3. *Чернухин Ю.В.* Нейропроцессорные сети. – Таганрог: Изд-во ТРТУ, 1999. – 439 с.
4. *Чернухин Ю.В., Писаренко С.Н.* Нейросетевая экстраполяция в системах управления интеллектуальных мобильных роботов. Сборник докладов Юбилейной Международной конференции по нейрокибернетике. – Ростов-на-Дону: Изд-во ООО «ЦВВР», 2002, Т2. – С. 147-151.

Б.К. Лебедев, С.А. Степаненко

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ РАЗМЕЩЕНИЯ, УПРАВЛЯЕМЫЙ ВРЕМЕННЫМИ ОГРАНИЧЕНИЯМИ*

Введение. Интенсивное исследование проблемы размещения элементов СБИС, управляемого временными ограничениями, проводилось в последние 20 лет, и продолжается в настоящее время. Производительность схемы определяется задержкой наиболее длинного пути, но временные ограничения являются предельно сложными. Число путей прохождения сигнала растет экспоненциально с ростом размера схемы. Даже схема умеренного размера может иметь огромное количество путей. Более того, на различные пути могут накладываться разные ограничения.

Существующие алгоритмы размещения, управляемого временными ограничениями, можно разделить на 2 категории: основанные на оценке путей распространения сигналов (path-based) и основанные на оценке цепей схемы (net-based).

Path-based алгоритмы [1-3] непосредственно пытаются минимизировать задержку наиболее длинного пути. Популярным подходом является минимизация суммы длин некоторого множества критических путей. Этот набор критических путей либо предварительно определяется методом статического временного анализа, либо может динамически изменяться от итерации к итерации. Большинство алгоритмов этого класса основано на методах математического программирования. Преимуществом алгоритмов типа path-based является достаточно точный учет временных ограничений в процессе оптимизации. Тем не менее, их недостаток заключается в том, что они требуют существенное количество вычислительных ресурсов из-за экспоненциального числа путей, которые необходимо одновременно минимизировать. Кроме того, в некоторых типах алгоритмов размещения, например, в алгоритмах, основанных на нисходящем разбиении, очень трудно, а иногда невозможно, поддерживать точный глобальный учет временных ограничений.

Алгоритмы **net-based** [4-7] в отличие от **path-based** алгоритмов непосредственно не накладывают ограничения на пути прохождения сигнала. Вместо этого, временные ограничения или требования на пути преобразовываются в ограничения на длину цепей или веса цепей. Основная идея заключается в том, чтобы более критическим цепям присвоить больший вес. Эта информация затем используется алгоритмом минимизации взвешенной длины соединений, с целью получения размещения с лучшей временной задержкой. Этот полученный вариант размещения рассматривается статическим анализатором, после чего получается новый набор временной информации, которая используется на следующей итерации. Обычно этот процесс повторяется на нескольких итерациях, до тех пор, пока наблюдается улучшение или пока не будет выполнено заданное число итераций. Методики, основанные на вычислении весов цепей, обладают некоторыми привлекательными свойствами: сравнительно низкая временная сложность, большая гибкость (могут быть интегрированы в существующие алгоритмы, минимизирующие длину про-

* Работа выполнена при финансовой поддержке программы развития научного потенциала высшей школы РНП.2.1.2.2238

водников), простота реализации. В условиях постоянного усложнения временных ограничений современных схем, эти преимущества придают методикам, основанным на вычислении весов цепей все большую привлекательность.

В данной работе предлагается генетический алгоритм решения задачи размещения, управляемого временными ограничениями.

Моделирование временных ограничений. В алгоритмах размещения, управляемых временными ограничениями обычно применяется методика статического временного анализа для вычисления задержки наиболее длинного пути схемы. Результаты этого анализа далее используются алгоритмом, минимизирующим задержку наиболее длинного пути схемы.

