АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 1 1991

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ

УДК 681.32.06 : 519.87

З. А. ЛИВШИЦ, К. К. СМИРНОВ
(Новосибирск)

LOGIC — ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ГЕНЕРАЦИИ ТОПОЛОГИИ ПЛМ

Введение. Использование программируемых логических матриц (ПЛМ) является в настоящее время наиболее распространенным методом реализации управляющей логики больших интегральных схем. Это обусловлено рядом хорошо известных преимуществ такого рода регулярных структур: упрощением планирования и модификации проектов, основанных на применении ПЛМ, легкой тестируемостью и самое главное возможностью полностью автоматизировать процесс синтеза управляющей логики, включая этап генерации топологии.

Современные требования к средствам автоматизации проектирования ПЛМ не ограничиваются тем, чтобы возложить на компьютер отображение спецификации логической задачи в геометрию соответствующей матрицы: речь идет об оптимизации синтезируемых структур, которая позволила бы нейтрализовать присущий ПЛМ недостаток — относительно низкую плотность транзисторов и связанные с этим снижение коэффициента использования площади кристалла и ухудшение характеристик производительности.

Убедительные количественные подтверждения эффективности применения «ПЛМ-ориентированной» методологии проектирования в сочетании с развитыми средствами автоматизированного синтеза ПЛМ представлены в [1], авторы которой провели перепроектирование ряда известных серийных микропроцессоров, максимально используя ПЛМ. При этом были произведены сравнительные оценки таких параметров, как площадь кристалла, производительность, фактор регулярности (отношение площади, занимаемой программируемыми логическими матрицами, к общей площади кристалла). Проведенное в [1] исследование показало, что ПЛМ обеспечивает удачный компромисс между качеством, присущим «ручному» проектированию, и временны́ми затратами на разработку кристалла.

Следует, однако, отметить, что оптимизационные алгоритмы, эффективность которых критична для систем автоматизации проектирования СБИС, основаны, как правило, на эвристических методах и являются весьма времяемкими. Поэтому к вопросам их совершенствования в течение последнего десятилетия проявляется большой интерес, и им посвящено значительное количество работ, среди которых можно выделить [2—5].

В данной статье рассматривается разработанный в Институте автоматики и электрометрии СО АН СССР программный комплекс LOGIC, предназначенный для генерации топологии ПЛМ, ядром которого является программа многозначной логической минимизации, обладающая достаточно высокими (по сравнению с известными программами аналогичного назначения) временны́ми характеристиками. Комплекс реализо-

ван на ЭВМ типа «Электроника 82» в операционной среде VAX/VMS.

1. ПЛМ и минимальные покрытия. Топология традиционных ПЛМ представляет собой объединение двух частей, называемых AND- и OR-плоскостями. AND-плоскость формируется расположенными параллельно линиями входных сигналов (прямых и инверсных) и идущими в ортогональном направлении линиями термов. «Программирование» AND-плоскости осуществляется помещением активных транзисторов в местах пересечения определенных линий термов с определенными линиями сигналов, в результате чего на линиях термов вырабатываются значения заданных элементарных конъюнкций входных сигналов и их отрицаний. OR-плоскость формируется пересечением продолжений линий термов AND-плоскости и линий выходных сигналов, что позволяет получать на выходной линии значение сигнала, представляющего дизъюнкцию выбранных термов.

Таким образом, ПЛМ обеспечивает возможность непосредственной реализации любой системы булевых функций:

$$y_i = f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, m, \tag{1}$$

представленных в виде дизъюнктивных нормальных форм.

На ПЛМ-структуры могут отображаться как комбинационные схемы, так и конечные автоматы. В последнем случае часть входных и выходных переменных используется для кодирования состояния автомата (код состояния, выработанный на выходе ПЛМ на очередном такте, с некоторой задержкой подается на вход ПЛМ).

