [7] и минимально взаимодействующих подалгоритмов.

2.2. Реализация параллельных алгоритмов логического управления

Схемная реализация параллельного алгоритма логического управления рассматривается как процесс эквивалентных преобразований его моделей: алгоритм управления → параллельный автомат → секвенциальный или булев автомат → логическая схема [7].

Параллельный автомат как автоматная модель параллельного алгоритма управления впервые предложен А.Д. Закревским. Параллельный автомат с абстрактным внутренним и структурными входными и выходными состояниями получается из алгоритма управления путем введения дополнительных меток в его описание и «разрезания» его цепочек на фрагменты автоматного описания: «μ_i: $-k_i^{вх}\rightarrow$ $k_i^{вых}\rightarrow \nu_i$». Отличительной особенностью параллельного автомата является то, что переходы в нем осуществляются между множествами (μ_i и ν_i) состояний, называемых частичными. По аналогии с последовательным параллельный автомат также может быть *Мили* или *Мура*, синхронным или асинхронным.

В качестве структурных моделей алгоритма управления, получаемых после кодирования частичных состояний параллельного автомата, рассматриваются *секвенциальный* и *булев* автоматы, выбираемые в зависимости от базиса синтеза и интерпретации операций действия алгоритма управления. *Секвенциальный* автомат есть система секвенций $f_i(X,Z) \mid\!- k_i(Y,Z)$, где X, Y и Z – множества входных, выходных и внутренних переменных, f_i – некоторая функция и k_i – конъюнкция. В зависимости от вида функций f и k определены разные типы секвенций и секвенциальных автоматов (точечный, простой функциональный), а также методы их оптимизационных преобразований [18]. Язык секвенций был описан как язык задания алгоритмов функционирования устройств управления [18]. Были предложены методы анализа, моделирования, синхронной и асинхронной реализации систем секвенций, минимизации инерционного секвенциального автомата (реализующего параллельный алгоритм управления) [7, 16].

Кодирование состояний параллельных автоматов. Принципиальным отличием параллельного автомата от последовательного является то, что в любой момент времени он может находиться в нескольких частичных состояниях, что учитывается на этапе их кодирования. Введено понятие интервально-вытесняющего кода и требований [7], которым он должен удовлетворять. В частности, параллельные частичные состояния должны кодироваться неортогональными троичными векторами, непараллельные – ортогональными.

Предложенные методы кодирования частичных состояний синхронных параллельных автоматов делятся [16] на *итерационные*, *блочного кодирования*, *декомпозиционные*, а также на основанные на *покрытии графа* или *матрицы* непараллельности. В качестве критериев оптимизации при кодировании принимается число вводимых кодирующих переменных и простота

функций возбуждения элементов памяти. В частности, число необходимых кодирующих переменных сокращается за счет использования в качестве элементов кода значений уже имеющихся выходных и внутренних переменных [7, 16]. Более глубокая оптимизация достигается при учете свойств объекта, управляемого данным параллельным автоматом [16].

Асинхронная реализация параллельного автомата ставит дополнительное условие при кодировании его частичных состояний – устранить опасные состязания между элементами памяти. В работе [16] показано, что для исключения опасных состязаний на переходах между множествами частичных состояний, осуществляемых при одном входном состоянии, достаточно обеспечить отсутствие опасных состязаний на одной паре элементарных переходов, которые происходят между попарно непараллельными частичными состояниями из этих множеств. На этом утверждении основан ряд точных и приближенных методов кодирования частичных состояний асинхронных параллельных автоматов. Их можно разделить на *декомпозиционные* методы, методы *блочного*, *унитарного* и *противогоночного кодирования*.

Программная реализация алгоритмов управления. При реализации языка ПРАЛУ на однопроцессорной вычислительной системе процесс трансляции описания на языке ПРАЛУ в программу рассматривался как последовательная замена одной семантики языка ПРАЛУ другой, более детальной. В качестве средства разработки трансляторов языка ПРАЛУ применялся *инструментальный комплекс* [19] и входящая в его состав *система построения трансляторов*, включающая языки и средства для программирования блоков анализа и генерации объектного кода. В качестве целевых при реализации алгоритмов на языке ПРАЛУ использовались системы с микропроцессорами К580, X86, Intel 80C51.

Показано, что выполнение программы для контроллера с языком типа релейно-контактных схем (РКС) моделируется асинхронным инерционным функциональным секвенциальным автоматом (ФСА) [16]. Процесс трансляции алгоритма управления в программу рассматривался как последовательность оптимизирующих преобразований его моделей: алгоритм управления $\rightarrow$ асинхронный параллельный автомат Мура $\rightarrow$ асинхронный ФСА $\rightarrow$ программа для контроллера в символьном виде $\rightarrow$ программа на языке РКС выбранного контроллера.