В статическом временном анализе пути распространения сигналов в схеме обычно моделируются направленным ациклическим графом $G(V,E)$: вершинами графа является множество входных и выходных контактов элементов схемы, а ребрами являются связи между элементами. Каждому ребру присваивается величина временной задержки. Путь распространения сигнала начинается с первичного входа или выхода элемента памяти и заканчивается в первичном выходе или входе элемента памяти. Через P_I и P_O обозначаются множества входных и завершающих выходных контактов пути, соответственно.

Задержка наиболее длинного пути может быть определена путем итеративного вычисления времени прибытия сигнала (*arrival time*) по следующей формуле:

$$ARR(t) = \begin{cases} 0 \, if \; t \in P_I; \\ \max_{(s,t)\in E}\{ARR(s) + d(s,t)\} \, otherwise; \end{cases}$$

где $d(s,t)$ – задержка ребра (s,t).

Задержка наиболее длинного пути определяется по формуле:

$$T = \max_{t\in P_O} ARR(t).$$

Подобным образом можно вычислить требуемое время прибытия сигнала (*required arrival time*) для каждого контакта, по следующей формуле:

$$REQ(s) = \begin{cases} T \, if \; s \in P_O; \\ \min_{(s,t)\in E}\{REQ(t) - d(s,t)\} \, otherwise. \end{cases}$$

Теперь для каждого ребра можно определить величину резерва задержки, которая показывает, как дополнительная величина задержки может быть добавлена к ребру, без увеличения задержки наиболее длинного пути всей схемы. Резерв задержки для заданного ребра (s,t) определяется по формуле:

$$slack(s,t) = REQ(t) - ARR(s) - d(s,t).$$

Задержка какого-либо пути есть сумма задержек ребер, составляющих путь:

$$slack(\pi) = \sum_{e\in\pi} d(e).$$

Аналогично резерв задержки пути определяется как максимальная величина задержки, которая может быть добавлена к пути, чтобы он стал критическим. Резерв задержки пути определяется по формуле:

$$slack(\pi) = T - d(\pi).$$

Хороший алгоритм, вычисляющий веса цепей должен учитывать эффект разделения путей (path sharing). Интуитивно понятно, что если два критических пути содержат общий сегмент, ребрам этого общего сегмента должен быть присвоен

больший вес. Для учета этого эффекта применяется распространенный подход, основанный на пересчете путей (path counting).

В работе [8] был предложен алгоритм PATH, выполняющий назначение весов цепей на основе перебора путей. Алгоритм PATH мотивируется следующей проблемой: имея временной граф, после выполнения анализа путей прохождения сигналов, для ребер принадлежащих некоторому критическому пути, необходимо вычислить общее число критических путей, проходящих через каждое ребро.

Эта проблема может быть легко решена следующим простым алгоритмом. Для каждого контакта *p* временного графа определяются две переменные:

- Прямой путь F(p) – число различных критических путей, начинающихся от контактов элементов P_I и заканчивающихся в *p*.
- Обратный путь *B(p)* – число различных критических путей, начинающихся от контактов элементов P_O и заканчивающихся в *p*, если изменить на противоположное направление все пути распространения сигналов.

Число различных критических путей проходящих через ребро (s,t) определяется по формуле: $GP(s,t) = F(s) \times B(t)$.

Проблема минимизации временных задержек в общем виде формулируется следующим образом:

$$\min F(x), \sum_{e \in \pi} d(e) \le T, \forall \pi \in P, \quad (1)$$

где *F(x)* – некоторая целевая функция задачи размещения, *P* – множество всех путей в схеме.

Можно преобразовать данную постановку в постановку без ограничений:

$$\min F(x) + \sum_{\pi \in P} D(slack(\pi), T) * d(\pi). \quad (2)$$

Веса для отдельных ребер $e(s,t)$ определяются по формуле:

$$PW_D(e) = \sum_{\pi \notin e} D(slack(\pi), T), \quad (3)$$

где $D(x,y)$ – заданная функция скидок (discount function). Функция скидок $D(x,y)$ называется так потому, что это монотонно убывающая функция от *x*, поскольку мы хотим присвоить меньший вес (больше скидок) для путей с большей величиной резерва задержки.