Площадь кристалла, занимаемая ПЛМ, реализующей систему логических функций (1), примерно пропорциональна величине $(2n + m)t$, где t — число линий термов, равное общему количеству различных дизъюнктивных членов в представлении функций системы (1). Таким образом, здесь имеет место весьма редкая ситуация, когда оптимизация геометрических (и электрических) характеристик схемы прямо сводится к минимизации по чисто логическому критерию.

Минимизация совокупного числа термов (нахождение минимального покрытия) *заданной* системы логических функций является не единственным способом улучшения параметров ПЛМ: имеются и возможности, связанные с *варьированием* самой реализуемой *системы* при сохранении функционирования схемы. К таким методам можно отнести, например, «назначение фазы» выходов комбинационной схемы (т. е. выбор для реализации «внутри» ПЛМ функции f_i или $\bar{f}_i$, осуществляемый для всех $i = 1, \ldots, m$), оптимизацию кодирования состояний конечного автомата и т. п. Однако и во всех этих случаях, и при использовании методов топологической оптимизации (типа «свертывания» ПЛМ) задача о минимальном покрытии является базовой в том смысле, что все оптимизационные алгоритмы используют либо результаты ее решения, либо процедуры его нахождения для некоторых систем функций.

В следующем разделе статьи обсуждаются соображения, положенные в основу разработанной программы логической минимизации, а также приводятся некоторые результаты экспериментов, характеризующие ее производительность.

В состав первой очереди комплекса для синтеза оптимизированных ПЛМ входят еще две программные компоненты (следует отметить, что все эти программы вполне автономны и могут использоваться независимо друг от друга): программа преобразования системы логических функций в формат таблицы истинности (т. е., по существу, выбор некоторых дизъюнктивных нормальных форм для функций системы) и собственно программа генерации описания топологии в формате CIF. Первая из указанных программ просто обеспечивает достаточно удобный интерфейс с пользователем, вторая — более подробно рассмотрена в разд. 3.

2. Алгоритмы логической минимизации. 2.1. Как отмечалось выше, задача минимизации площади ПЛМ, реализующей систему частично определенных логических функций, состоит в нахождении таких разложений

этих функций в дизъюнктивные нормальные формы, которые совместно имеют минимальное число различных термов (конъюнкций).

Выясняется, однако, что в ряде случаев полезно рассматривать более общую задачу, связанную с минимизацией булевых функций, зависящих от переменных, принимающих произвольное конечное множество значений. Такие ситуации возникают, например, при оптимизации задания состояний конечных автоматов, синтезе ПЛМ с предварительным декодированием и т. п. Поэтому разрабатываемые в последние годы алгоритмы минимизации ориентированы на работу с функциями многозначных переменных. Заметим попутно, что при этом минимизация системы функций $y_i = f_i(x_1, \ldots, x_n)$, $i = 1, \ldots, m$, может быть сведена к минимизации одной функции $F(x_1, \ldots, x_n, z)$, определенной следующим образом:

$$F(x_1, \ldots, x_n, z) = f_{z+1}(x_1, \ldots, x_n),$$

где z — целочисленная переменная, принимающая значения 0, 1, ..., $m-1$.

Задачу поиска минимальной формы для функции от многозначных переменных можно сформулировать так.

Пусть $Y_1, Y_2, \ldots, Y_s$ — конечные множества; $U = Y_1 \times Y_2 \times \ldots \times Y_s$ — их прямое произведение. Точки из U (по аналогии с «двоичным случаем») будем называть минтермами, а множества вида $\varsigma_1 \times \ldots \times \varsigma_s$, где $\varsigma_i \subset Y_i$ — произвольное непустое подмножество Y_i,— кубами или термами.

Пусть $F : U \to B = \{0, 1\}$ — частично определенная функция. Обозначим через $H^0 = F^{-1}(0)$ и $H^1 = F^{-1}(1)$ множества точек из U, на которых функция F принимает значения «0» и «1» соответственно. Требуется найти минимальное покрытие кубами из U множества H^1, т. е. минимальную по количеству совокупность кубов вида $\{\varsigma_1^j \times \ldots \times \varsigma_s^j\}$, $j = 1, \ldots, N$, такую, что

$$H^1 \subset \bigcup_{j=1}^{N} (\varsigma_1^j \times \ldots \times \varsigma_s^j) \text{ и } H^0 \cap (\varsigma_1^j \times \ldots \varsigma_s^j) = 0$$

для всех j. (Отметим, что если все $Y_i = \{0, 1\}$, то приведенная выше формулировка означает просто нахождение для булевой функции дизъюнктивной нормальной формы с минимальным числом членов.)