3. Проектирование дискретных управляющих и вычислительных устройств

3.1. Задачи логического проектирования

Исследовались традиционные для логического проектирования задачи анализа, синтеза и верификации.

Анализ логических схем. Задачи анализа состоят в вычислении функций, реализуемых схемой, нахождении реакции схемы на заданное входное воздействие (тест), поиске входного сигнала, вызывающего заданную реакцию, и др. Эти задачи служат базовыми при автоматизации построения тестов, при обнаружении и диагностике неисправностей в дискретных устройствах. Более детальные сведения о поведении дискретного устройства может дать моделирование, которое учитывает переходные процессы, не отображаемые в функциональных моделях. С его помощью определяется поведение устройства на заданной входной последовательности. В монографии [20] предложена базовая система операций над булевыми функциями, к которым сводятся типичные задачи анализа, моделирования и построения тестов.

Верификация. В общем виде задача верификации формулируется следующим образом: даны два описания (например, логических схем), требуется проверить, являются ли они функционально эквивалентными. Для случая комбинационных схем эта задача сводится к известной комбинаторной задаче «*выполнимость*», в решении которой достигнуты значительные успехи [21], в том числе и на мировом уровне. Последние из разработанных программ (SAT-solvers) [22] заняли призовые места (в том числе и первое) в международном конкурсе аналогичных программ.

Синтез логических схем является центральной проблемой этапа логического проектирования. После ее решения определяются основные характеристики дискретных устройств: сложность (стоимость, площадь схемы), быстродействие, тестопригодность. *Задача синтеза* формулируется следующим образом. Дано функциональное описание S дискретного устройства и ба-

зис логических элементов. Требуется построить из элементов этого базиса логическую схему, функционально эквивалентную *S*. Задача синтеза традиционно разбивается на три этапа [12].

1. *Высокоуровневый синтез* – переход от функционального описания на языке высокого уровня (VHDL, Verilog, ПРАЛУ) к функционально-структурному описанию на языке SF (сходном с RTL, который принят в западной литературе).

2. *Технологически независимая оптимизация* – синтез и оптимизация многоуровневой логической сети в базисе простых вентилей технологически независимого базиса.

3. *Технологическое отображение* – преобразование (с оптимизацией) технологически независимой схемы в функционально эквивалентную схему в целевом технологическом базисе.

Задачи логического проектирования могут быть декомпозированы на подзадачи из одного общего набора логико-комбинаторных задач на матрицах и графах [23 – 25].

Элементная база логического синтеза. Выпускаемые в настоящее время БИС можно разделить на два класса: *стандартные* и *специализированные*. К первым относятся микросхемы памяти, регистры, элементы малой (на основе простых вентилей) и средней (программируемые логические матрицы (ПЛМ), мультиплексоры, постоянные запоминающие устройства (ПЗУ)) степени интеграции и т. д. Специализированные БИС делятся на полностью заказные (на основе регулярных матричных структур); полузаказные – матрицы логических вентилей (МЛВ) или базовые матричные кристаллы (БМК); программируемые пользователем интегральные схемы (ПЛИС) – программируемые пользователем вентильные матрицы (ППВМ или FPGA) и сложные ПЛИС на основе матричной логики (ПЛМ и программируемых матриц логики (ПМЛ)). По мере появления новой элементной базы сотрудниками лаборатории разрабатывались новые эффективные методы анализа и синтеза логических схем в этих базисах.

3.2. Технологически независимая оптимизация

Задача состоит в построении и оптимизации многоуровневой схемы (или представлении системы функций) в базисе элементов, не обязательно привязанных к конкретной библиотеке СБИС. Обычно это простые вентили одного или нескольких типов. Основное внимание уделяется решению следующих классов задач технологически независимой оптимизации.

Минимизация системы булевых функций в классе ДНФ. При синтезе схем в разных технологических базисах полезны следующие типы минимизации систем полностью и частично определенных функций: *совместная* минимизация функций системы (модификация программы ESPRESSO, метод конкурирующих интервалов и разные его модификации [18, 27, 26]), применяемая при минимизации площади ПЛМ [18, 28, 29 и др.] и синтезе схем в базисах нерегулярной логики [8, 26, 30]); *раздельная* минимизация функций системы, наиболее естественная при реализации на ПМЛ и регулярных МОП-схемах [26, 31]); минимизация (совместная или раздельная) функций системы с *учетом их полярности* (возможности инверсирования), полезная при реализации на регулярных матричных структурах с биполярными выходами.