Алгоритм. Предлагаемый алгоритм размещения управляемого временными ограничениями, основан на методе генетического поиска. Генетические алгоритмы – это поисковые алгоритмы, основанные на механизмах натуральной селекции и натуральной генетики. Они реализуют "выживание сильнейших" среди рассмотренных структур, формируя и изменяя поисковый алгоритм на основе моделирования эволюции поиска. Генетические алгоритмы с успехом применялись при решении различных оптимизационных задач, в том числе задач автоматизированного проектирования СБИС [11], таких как разбиение, размещение, планирование, трассировка. Одним из достоинств генетических алгоритмов является возможность избегания локальных оптимумов.

Генетический алгоритм состоит из следующих основных частей: кодирование решения в виде хромосомы, набор генетических операторов (кроссинговер, мутация, селекция, отбор), целевая функция.

Кодирование решения. Решение задачи размещения представляется в виде негомологичной хромосомы. В нашем подходе представление плана топологии

выполняется с помощью Одиночной последовательности (Single-Sequence или SS) [10]. Одиночная последовательность представляет собой перестановку целых чисел $1,2,3,...,n$, каждое число соответствует размещаемому на плоскости элементу без перекрытия с ранее размещенными элементами. SS представляет собой систему ABLR-отношений (above (сверху), below (снизу), left-of (слева от), right-of (справа от)) между каждой парой чисел, которые выражаются следующим правилом: Если (x, y) в SS расположены в таком порядке и $x < y$, тогда элемент *x* расположен слева от элемента *y*. Если (x, y) в SS расположены в таком порядке и $x > y$, тогда элемент *x* расположен ниже элемента *y*. Одиночная последовательность – один из наиболее простых способов представления системы отношений ABLR. Его можно рассматривать как нормализованную последовательность пар (Sequence-Pair или SP) [9]. Для представления решения с помощью SS требуется в два раза меньше памяти по сравнению с представлением с помощью SP. SS требуют пересчета после каждого шага. Поскольку не разработаны алгоритмы инкрементной переоценки SS, довольно большое время тратится на оценивание одиночных последовательностей. Некоторым недостатком данного представления является негомологичность хромосом, что требует применения специальных генетических операторов, таких как, например, циклический, частично-соответствующий и упорядочивающий операторы кроссинговера, а также мутация обмена, инверсии.

Целевая функция. В качестве целевой функции используется двухкритериальная целевая функция, минимизирующая длину проводников, и временные задержки. Целевая функция выражается формулой (2).

Структура генетического поиска такова: создание начальной популяции, оценка хромосом, выбор пар хромосом и выполнение операторов кроссинговера, выбор хромосом и применение операторов мутации, вероятностный отбор наилучших хромосом. После выполнения заданного числа операций выбор и декодирование наилучшего решения, и конец работы алгоритма.

Заключение. Алгоритмы решения задачи крупномасштабного размещения играют жизненно важную роль в современных нанометровых стилях проектирования, в которых производительность полученных схем в значительной степени зависит от межсоединений проводников.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *M. Jackson and E. S. Kuh*. Performance-driven placement of cell based IC's. In Proc. Design Automation Conf, pp. 370-375, 1989.
2. *A. Srinivasan, K. Chaudhary and E. S. Kuh*. RITUAL: A performance driven placement for small-cell ICs. In Proc. Int. Conf. on Computer Aided Design, pp. 48-51, 1991.
3. *Hamada, C. K. Cheng and P. M. Chau*. Prime: a timing-driven placement tool using a piecewise linear resistive network approach. In Proc. Design Automation Conf, pp. 531-536, 1993.
4. *A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak and M. Wiesel*. Chip layout optimization using critical path weighting. In Proc. Design Automation Conf, pp. 133-136, 1984.
5. *R. Nair, C. L. Berman, P. Hauge and E. J. Yoffa*. Generation of performance constraints for layout. IEEE Trans. on Computer-Aided Design, vol. 8, no. 8, pp. 860-874, 1989.
6. *R. S. Tsay and J. Koehl*. An analytic net weighting approach for performance optimization in circuit placement. In Proc. Design Automation Conf, pp. 620-625, 1991.
7. *H. Eisenmann and F. M. Johannes*. Generic global placement and Floorplanning. In Proc. Design Automation Conf, pp. 269-274, 1998.
8. Kong, T. A novel net weighting algorithm for timing-driven placement. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. ACM, New York, pp. 172–176, 2002.