Алгоритм определения точного решения задачи минимизации функции от многозначных переменных может быть получен обобщением хорошо известной процедуры для функций булевых переменных [6, 7]: вначале находятся все простые импликанты исходной функции (т. е. максимальные кубы множества $U \backslash H^0$) и строится матрица формальных импликаций, а затем отыскивается минимальное строчное покрытие этой матрицы, которое определяет все кубы одного из минимальных покрытий множества H^1.

Вычислительная сложность этого алгоритма является, однако, чрезвычайно высокой: первый его этап — нахождение всех простых имиликант — представляет собой наименее трудоемкий шаг, и в то же время известен класс функций, зависящих от n булевых переменных, для которых число импликант пропорционально $3n/n$ [8].

По этим причинам алгоритмы нахождения точного минимума применимы лишь для функций от небольшого числа переменных, в то время как сегодняшние практические потребности связаны с синтезом ПЛМ, могущих иметь более 50 входов и 50 выходов.

Вышеизложенное свидетельствует о необходимости применения эвристических алгоритмов минимизации, которые должны обеспечивать достаточно хорошее приближенное решение задачи за приемлемое время.

2.2. В работе [2] предложен подход к нахождению приближенного решения задачи минимизации, основные идеи которого применялись во многих последующих разработках эвристических программ минимизации (в том числе и в рассматриваемой в данной статье). Его суть состоит в использовании итеративной процедуры последовательного улучшения

(т. е. уменьшения количества термов) покрытия до достижения локального минимума функции числа термов.

Каждый шаг итеративного алгоритма, описанного в [2], включает два этапа: процесс расширения и процесс редукции.

В рамках процесса расширения последовательно каждый куб покрытия заменяется содержащим его максимальным в $U \setminus H^0$ кубом (простой импликантой), включающим в себя по возможности наибольшее количество других кубов из покрытия, которые после этого из текущего покрытия удаляются. Этот процесс дает безызбыточное относительно включения покрытие множества H^1, состоящее из простых импликант. В то же время, поскольку каждая простая импликанта является максимальным элементом, что «новое» множество кубов все еще остается покрытием H^1. Процесс редукции приводит к безызбыточному покрытию множества H^1.

Итеративная процедура считается завершенной, если полученное на очередном этапе расширения покрытие уже не может быть улучшено с помощью процессов редукции и расширения. Это покрытие и принимается в качестве приближенного решения задачи.

Наиболее существенным шагом процедуры улучшения покрытия является определение на этапе расширения «наилучшей» простой импликанты, содержащей данный куб покрытия. Основное наблюдение авторов [2] состоит в том, что для этого можно использовать любое покрытие множества H^0. Действительно, пусть $R = \{M^\alpha\}$ — произвольное покрытие H^0, $\varsigma = \varsigma_1 \times \ldots \times \varsigma_s$ — куб, содержащийся в $U \setminus H^0$ и $j \in 1, \ldots, s$. Положим $\tau_j(\varsigma) = \varsigma_1 \times \ldots \times z_j(\varsigma) \times \ldots \times \varsigma_s$, где $z_j(\varsigma) = \bigcup\limits_\alpha \overline{M_j^\alpha}$, а объединение берется по всем кубам покрытия R таким, что $\varsigma_i \cap M_i^\alpha \neq \varnothing$, если $i \neq j$. Тогда $\tau_j(\varsigma) \subset U \setminus H^0$ и содержит ς, и, кроме того, куб $\tilde{\varsigma} = z_1(\varsigma) \times \times z_2(\varsigma) \times \ldots \times z_s(\varsigma)$ является наименьшим кубом, содержащим все импликанты множества $U \setminus H^0$, которые включают куб ς. В частности, если куб $\tau \subset U \setminus H^0$ может быть покрыт некоторым расширением куба ς, то τ покрывается кубом $\tilde{\varsigma}$.

Дальнейшее развитие предложенного в [2] подхода было направлено прежде всего на повышение вычислительной эффективности основных процедур, входящих в алгоритм поиска приближенного решения: нахождения кубического покрытия дополнения к заданному покрытию; проверки на покрываемость некоторого куба данным множеством кубов; определения минимального куба, содержащего дополнение к некоторому множеству кубов, и т. п.

Так, в работе [4] описана эффективная процедура построения кубического покрытия дополнения для логической функции многозначных переменных, дающая, как правило, значительно меньшее число термов в покрытии, чем процедура из [2], основанная на использовании операции #.

В работах [3, 5] показано, что за счет рекурсивного понижения размерности задачи минимизации (в данном случае разложения пространства U на непересекающиеся подпространства вида

$$U_i = Y_1^i \subset \times \ldots \times Y_s^i \subset Y_1 \times \ldots \times Y_s = U, \quad i = 1, \ldots, k)$$

можно получить экономные по вычислительным затратам процедуры как для определения покрытости куба, так и для нахождения его максимальной редукции относительно некоторого множества кубов.

2.3. Программа логической минимизации, входящая в состав комплекса LOGIC, так же, как и рассмотренные выше, базируется на итера-

тивном алгоритме расширения (редукции) с рекурсивным понижением размерности. Основными отличительными ее особенностями, наиболее существенно влияющими на параметры производительности, являются следующие:

использование стратегии понижения размерности, основанной на выборе подходящего куба, по которому осуществляется разложение покрытия;

специальная схема совмещения процессов редукции и расширения, позволяющая использовать для построения «начального приближения» на шаге редукции информацию, полученную на шаге расширения.

Поясним вкратце эти механизмы. Следует отметить, прежде всего, что выбор конкретного способа разложения универсального пространства на подмножества, на которые ограничиваются кубы исходного покрытия при понижении размерности, заметно влияет на эффективность алгоритма. В методике понижения размерности, описанной в [3, 5], использовалось разложение по определенным образом выбирающейся переменной. Однако эксперименты с функциями многозначных переменных показали, что предпочтительным является подход, связанный с разложением покрытия по некоторому кубу. Оно всегда оказывается возможным, так как если

$$\varsigma_1 \times \ldots \times \varsigma_s \subset Y_1 \times \ldots \times Y_s = U$$

— некоторый куб, то, как легко убедиться, совокупность кубов

$$\varsigma_1 \times \ldots \times \varsigma_s \text{ и } \varsigma_1 \times \ldots \times \varsigma_{i-1} \times \bar{\varsigma}_i \times Y_{i+1} \times \ldots \times Y_s, \quad i = 1, \ldots, s,$$

является разложением U на непересекающиеся подпространства.

Далее, существенное внимание уделено сокращению временных затрат на осуществление процесса редукции покрытия, полученного на шаге распространения — наиболее трудоемкой части итеративной процедуры (особенно при «обширности» множества точек, где значение функции не определено).

Ясно, например, что если для редукции некоторого куба имеется хорошее начальное приближение (т. е. меньший куб, которым можно заменить исходный с сохранением свойства быть покрытием), то естественно, может быть сокращено и время, требуемое для нахождения минимальной редукции исходного куба.

Основная идея применяемой в данной программе схемы состоит в том, что при реализации очередного этапа алгоритма могут быть с успехом использованы не только «окончательные», но и «промежуточные» результаты, полученные на предыдущих этапах.

Так, в частности, если при расширении куба ς, входящего в некоторое покрытие, получается куб, включающий кубы $\varsigma_1, \ldots, \varsigma_r$, этого покрытия, то минимальный куб, содержащий $\varsigma, \varsigma_1, \ldots, \varsigma_r$, будет импликантой. Этот куб может быть хорошим начальным приближением на шаге редукции (причем следует отметить, что это приближение возможно будет еще «улучшено» при продолжении процесса расширения кубов из текущего покрытия).

