In the next few years, logic capacities for fieldprogrammable gate arrays are expected to exceed one million gates per device. While this expansion of FPGA device resources offers the promise of exceptional finegrained performance for developing technologies such as ASIC prototyping and FPGA computing, supporting computer-aided design tools have yet to be developed to target these devices rapidly and efficiently. This paper addresses the compilation time issue for routing array FPGAs with segmented routing architectures. By treating the routing problem as an A∗ search, it is possible to trade additional device routing resources for decreased router run-time by converting an exhaustive breadth-first maze route into a shorter depth-first route. It is shown that for the depth-first case, the sparse nature of FPGA routing switches in commerical architectures, such as the Xilinx XC4000 family, necessitates an additional localized search near net inputs, called domain negotiation, to aid in directing the route of each design net onto a set of routing resources most likely to lead to a successful route. For a set of large FPGA benchmarks, a route time speedup of over an order of magnitude for an iterative maze router configured for depth-first routing is shown when compared to the same router configured for a breadth-first search.

В следующие несколько лет ожидается, что логический объём для ПЛИС превысит 1 миллион затворов на устройство. В то время, как расширение ресурсов ПЛИС обещает исключительные возможности для развивающихся технологий, таких как прототипирование интегральных схем специального назначения и программирование ПЛИС, поддержка САПР ещё недостаточно развита, чтобы использовать эти устройства быстро и эффективно. В этой статье рассматривается проблема времени компиляции для ПЛИС с сегментированной архитектурой маршрутизации. Рассматривая проблему маршрутизации с помощью алгоритма А\*, представляется возможным обменять дополнительные ресурсы маршрутизации на уменьшенное время работы маршрутизатора путём конвертации ресурсоёмкого алгоритма поиска пути в ширину в более быстрый(короткий) алгоритм поиска в глубину. Показано, что для случая поиска в глубину, малое количество переходов в коммерческих архитектурах, таких как семейство Xilinx XC4000, требует дополнительный локализованный поиск рядом с входами шин, называемый «доменное согласование» для помощи в поиске маршрута каждой спроектированной шины в набор ресурсов маршрутизации, что чаще всего ведёт к успешному построению маршрута. Для большого количества тестов FPGA, видно увеличение скорости маршрутизации на порядок для итеративного алгоритма поиска в глубину, по сравнению с тем же маршрутизатором, настроенного для поиска в ширину.

Abstract

The primary advantage of using 3D-FPGA over 2D-FPGA is that the vertical stacking of active layers reduce the Manhattan distance between the components in 3D-FPGA than when placed on 2D-FPGA. This results in a considerable reduction in total interconnect length. Reduced wire length eventually leads to reduction in delay and hence improved performance and speed. Design of an efficient placement and routing algorithm for 3D-FPGA that fully exploits the above mentioned advantage is a problem of deep research and commercial interest. In this paper, an efficient placement and routing algorithm is proposed for 3D-FPGAs which yields better results in terms of total interconnect length and channel-width. The proposed algorithm employs two important techniques, namely, Reinforcement Learning (RL) and Support Vector Machines (SVMs), to perform the placement. The proposed algorithm is implemented and tested on standard benchmark circuits and the results obtained are encouraging. This is one of the very few instances where reinforcement learning is used for solving a problem in the area of VLSI.

Основное преимущество при использовании 3D ПЛИС над 2D состоит в том, что при вертикальном расположении активных слоёв уменьшается Манхеттеновское расстояние между компонентами в 3D ПЛИС по сравнению с его 2D аналогом. Это приводит к значительному сокращению в суммарной длине межсоединений. Уменьшенная длина проводов в конечном итоге ведёт к сокращению задержки и, следовательно, улучшенной производительности и скорости. Разработка эффективных размещений и алгоритмов маршрутизации для 3D ПЛИС, которые в полной мере могли бы использовать вышеописанные преимущества – проблема глубокого исследования и коммерческого интереса. В этой статье предлагается эффективный алгоритм размещения и трассировки для 3D ПЛИС, который приносит лучшие результаты в терминах общей длины межсоединений и ширины канала. Предложенный алгоритм для осуществления размещения применяет две важные техники(методики), называемые “обучение с подкреплением” и “метод опорных векторов”. Предложенный алгоритм внедрён и оттестирован на стандартных тестовых схемах и полученные результаты обнадёживают. Это один из не многих примеров, где “обучение с подкреплением” используется для решения проблем в области VLSI (прим. VLSI - very large scale integration – сверхбольшая степень интеграции, на русский чаще всего не переводится).

Abstract—We present a method for accelerating server applications using a hybrid CPU+FPGA architecture and demonstrate its advantages by accelerating Memcached, a distributed key-value system. The accelerator, implemented on the FPGA fabric, processes request packets directly from the network, avoiding the CPU in most cases. The accelerator is created by profiling the application to determine the most commonly executed trace of basic blocks which are then extracted. Traces are executed speculatively within the FPGA. If the control flow exits the trace prematurely, the side effects of the computation are rolled back and the request packet is passed to the CPU. When compared to the best reported software numbers, the Memcached accelerator is 9.15x more energy efficient for common case requests.

Мы представляем метод ускорения серверных приложений с использованием гибридной процессор+ПЛИС архитектуры и демострируем её преимущества, ускоряя распределённую систему ключ-значение Memcached (прим. Memcached – программное обеспечение, реализующее сервис кэширования данных в оперативной памяти на основе хэш-таблицы). Ускоритель, внедрённый на FPGA-матрице, обрабатывает запрошенные пакеты непосредственно из сети, избегая использования CPU в большинстве случаев. Ускоритель создаётся путём профилирования приложения для определения наиболее часто выполняемой трассировки базовых блоков, которые затем извлекаются. Предполагается, что трассировка выполняется внутри ПЛИС. Если контрольный поток прерывает трассировку преждевременно, побочные эффекты вычисления откатываются назад и запрошенный пакет передаётся процессору. При сравнении с наилучшими цифрами в отчётах ПО, ускоритель Memcached в 9.15 раз более энергоэффективный для обычных запросов.