9. H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI Module Placement Based on Rectangle-packing by the Sequence-Pair. IEEE Trans. on CAD, Vol. 15, No. 12, pp. 1518–1524, 1996.
10. Xuliang Zhang, Yoji Kajitani. Space Planning: Placement of Modules with Controlled Empty Area by Single Sequence. ASP-DAC 2004, pp. 25-31.
11. M. Srinivas and Lalit M. Patnaik. Genetic Algorithms – A Survey. IEEE Computer, Vol.-27, No.-6, June 1994, pp 17-26.
12. *J. Cong, T. Kong, J. R. Shinnerl, M. Xie, and X. Yuan.* Large-scale circuit placement: Gap and promise. In Proc. of the International Conference on Computer-Aided Design, November 2003.

В.Б. Тарасов, П.В. Афонин, Д. Картежников

МОДЕЛЬ ФОРМИРОВАНИЯ СТРУКТУР ОБЪЕДИНЕНИЙ ВИРТУАЛЬНЫХ ПРЕДПРИЯТИЙ НА ОСНОВЕ ПОПУЛЯЦИИ ЭВОЛЮЦИОНИРУЮЩИХ АГЕНТОВ

Введение. Предлагается модель формирования структур объединений виртуальных предприятий, разработанная на основе анализа возможных видов взаимодействия между физическими предприятиями.

Рассматривается сообщество виртуальных предприятий, представляющее собой популяцию эволюционирующих агентов, т.е. эволюционирующую многоагентную систему (ЭМАС) с определенным набором параметров. Основу модели составляет эволюционный алгоритм, использующий модифицированные генетические операторы. При этом ставится задача нахождения оптимальных структур объединений предприятий. С позиции функционирования ЭМАС эволюционный алгоритм выполняет роль некоторого вышестоящего координатора, накладывающего ограничения на деятельность всей популяции агентов. Анализ результата наложения этих ограничений позволяет накапливать в популяции положительные свойства и формировать наиболее подходящие под конкретные условия структуры объединений предприятий.

1. Формализованное представление модели. Формализованное описание модели можно представить следующим образом.

Рассматриваемое сообщество предприятий представляется в виде многоагентной эволюционирующей системы:

$$EMAS = \{A_1, A_2, \ldots, A_i, \ldots, A_n\}.$$

Каждый ***i***-ый агент системы характеризуется набором параметров:

$$A_i : (a_{i1}, a_{i2}, a_{i3}, a_{i4}).$$

Для решения задачи в ***МАС*** требуется, используя подходящие эволюционные (генетические) операторы, сформировать организационные структуры, оптимальным образом удовлетворяющие внутреннему состоянию и намерениям каждого агента. Образованию каждой организационной структуры предшествует создание нового члена популяции (порождение потомка агентами, образующими объединение).

Таким образом, множество агентов популяций можно разделить на два подмножества:

$$\{A_1, A_2, \ldots, A_i, \ldots, A_n\} = \{A_{p1}, A_{p2}, \ldots, A_{pm}\} \bigcup \{A_{d\,m+1}, A_{d\,m+2}, \ldots, A_{dn}\},$$

где $\{A_{p1}, A_{p2}, \ldots, A_{pm}\}$ – множество родителей; $\{A_{d\,m+1}, A_{d\,m+2}, \ldots, A_{dn}\}$ – множество потомков.

Появление ***i***-го потомка связано с взаимодействием его родителей: