

Разработка инфраструктуры для создания специализированных ускорителей на основе Interaction Nets

Кубышкин Е.А., СПбГУ, Санкт-Петербург st098235@student.spbu.ru,
Пономарев Н.А., СПбГУ, Санкт-Петербург n.ponomarev@spbu.ru

Аннотация

Нерегулярный параллелизм — ситуация, когда сложно заранее определить количество независимых подзадач — важная проблема на пути к высокой производительности в таких важных областях, как искусственный интеллект, анализ графов, и многих других. Одним из решений этой проблемы является модель вычислений Interaction Nets, для которой параллелизм подобного рода естественен. Тем не менее на данный момент существуют только программные реализации данной системы. В работе представлен стенд для прототипирования сопроцессоров, основанных на Interaction Nets: от программы на высокоуровневом языке до прошивки ПЛИС.

Введение

Потребность в параллельной обработке данных нарастает. Это связано как с усложнением используемых алгоритмов, так и с ростом объёма обрабатываемых данных. Однако в ряде задач, например, связанных с обработкой графов, искусственным интеллектом и разреженной линейной алгеброй, возникает проблема нерегулярного параллелизма с которой традиционные архитектуры в виду своей специфики справляются неэффективно.

Interaction Nets [1] — модель вычислений, основанная на переписывании графов, для которой естественен нерегулярный параллелизм. Программы в ней представляются в виде неориентированного меченого графа, называемого сетью. Процесс вычисления (редукции) происходит за счёт переписывания подграфов. Важным свойством системы является то, что редукции локальны, что позволяет применять все доступные переписывания одновременно. Кроме того известно, что модель полна по Тьюрингу, а результат вычислений не зависит от порядка редукций.

Важно отметить, что, меняя правила редукции и метки на узлах, можно параметризовать сети и оптимизировать вычисления для какой-то предметной области. Таким образом Interaction Nets можно рассматривать, как

крупно-гранулярную реконфигурируемую архитектуру (CGRA) [2]. Такие архитектуры сочетают гибкость программных моделей с высокой производительностью специализированных аппаратных решений.

На данный момент существуют только программные реализации Interaction Nets [3], поэтому нам хотелось бы изучить можно ли ускорить некоторые алгоритмы в помощью сопроцессора на Interaction Nets.

Цели и задачи

Для исследования данной гипотезы нами был начат проект Lamagraph¹, целью которого является разработка параметризуемого многоядерного вычислителя на основе Interaction Nets.

Для её достижения были выделены следующие этапы.

1. Создание минимальной инфраструктуры для создания специализированных вычислителей на основе Interaction Nets.
2. Разработка eDSL для спецификации Interaction Nets.
3. Добавление поддержки одновременной редукции в ускоритель.
4. Проведение сравнительных экспериментов на задачах разреженной линейной алгебры, анализа графов и искусственного интеллекта.

На данный момент проект находится на первом этапе: создание минимального рабочего окружения для прототипирования вычислителей и работы с ними, который состоит из следующих подшагов.

- 1.1. Реализация высокоуровневого языка программирования.
- 1.2. Разработка транслятора из высокоуровневого языка в Interaction Nets.
- 1.3. Реализация интерпретатора Interaction Nets.
- 1.4. Разработка генератора прошивки вычислителя для ПЛИС.

Требования к первому этапу

Первый этап главным образом создаёт фундамент для дальнейшей разработки и экспериментов. Так что фокус смещён в сторону скорости разработки и анализа результатов для возможности оперативно проверять рабочие гипотезы. Исходя из этого, к проекту были выставлены следующие требования.

¹Репозитории проекта доступны по ссылке: <https://github.com/Lamagraph>

Гомогенность. Использование единого стека технологий позволяет упростить создание и поддержку программно-аппаратного стека, а также цепочку обработки данных.

Целостность. Получение полнофункционального прототипа, содержащего все компоненты, важнее, чем детальная проработка какого-то отдельного компонента. На данном этапе необходимо в первую очередь выстроить взаимодействие между компонентами, а также увидеть, какие сложности возникают не только при реализации отдельных компонент, но и при интеграции между ними. Это позволяет быстрее выявлять и исправлять неудачные архитектурные решения.

Гранулярность. На каждом шаге работы программно-аппаратного стека есть результат исполнения программы. Такой подход позволяет тестировать как каждую отдельную компоненту, так и сразу несколько.

Реализация

В качестве основного языка был выбран Haskell из-за его удобства для работы высокоуровневыми абстракциями, лёгкостью в создании DSL (и eDSL) и наличии HDL, основанного на нём.

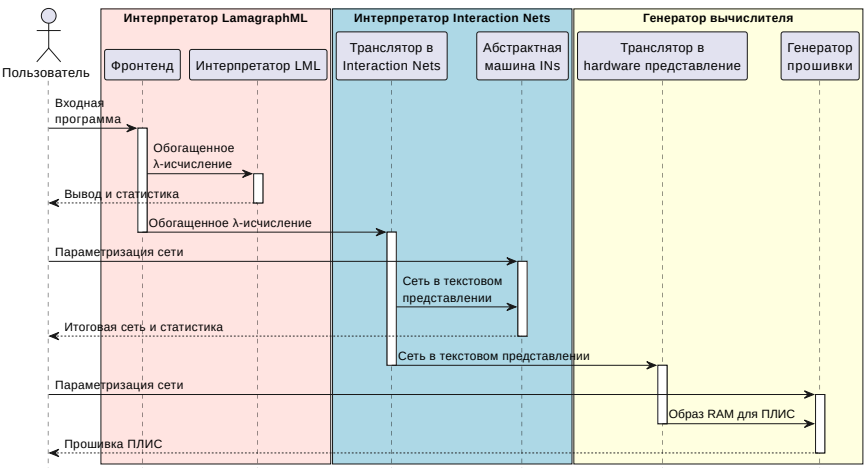


Рис. 1: Диаграмма взаимодействия пользователя с программно-аппаратным стеком Lamagraph.

Взаимодействие с пользователем

Входная программа на LamagraphML сначала транслируется в обогащённое λ -исчисление; далее это представление можно транслировать в сеть в текстовом представлении, для которого реализована абстрактная машина; и из текстового представления можно получить образ памяти для ПЛИС и получить прошивку. На каждом этапе можно исполнить программу и получить статистику и результат исполнения. Весь программно-аппаратный стек представлен на Рис. 1. Было решено разделить проект на три большие части, далее они будут описаны более подробно.

Интерпретатор LamagraphML

Многие реализации Interaction Nets придумывают собственные языки программирования (например, [4] или Bend²), которые не привычны программисту, поскольку пытаются текстово описывать граф. Lamagraph решает данную проблему, поскольку в нашем проекте Interaction Nets используется в качестве промежуточного представления, а пользователю предоставляется относительно привычный высокоуровневый ML-подобный язык — LamagraphML. Функциональный язык выбран из-за наличия схем трансляции λ -исчисления в Interaction Nets. После работы фронтенда мы получаем программу в виде обогащенного λ -исчисления, которую потом можно либо интерпретировать, либо транслировать в Interaction Nets.

Интерпретатор Interaction Nets

Стандартным представлением для Interaction Nets является графовое, однако с ним (как и с любым графом) трудно работать в функциональных языках программирования. Тем не менее существует текстовое представление сетей [5], для которого разработана абстрактная машина [6]. Поэтому для реализации было выбрано именно оно.

В процессе реализации транслятора в Interaction Nets была обнаружена проблема: существующие схемы трансляции работают только с чистым λ -исчислением. Попытки расширить схему трансляции предпринимались в [7], однако такой подход требует использование динамического базиса агентов. Но поскольку вычислитель прошивается на ПЛИС, то базис агентов должен быть фиксирован, и подход предложенный в статье нам не подходит. Поэтому трансляция в Interaction Nets конструкций обогащённого λ -исчисления —

²Репозиторий проекта: <https://github.com/HigherOrderCO/Bend>

предмет дальнейшей работы.

Генератор вычислителя

Языком программирования ПЛИС был выбран Clash [8] — HDL, основанный на Haskell, который потом транслируется в традиционные HDL. Такое решение продиктовано простотой, в сравнении с традиционными HDL, создания и работы с высокоуровневыми абстракциями. Clash позволяет описывать комбинационные схемы почти как обычный Haskell код, что открывает возможность писать модульные и даже property-based тесты. Более сложные последовательностные схемы же можно симулировать с помощью симулятора Clash и писать для них test bench файлы.

Дизайн вычислителя сделан таким образом, чтобы явно отделить разные этапы выполнения редукции: работа с памятью, применение правила редукции и встраивание локальных изменений в глобальную сеть. Это позволило оставить пространство для дальнейших оптимизаций. Также такое решение помогло вынести реализацию спецификации Interaction Nets почти полностью на уровень программирования на типах, что значительно упрощает отладку и интеграцию с остальными компонентами системы.

Заключение

По итогам работы над первым этапом, можно сказать, что получилось создать полный программно-аппаратный стек разработки вычислителя, основанного на Interaction Nets: от программы на высокоуровневом языке до прошивки ПЛИС. Более подробно, на основе анализа предметной области и общего виденья проекта, были выделены ключевые требования (**гомогенность, целостность и гранулярность**), и в соответствие с ними реализованы следующие компоненты.

- Интерпретатор языка LamagraphML, поддерживающий невзаимную рекурсию и встроенные алгебраические типы данных `list` и `option`.
- Транслятор из чистого λ -исчисления в Interaction Nets в стратегии Call-by-Value.
- Полноценный интерпретатор для Interaction Nets на основе абстрактной машины.

- Генератор прошивки одноядерного вычислителя на основе Interaction Nets, поддерживающий параметризацию агентами и правилами редукции.

Также удалось выявить узкое место в виде трансляции обогащённого λ-исчисления в Interaction Nets.

Список литературы

- [1] Lafont Yves. Interaction nets // Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. – POPL '90. – New York, USA: Association for Computing Machinery, 1989. – С. 95–108
- [2] Liu L. et al. A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications //ACM Computing Surveys (CSUR). – 2019. – Т. 52. – №. 6. – С. 1-39.
- [3] Hassan Abubakar, Mackie Ian, Sato Shinya. An Implementation Model for Interaction Nets // Electronic Proceedings in heoretical Computer Science. – 2015. – may. – Vol. 183. – С. 66–80
- [4] Sato S. Design and implementation of a low-level language for interaction nets : дис. – University of Sussex, 2015.
- [5] Fernández M., Mackie I. A calculus for interaction nets //International Conference on Principles and Practice of Declarative Programming. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1999. – С. 170-187.
- [6] Pinto J. S. Sequential and concurrent abstract machines for interaction nets //International Conference on Foundations of Software Science and Computation Structures. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2000. – С. 267-282.
- [7] Jiresch E. R. W. Extending interaction nets towards the real world : дис. – Technische Universität Wien, 2012.
- [8] Christiaan Baaij, Matthijs Kooijman, Jan Kuper et al. CLaSH: Structural Descriptions of Synchronous Hardware Using Haskell // 2010. – 09. – С. 714–721