Разработанные в лаборатории логического проектирования программы оказались конкурентоспособными с лучшими из зарубежных программ минимизации [32].

Минимизация числа аргументов системы функций [18, 28]. При синтезе сетей на базе ПЛИС (ПМЛ, ПЛМ, ПЗУ) с ограниченным числом входных шин полезна процедура сокращения числа различных литералов в системе в целом или в некоторых функциях.

Минимизация числа ДНФ в системе достигается за счет сокращения числа внутренних переменных системы уравнений. Задача сводится к элиминации этих переменных (раскрытию скобок), полной или частичной в зависимости от критериев оптимизации и ограничений [30, 33].

Минимизация сложности системы булевых функций или ДНФ сводится к получению представления функций исходной системы в виде суперпозиции более простых функций из одного общего набора. Декомпозиция выполняется на основе *алгебраических* и *булевых* операций.

Декомпозиция на алгебраическом уровне. Методы этого класса работают с системами ДНФ как с системами алгебраических выражений. К этому классу относятся методы алгебраического деления, или *факторизации*, обеспечивающие получение факторизованной системы ДНФ, содержащей минимальное число букв или реализуемой схемой с минимальным числом вентилей заданного типа [26, 33]. Частные виды факторизации используются при представле-

нии логических уравнений многоуровневыми схемами в базисах элементов БМК [30], регулярной логики типа ПЛИС и матриц Вайнбергера [5, 33].

Декомпозиция на булевом уровне. Методы этого класса работают на уровне наборов значений аргументов и интервалов в булевом пространстве области задания системы полностью или частично определенных функций. Сюда относятся: все типы *общих декомпозиций*, введенных в работах Ашенхерста, Кертиса, Рота и Карпа; *дизъюнктивные* и *конъюнктивные* разложения; разложения в базисе *фиксированных функций*.

Декомпозицией булевой функции или системы булевых функций называется нахождение ее представления в виде суперпозиции функций, более простых в некотором смысле. Критерием простоты в методах чаще всего служит число аргументов компонентных функций (они должны быть меньше числа аргументов исходной функции). На булевом уровне декомпозиция функции (или системы) $f(x)$ состоит в поиске ее функционального разложения по заданному $(k+1)$-блочному покрытию $Y^1, \ldots, Y^k, Z$ множества аргументов X ($Y^1 \cup Y^1 \cup \ldots \cup Y^k \cup Z = X$) в виде $f(X) = g(\boldsymbol{h}^1(Y^1), \ldots, \boldsymbol{h}^k(Y^k), Z)$, где $\boldsymbol{h}^i(Y^i) = (h_1^i(Y^i), \ldots, h_{pi}^i(Y^i))$, $i = 1, \ldots, k$.

В работах лаборатории основное внимание уделялось двум частным видам декомпозиции по двухблочному разбиению Y, Z множества переменных: *многократной* $f(x) = g(h^1(Y),\ldots, h^p(Y), Z)$ и *простой* $f(x) = g(h(Y), Z)$ непересекающимся декомпозициям. В монографии [28] (второй в мире, посвященной декомпозиции булевых функций) методы декомпозиции были обобщены для интервальных матричных форм задания систем функций, описаны новые виды совместных разложений и методы их построения, рассмотрено применение декомпозиции к синтезу схем из настраиваемых и программируемых элементов (ПЛМ, ПЗУ, универсальных логических модулей). Перебор решений задачи декомпозиции сведен также к перебору корней логического уравнения, выражающего условия существования функционального разложения. Методы декомпозиции незаменимы при синтезе схем из мультиплексоров, ПЛМ, ППВМ.

Декомпозиция в базисе фиксированных функций представляет собой частный случай общей декомпозиции, когда тип компонентных функций задан (И-НЕ, ИЛИ-НЕ, И-ИЛИ-НЕ [30], мультиплексоры [34]). Эти методы отличаются высоким быстродействием и качеством решений даже по сравнению с методами декомпозиции на алгебраическом уровне.

Дизъюнктивные и конъюнктивные разложения являются важными частными случаями общей декомпозиции, когда функция g является многовходовой дизъюнкцией или конъюнкцией. Разложения используются при синтезе логических схем в базисе стандартных ПЛМ [35], ПЛИС, имеющих матричную структуру типа ПЛМ, ПМЛ, регулярных МОП-схем. Компоненты дизъюнктивно-конъюнктивного разложения частичной булевой функции строятся путем деления областей нулевых и единичных значений этой функции. Задача реализации системы ДНФ в базисах ПЛИС сводится к дизъюнктивному разложению этой системы на минимальное число ограниченных по всем параметрам систем ДНФ.