Кроме того, сохранение информации о наличии такого «меньшего» расширяющего куба повышает вероятность его покрытия при расширении следующих кубов и тем самым уменьшения общего числа кубов в покрытии.

Рассматриваемая программа обладает также некоторыми опциональными (осуществляемыми по запросу пользователя) возможностями предобработки и постобработки. К таковым относится, например, предварительное выделение множества всех существенных импликант, т. е. импликант, которые обязаны входить в любое простое покрытие данного пространства. Поскольку для большинства практических задач количество существенных импликант составляет 30—40 % от общего числа термов в минимальном покрытии, то такая предобработка может заметно сократить общее время поиска минимального покрытия (удаление существен-

Наименование схемы	Число входов	Число выходов	Исходное число термов	Результирующее число термов		Временные затраты (CPU, с)	
				LOGIC	ESPRESSO	LOGIC	ESPRESSO
dist	8	5	165	120	121	110,2	129,9
xor 10	10	2	512	512	512	390,7	403,0
mult 2	4	4	12	7	7	1,5	2,2
mult 4	8	8	225	128	133	168,5	305,7
add 2	4	3	17	11	11	0,8	2,5
add 4	8	5	124	75	75	36,3	83,0

ных импликант понижает размерность задачи, уменьшая количество кубов, участвующих в итеративном процессе).

Другая возможность — это редукция результирующего покрытия относительно некоторой отмеченной переменной. Такая редукция может быть полезна для удаления избыточных транзисторов в OR-плоскости.

2.4. В качестве языка реализации программы логической минимизации был выбран макроассемблер. Это обстоятельство позволило повысить эффективность программы (как в отношении временны́х характеристик, так и по объемам используемой памяти).

Качество работы программы исследовалось при решении задач минимизации системы логических функций, входные спецификации которых содержали от нескольких десятков до нескольких сотен термов. Опыт ее эксплуатации показал, что программа обеспечивает весьма хорошее приближение к точному решению задачи нахождения минимального покрытия (в действительности, как правило, программа приводит к точному решению). Результаты сравнения минимизатора LOGIC с известной программой аналогичного назначения ESPRESSO-IIC (характеристики последней почерпнуты нами из [9]) сведены в таблицу. Временны́е параметры таблицы соответствуют случаю исполнения программы на ЭВМ типа «Электроника 82».

3. Генерация топологии ПЛМ. Основное требование, предъявлявшееся к разработке программы генерации топологии ПЛМ, состояло в обеспечении ее технологической независимости или, говоря более точно, максимальной простоты «настройки» программы на конкретные технологию и стиль проектирования.

Подход, реализованный в LOGIC, основан на методе библиотечных элементов и идейно близок к рассмотренному в [10].

Анализ структуры ряда образцов ПЛМ, применяемых в массовых кристаллах, показывает, что при незначительных ограничениях на строение программируемой логической матрицы можно определить функциональные группы ячеек (библиотечных элементов) и общие для всех типов матриц правила, позволяющие синтезировать топологические маски конкретной ПЛМ, имея ее описание в виде таблицы истинности и полный набор библиотечных элементов. Поскольку при разработке ПЛМ обычно используются специальные приемы экономии площади (такие, как, например, совмещение контактов), это необходимо учитывать при построении алгоритма генерации масок. В данной программе предусматривается включение для этих целей в библиотеку стандартных элементов особой группы ячеек. Эти ячейки не являются носителями геометрических структур, размещаемых в плоскостях ПЛМ, а лишь определяют необходимые перекрытия между определенными фрагментами генерируемой топологии. В частности, отсутствие некоторой ячейки этой группы в момент генерации означает лишь отсутствие соответствующего ей перекрытия.