Декомпозиция автоматов (синхронных и асинхронных) в сеть последовательно или параллельно связанных компонентных автоматов с меньшим (по сравнению с исходным автоматом) числом внутренних состояний является еще более сложной задачей [36], чем декомпозиция булевых функций. Однако ее выполнение на ранних стадиях проектирования дискретного устройства в ряде случаев может дать и больший эффект, чем декомпозиция булевых функций.

3.3. Технологическое отображение

Технологическое отображение является конечным этапом логического синтеза, определяя качество искомой сети в технологическом элементном базисе (типа БМК). Это отображение выполняется исходя из структуры логических выражений, полученных на этапе технологически независимой оптимизации и рассматриваемых как многоуровневая многовыходная логическая сеть в базисе вентилей, называемая объектной сетью. В процессе технологического отображения объектная сеть преобразуется в эквивалентную логическую сеть из элементов целевого базиса путем покрытия фрагментов объектной сети элементами технологической библиотеки. Задача покрытия NP-трудна, разработанные сотрудниками лаборатории методы покрытия (и точные, и приближенные) разделяются на структурные и функциональные.

Методы структурного покрытия сводят задачу технологического отображения к покрытию ориентированного ациклического графа, задающего исходную объектную сеть, ацикличе-

скими подграфами, задающими структуры библиотечных элементов в том же вентильном базисе, что и покрываемая сеть. Предложенные методы структурного покрытия объектной сети ориентированы на повышение качества покрытия за счет отказа от декомпозиции объектного графа в лес деревьев (разрешения перекрытия) и быстродействия за счет разбиения покрывающих графов на классы, топологической сортировки и упорядочения (по эффективности) покрывающих графов.

Методы функционального покрытия [12] основаны на сравнении функций (проверке на эквивалентность или реализуемость), реализуемых фрагментами объектной сети и библиотечными элементами. Основные операции методов заключаются в выделении фрагмента объектной сети, вычислении и сравнении функции фрагмента с библиотечной функцией. Наиболее сложной (комбинаторно) является задача сравнения функций. Сокращение перебора при сравнении функций достигается проверкой таких свойств функций, как однородность, симметричность по переменным, линейность и т. д. Если все эти свойства для сравниваемых функций совпадают, то функции подвергаются более дорогостоящей проверке на эквивалентность.

4. Синтез регулярных структур заказных СБИС

В заказных СБИС управляющая логика реализуется в виде программируемых структур, имеющих регулярную топологию и потому допускающих автоматическую генерацию их топологии структурному (или функциональному) описанию. Такими структурами, в частности, являются двухмерные структуры типа ПЛМ, матриц Вайнбергера (МВ), транзисторных матриц, регулярных схем последовательно соединенных МОП-транзисторов (РМОП-схем) [5]. Основным критерием оптимизации СБИС на основе регулярных структур является площадь кристалла.

4.1. Методы синтеза регулярных матричных структур

Синтез структуры ПЛМ. ПЛМ предназначена для реализации систем ДНФ и состоит из двух транзисторных матриц И и ИЛИ с варьируемыми размерами. Минимизация площади ПЛМ в значительной степени сводится к задаче совместной минимизации системы булевых функций в классе ДНФ с учетом возможности изменения их полярности.

Синтез структуры РМОП-схемы. РМОП-схема так же, как и ПЛМ, реализует систему ДНФ или их инверсий. Ее ядром является матрица И, разделенная на секции, распределяемые между взаимно непересекающимися столбцами матрицы ИЛИ. Конструктивной особенностью РМОП-структур является также ограничение на число транзисторов в каждой из строк матрицы И. Синтез структуры РМОП-схемы с минимальной площадью сводится к раздельной минимизации в классе ДНФ функций системы и последующей факторизации. В работе [37] описаны модификации РМОП-схем на основе факторизации и свертки, а также методы их синтеза.

Синтез структуры матрицы Вайнбергера. Конструктивно матрица Вайнбергера состоит из столбцов последовательно связанных транзисторов. Несколько связанных между собой столбцов, снабженных одним нагрузочным транзистором, реализуют функцию отрицания дизъюнкции конъюнкций, получаемых на связанных столбцах. Конструктивной особенностью матрицы Вайнбергера является ограничение на число перемычек на пересечении с каждой из строк и на число связываемых между собой столбцов. Таким образом, синтез логической структуры матрицы Вайнбергера сводится к синтезу укладываемой на нее многоуровневой схемы в базисе «отрицаний ДНФ» с ограниченными параметрами [38].

4.2. Топологическая оптимизация регулярных матричных структур

Свертка матричных структур. Минимизация площади матричной структуры на этапе топологического проектирования достигается за счет свертки столбцов и строк матричной структуры – разрыва шин и реализации на одной вертикальной (или горизонтальной) шине двух или более исходных столбцов (или строк). В зависимости от числа разрывов рассматривается *простая* и *многократная* свертка. Свертка может быть *одномерной* (столбцовой или строчной) или *двухмерной* (столбцово-строчной). Кроме того, рассматривается интересный для практики частный вид свертки – *двудольная* свертка, все разрывы которой находятся на одном уровне. Свертке могут быть подвергнуты любые регулярные матричные структуры. Для ПЛМ-

структур возможны все виды сверток для обеих матриц И и ИЛИ, для матриц Вайнбергера – многократная строчная свертка, для РМОП-структур – строчная свертка матрицы И ограниченного типа.

Формальной топологической моделью матричной структуры при решении задачи свертки является булева матрица, ее единичные элементы говорят о наличии транзисторов на пересечении строк и столбцов. В этих терминах формулируются понятия свертываемого набора – набора столбцов или строк, укладываемых на одну шину, и их множества – множества свертки. Упорядоченное множество столбцовой свертки (УМС) реализуемо топологически, если порождаемое им отношение порядка на множестве строк есть отношение частичного порядка. Реализуемое УМС содержит всю необходимую для свертки матричной структуры информацию. Задача свертки состоит в нахождении реализуемого УМС максимальной мощности. Наиболее сложным этапом, ответственным за экспоненциальную сложность задачи свертки, является проверка УМС на реализуемость, она сводится к проверке на асимметричность транзитивного замыкания отношения частичного порядка, порождаемого на множестве строк (или столбцов).

Методы свертки матричных структур. В зависимости от используемых способов поиска УМС при простой или многократной свертке предложенные алгоритмы можно разделить на *итерационные* [38], основанные на последовательном построении свертываемых пар или наборов, лучших по некоторым критериям, и добавлении их в формируемое множество свертки; *графовые*, основанные на предварительном построении графа, задающего отношение совместимости свертываемых пар на множестве столбцов и/или строк, и поиске наибольшего полного графа, порождающего реализуемое множество свертки; основанные на приведении матрицы, задающей отношение свертываемости на множестве столбцов, к диагональному виду путем ее *сжатия* [39]; итеративные, основанные на процедуре *моделирования отжига* [41], где текущее состояние матрицы изменяется путем перестановки столбца (или строки) в соответствии с принятой оценочной функцией.

5. Научно-технические разработки

Ниже приводится краткое описание наиболее значимых (из разработанных в лаборатории) программных комплексов автоматизации алгоритмического и логического проектирования управляющих устройств и сверхбольших интегральных схем.

«Автомат-74» [26] – система автоматического синтеза асинхронных логических схем в базисе микросхем серии К133 малой степени интеграции, реализующих конечные автоматы.

ЛОГИКА-М [8] явилась непосредственным развитием разработанного ранее пакета ЛОГИКА-1 [42] на ЭВМ серии ЕС (и БЭСМ-6) и предназначалась для синтеза и оптимизации синхронных и асинхронных логических схем в базисах ПЛМ и микропроцессоров, элементов малой и средней степени интеграции. Кроме традиционных функциональных описаний проектируемых устройств (таблиц переходов и выходов, систем секвенций, полностью и частично определенных булевых функций), в системе появилась возможность использования языка ПРАЛУ.

Система логического проектирования дискретных устройств на программируемых матричных БИС [29] (на ПЭВМ и СМ-4) предназначена для синтеза и оптимизации сетей из стандартных и секционированных ПЛМ, ПЗУ. Исходные данные – секвенциальные и микропрограммные автоматы, системы матричных и скобочных форм булевых функций.

Комплекс программ синтеза комбинационных схем в базисе ПЛМ и МЛВ [30] (на БЭСМ-6) явился развитием подсистемы синтеза комбинационных схем пакета ЛОГИКА-М с ориентацией на базис ПЛМ и МЛВ, в библиотеку которых включены вентили типа И, ИЛИ, И-НЕ, ИЛИ-НЕ, И-ИЛИ-НЕ.

«Диана» [43] – диалоговая система (на ЕС-ЭВМ, СМ-4) анализа и моделирования сложных дискретных систем, имеющих многоуровневое иерархическое функционально-структурное описание и выполненных на микросхемах разной степени интеграции. Система обеспечивала моделирование на заданной входной последовательности исправного дискретного устройства, а также устройства с неисправностями, анализ неисправностей и построение проверяющих тестов.

«*Локус-Л»* [9] – диалоговая система (на ЕС-ЭВМ), предназначенная для автоматизации проектирования логики управляющих систем (типа автоматических станочных линий) на базе

программируемых логических контроллеров. Входным языком системы являлся язык ПРАЛУ, базовой моделью контроллера – контроллер PLC (с языком РКС). Система содержала средства разработки и отладки алгоритмов функционирования системы управления, синтеза и верификации программ на языке РКС, имела средства для настройки на заданный тип контроллера.

ИСАПР [44] – исследовательская САПР (на ПЭВМ) дискретных управляющих устройств, включившая в себя все выполненные ранее программные разработки лаборатории и служившая полигоном для разработки и отладки прикладных программных комплексов. В ее состав входил ряд подсистем: разработки и отладки алгоритмов функционирования, анализа, оптимизации и синтеза логических схем, синтеза программ для микропроцессоров. Входными данными служили ПРАЛУ-описания, ГСА, системы секвенций, скобочных и матричных форм булевых функций. В качестве элементного базиса использовались БИС ППЗУ, ПЛМ, микропроцессоры К580 и элементы малой степени интеграции.

ЛОГИКА СБИС [31] – система (на ПЭВМ), ориентированная на синтез регулярных структур СБИС на базе макроэлементов ПЛМ, ПЛМ с дешифраторами на входах (ДПЛМ), ПЗУ, РМОП-схем. Для оптимизации (по площади) структур макроэлементов использовались методы свертки и минимизации систем функций в классе ДНФ.

SCAS (Silicon Compiler of National Academy of Sciences of Belarus) [5] – кремниевый компилятор (на ПЭВМ в среде MS DOS и Windows). Компилятор предназначен для синтеза топологии управляющих функциональных блоков заказных цифровых СБИС – макроэлементов СБИС, имеющих регулярную топологию (ПЗУ, ПЛМ, РМОП-схемы, матрицы Вайнбергера, транзисторные матрицы). Компилятор поддерживается топологическими библиотеками КМОП-технологии для фиксированных норм проектирования НПО «Интеграл».

«Синтез ПЛИС» – система (на ПЭВМ), ориентированная на проектирование в базисе ПЛИС (параметрически описываемых ПЛМ, ПЗУ, ПМЛ, а также библиотечных элементов полузаказных СБИС на основе БМК). Базовым языком описания схем является язык SF, имеются средства конвертирования данных из языка EDIF 2.0.0, служащего обменным форматом в зарубежных САПР. Система имеет развитые средства работы с иерархическими описаниями проектируемых схем, оптимизации двух- и многоуровневых логических описаний, верификации. Важная особенность системы – наличие интеллектуальной поддержки процесса проектирования, разработанной на основе продукционно-фреймовой модели представления знаний [45].

«Синтез БМК» – подсистема [40] системы «Синтез ПЛИС», предназначенная для перепроектирования схем, реализованных на ППВМ (FPGA) средствами САПР Xilinx Foundation F1.5, в схемы, реализуемые на БМК серии К1574.

«Custom Logic» – система [33] (на ПЭВМ), являющаяся дальнейшим развитием кремниевого компилятора SCAS и опирающаяся на новые эффективные алгоритмы и программы решения оптимизации функциональных и структурных описаний объектов проектирования. Устройство управления может быть реализовано не только одним макроэлементом, но и сетью взаимосвязанных макроэлементов разных типов. Система обеспечивает проектирование устройств управления из макроэлементов ПЛМ, матриц Вайнбергера, ПЗУ, РМОП-схем.

ЭКСИЛОР – экспертная система логического распознавания в пространстве дискретных признаков [46] (на ПЭВМ). Знания о предметной области выражаются в виде импликативных закономерностей. Распознавание сводится к решению логических уравнений над конечными предикатами.

Заключение

Разработанные в лаборатории логического проектирования оригинальные языки программирования и проектирования, методы логического синтеза, а также программные комплексы являются крупным вкладом в теорию и практику логического проектирования. Они конкурентоспособны с лучшими из мировых достижений в данной научной области исследований.

Список литературы

1. Закревский А.Д., Торопов Н.Р. Система программирования ЛЯПАС-М. – Мн.: Наука и техника, 1978. – 238 с.

2. Торопов Н.Р. Диалоговая система программирования ЛЕС. – Мн.: Наука и техника, 1982. – 264 с.

3. Томашев В.Ф. Мобильный ЛЯПАС-М-транслятор (вариант для БЭСМ-6). – Мн.: Ин-т техн. кибернетики АН БССР, 1982. – 105 с.

4. Торопов Н.Р. Алгоритмический язык ЛЯПАС // Логическое проектирование: сб. науч. тр. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2001. – Вып. 6. – С. 6-25.

5. Бибило П.Н. Кремниевая компиляция заказных СБИС. – Мн.: Ин-т техн. кибернетики АН Беларуси, 1996. – 268 с.

6. Закревский А.Д., Василенок В.К. Описание алгоритмов логического управления. – Мн.: Ин-т техн. кибернетики АН БССР, 1985. – 67 с.

7. Закревский А.Д. Параллельные алгоритмы логического управления. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1999. – 202 с.

8. Система ЛОГИКА-М синтеза управляющих устройств в базисе ПЛМ и микропроцессоров / А.Д. Закревский, В.К. Василенок, Л.Д. Черемисинова и др. // Управляющие системы и машины. – 1987. – № 3. – С. 31-36.

9. Система логического синтеза устройств управления на базе программируемых контроллеров / Л.Д. Черемисинова, В.К. Василенок, Е.В. Желудко и др. – Мн.: Ин-т техн. кибернетики АН БССР, 1988. – 100 с.

10. Черемисинов Д.И. Система синтеза программ для микропроцессоров. – Мн.: Ин-т техн. кибернетики АН БССР, 1989. – 84 с.

11. Cheremisinov D.I. Schematic netlist converter // Proc. of the Fourth Int. Conf. on Computer-Aided Design of Discrete Devices (CAD DD'2001), Minsk, Belarus, November 14-16, 2001. – Minsk: Institute of Engineering Cybernetics, NAS of Belarus, 2001. – V. 1. – P. 121-125/

12. Бибило П.Н. Основы языка VHDL. – М.: Солон-Р, 2000. – 200 с.

13. Бибило П.Н. Синтез логических схем с использованием языка VHDL. – М.: Солон-Р, 2002. – 384 с.

14. Черемисинова Л.Д. Описание и верификация параллельных алгоритмов логического управления // Логическое проектирование: сб. науч. тр. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2001. – Вып. 6. – С. 69-87.

15. Тропашко В.В. Доказательство гипотезы о полной редуцируемости α-сетей // Проектирование систем логического управления: сб. науч. тр. – Мн.: Ин-т техн. кибернетики АН БССР, 1986. – С. 13-21.

16. Черемисинова Л.Д. Реализация параллельных алгоритмов логического управления. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2002. – 246 с.

17. Ковалев А.В., Поттосин Ю.В. К декомпозиции параллельных алгоритмов логического управления // Автоматика и вычислительная техника. – 1988. – № 1. – С. 8-13.

18. Закревский А.Д. Логический синтез каскадных схем. – М.: Наука, 1981. – 416 с.

19. Черемисинов Д.И. Инструментальный комплекс для разработки программного обеспечения микропроцессоров. – Мн.: Ин-т техн. кибернетики АН БССР, 1988. – 58 с.

20. Уткин А.А. Анализ логических сетей и техника булевых вычислений. – Мн.: Наука и техника, 1979. – 152 с.

21. Уткин А.А. Экспериментальное исследование алгоритмов «ВЫПОЛНИМОСТЬ» // Автоматика и вычислительная техника. – 1990. – № 6. – С. 66-74.

22. Goldberg E., Novikov Ya. BerkMin: A fast and robust SAT-solver // Proceedings of Design, Automation and Test in Europe Conference. – 2002. – P. 142-149.

23. Закревский А.Д. Комбинаторика логического проектирования // Автоматика и вычислительная техника. – 1990. – № 2. – С. 68-79.

24. Закревский А.Д. Комбинаторные задачи над логическими матрицами в логическом проектировании и искусственном интеллекте // Успехи современной радиоэлектроники. – 1998. – № 2. – С. 59-67.

25. Поттосин Ю.В. Задачи теории графов в логическом проектировании // Логическое проектирование: сб. науч. тр. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2001. – Вып. 6. – С. 106-130.

26. Синтез асинхронных автоматов на ЭВМ / под ред. А.Д.Закревского. – Мн.: Наука и техника, 1975. – 184 с.

27. Погарцев А.Г. Новые алгоритмы совместной минимизации булевых функций // Автоматика и вычислительная техника. – 1980. – № 1. – С. 34-41.

28. Бибило П.Н., Енин С.В. Синтез комбинационных схем методами функциональной декомпозиции. – Мн.: Наука и техника, 1987. – 189 с.

29. Система логического проектирования дискретных устройств на программируемых матричных БИС / П.Н. Бибило, Е.И. Гольдберг, И.П. Каркоцкая и др. – Мн.: Ин-т техн. кибернетики АН БССР, 1987. – 82 с.

30. Комплекс программ синтеза комбинационных схем в базисе ПЛМ и МЛВ / А.А. Дудкин, Ю.В. Поттосин, А.А. Синичка и др. – Мн.: Ин-т техн. кибернетики АН БССР, 1988. – 67 с.

31. Система логического проектирования заказных СБИС на базе регулярных матричных структур / А.Д. Закревский, П.Н. Бибило, С.Н. Кардаш и др. // Микроэлектроника. – 1995. – № 5. – С. 349-354.

32. Торопов Н.Р. Минимизация систем булевых функций в классе ДНФ // Логическое проектирование. – Мн.: Ин-т техн. кибернетики НАН Беларуси. – 1999. – Вып. 4. – С. 4-19.

33. Система «Custom Logic» автоматизированного проектирования управляющей логики заказных цифровых СБИС / П.Н. Бибило, И.В. Василькова, С.Н. Кардаш и др. // Микроэлектроника. – 2004. – Т. 32. – № 5.

34. Бутов А.А. Реализация систем не полностью определенных булевых функций в базисе схем малого и среднего уровней интеграции // Управляющие системы и машины. – 1979. – № 6. – С. 97-103.

35. Шнейдер А.А., Кардаш С.Н. Алгоритмы синтеза одноярусных комбинационных схем из ПЛМ // Автоматика и вычислительная техника. – 1987. – № 5. – С. 62-67.

36. Поттосин Ю.В. Декомпозиция асинхронных автоматов // Автоматика и вычислительная техника. – 1978. – № 4. – С. 1 – 7.

37. Бибило П.Н. Синтез комбинационных ПЛМ-структур для СБИС. – Мн.: Наука и техника, 1992. – 232 с.

38. Черемисинова Л.Д. Минимизация площади матрицы Вайнбергера, реализующей систему ДНФ // Управляющие системы и машины. – 1999. – № 2. – С. 39-46.

39. Cheremisinova L.D. Simple folding of array-based VLSI structures // 6th Int. Workchop on Boolean problems, Freiberg (Sachsen), Sept. 19-20, 2004. – 2004. – P. 245-250.

40. Система логического проектирования «Синтез БМК» / П.Н. Бибило, Н.А. Кириенко, Л.В. Красильникова и др. // Управляющие системы и машины. – 2001. – № 3. – С. 28-35.

41. Логинова И.П. «Моделирование отжига» и минимизация площади ПЛМ на основе многократной свертки // Логическое проектирование: сб. науч. тр. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1997. – Вып. 2. – С. 25-41.

42. Алгоритмы решения логико-комбинаторных задач: сб. науч. тр. – Мн.: Ин-т техн. кибернетики АН БССР, 1979. – Вып. 5.

43. Енин С.В., Филатченков А.И. Диалоговая система программ для анализа и моделирования дискретных устройств с неисправностями // Электронное моделирование. – 1986. – Т. 8. – № 3. – С. 25-29.

44. Торопов Н.Р. Исследовательская САПР дискретных управляющих устройств. – Мн.: Ин-т техн. кибернетики АН Беларуси, 1993. – 60 с.

45. Бибило П.Н. Логическое проектирование дискретных устройств с применением продукционно-фреймовой модели представления знаний // Известия РАН. Теория и системы управления. – 1999.– № 2. – I. – С. 130-138; 1999. – № 4. – II. – С. 124-132.

46. Экспертная система логического распознавания ЭКСИЛОР / А.Д. Закревский, А.А. Уткин, В.И. Романов и др. – Мн.: Ин-т техн. кибернетики АН Беларуси, 1993. – 55 с.

Поступила 10.11.2004

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: bibilo@newman.bas-net.by
cld@newman.bas-net.by

P.N. Bibilo, L.D. Cheremisinova

LOGICAL DESIGN AUTOMATION

Theoretical results are reviewed that have been obtained in the Laboratory of Logical Design of the United Institute of Informatics Problems in the field of discrete devices and large scale integrated circuit logic design. Software packages and logic design automation systems developed in the Laboratory are also described.