Преимущество данного подхода состоит в том, что имеется возможность формировать библиотеку элементов, «вырезая» нужные фрагменты непосредственно из какого-либо подходящего образца топологии ПЛМ средствами графического редактора. При этом требуется знать только

имена нужных ячеек и последовательность, в которой программа «складывает» ячейки в соответствии со строкой таблицы истинности реализуемой функции.

Именно таким способом была создана эксплуатируемая в ИАиЭ СО АН СССР библиотека стандартных элементов, содержащая несколько полных наборов базовых ячеек. При желании пользователь может дополнять ее собственными наборами, используя указанный выше способ.

Заключение. В целом можно констатировать, что LOGIC обеспечивает достаточно эффективные (в сравнении с известными разработками) средства логической минимизации систем функций и генерации топологии соответствующих логических матриц.

В настоящее время ведутся работы по развитию комплекса, предусматривающие включение в его состав программных средств для оптимального кодирования состояний конечных автоматов, реализуемых с помощью ПЛМ, оптимизации назначения фазы выходных сигналов, сокращения площади, занимаемой ПЛМ, за счет использования механизмов свертывания ПЛМ и т. п.

СПИСОК ЛИТЕРАТУРЫ

1. **Obrebska M., Chuquillanqui S., Dcrantonian И.** PLA and custom design // Advanced in CAD for VLSI.— V. 6. Design Methodologies.— N-H., 1986.
2. **Hong S. J., Cain R. G., Ostapko D. L.** MINI: a heuristic approach for logic minimization // IBM J. of Res. and Dev.— 1974.— 18, N 5.
3. **Brayton R. K., Hachtel G. D., McMullen C. T., Sangiovanni-Vincentelli A. L.** Logic Minimization Algorithms for VLSI synthesis.— N. Y.: Kluwer Academic Publishers, 1984.
4. **Sasao T.** An algorithm to derive the complement of a binary function with multiple-valued inputs // IEEE Trans. on Computers.— 1985.— **C-34**, N 2.
5. **Rudell R. L., Sangiovanni-Vincentelli A. L.** Multiple-valued minimization for optimization // IEEE Trans. on CAD.— 1987.— **CAD-6**, N 9.
6. **Quine W. L.** A way to simplify truth functions // Amer. Math. Mon.— 1955.— 62, N 6.
7. **McCluskey E. S.** Minimization of Boolean functions // Bell Syst. Techn. J.— 1956.— 35, N 4.
8. **Miller R. F.** Switching Theory. V. 1. Combinatorial Circuits.— N. Y., 1965.
9. **Dagenais M. R., Agarwali V. K., Rumin N. E.** The McBoole logic minimizer // Proc. 22nd Design Automation. Conf.— Las Vegas, 1985.
10. **Mayo R. N., Ousterhout S. K.** Pictures with parentheses: combining graphics and procedures in a VLSI layout tool // Proc. 20th Design Automation Conf.— Maiami Beach, 1983.

Поступила в редакцию 27 сентября 1990 г.

УДК 621.3.049.77 : 681.32.06

А. М. ИВАНОВ, З. А. ЛИВШИЦ, А. В. НИЧУЕВ, А. Г. РЯБЧЕНКО, Д. Г. ТИТОВ, С. А. ФРОЛОВ

(Новосибирск)

РАЗРАБОТКА ЗАКАЗНОЙ БИС СПЕЦИАЛИЗИРОВАННОГО АРИФМЕТИЧЕСКОГО УСТРОЙСТВА

1. Преимущества, обеспечиваемые использованием специализированных интегральных схем, разработка которых ориентирована на конкретные системные приложения, по сравнению с построением систем на базе стандартных серийных схем, хорошо известны. К ним в первую очередь относятся возможности повышения производительности и надежности, уменьшение габаритов изделий, потребляемой ими мощности, а также (начиная с определенных объемов производства) снижение стоимости. Эти обстоятельства стимулировали бурное развитие работ в области создания специализированных ИС: в настоящее время они составляют около половины номенклатуры выпускаемых в мире схем. В свою очередь, это привело к тому, что проектирование интегральных схем превратилось в