УДК 681.32

ОРГАНИЗАЦИЯ КОМПЛЕКСНОГО ВИЗУАЛЬНОГО АНАЛИЗА ПРИ ПАРАЛЛЕЛЬНОМ ВЫПОЛНЕНИИ ПРОЦЕДУР ОПТИМИЗАЦИИ В СИСТЕМЕ ЛОГИЧЕСКОЙ ОПТИМИЗАЦИИ ФУНКЦИОНАЛЬНО-СТРУКТУРНЫХ ОПИСАНИЙ ЦИФРОВЫХ СХЕМ

Логинова И.П.

ОИПИ НАН Беларуси, Минск, Республика Беларусь, irilog@email.ru

Аннотация: для анализа результатов логической оптимизации, выполняемой с использованием параллельных вычислений в системе логической оптимизации функционально-структурных описаний дискретных устройств FLC-2, предложен комплексный подход применения таких инструментов визуализации, как диаграммы Ганта и табличные представления. Визуальные табличные представления, получаемые после завершения проектной процедуры с применением параллельных вычислений, автоматически выгружаются в базу данных SQLite и используются для оценки времени выполнения и эффективности оптимизации.

Ключевые слова: автоматизация проектирования, параллельные вычисления, цифровая схема, диаграммы, модель, база данных, OpenMP.

I. ВВЕДЕНИЕ

Рост размерности СБИС, уменьшение технологических норм приводит к усложнению процесса проектирования и подводит к необходимости использования параллельных вычислений в САПР СБИС, как одного из основных способов уменьшения времени проектирования. Возможность распараллеливания прикладных задач логического проектирования определяется рядом факторов: разработкой параллельных алгоритмов для решения прикладных задач; построением эффективных параллельных программ, реализующих эти алгоритмы; наличием технологий для реализации параллельных программ на многопроцессорной (многоядерной) архитектуре.

Большая часть программ в составе различных САПР цифровых схем, реализующих очень эффективные алгоритмы решения логико-комбинаторных задач, представлены исполняемыми кодами без привязки к архитектуре используемых вычислительных средств. Однако разработка параллельных алгоритмов для большинства логико-комбинаторных задач, возникающих в логическом проектировании цифровых схем, в принципе, невозможна [1]. Один из вариантов организации параллельных вычислений на многоядерной вычислительной системе, при осуществлении которого не потребовалась модернизации или создания новых параллельных алгоритмов решения логико-комбинаторных задач, реализован в системе логической оптимизации функционально-структурных описаний дискретных устройств FLC-2, разработанной в ОИПИ НАН Беларуси [2-4].

В основу используемого подхода положена организация параллельной работы программ, выполняющих оптимизацию с частями исходного объекта (логической схемы). На разных этапах проведения оптимизации логическая схема представляется в виде различных описаний, преобразование которых осуществляется методами детализации, экстракции, декомпозиции. При проведении этих преобразований учитываются характеристики и состояние логической схемы. С учетом этих атрибутов формируются цепочки проектных процедур и операций, составляющих маршрут проектирования. Процесс декомпозиции сопровождается проведением ряда проектных операций по выявлению и устранению информационных зависимостей между частями объекта. Результатом таких действий является разбиение на независимые части объекта проектирования, появление возможности организации параллельных вычислений с этими независимыми частями. Включение в состав системы FLC-2 проектных процедур оптимизации, которые выполняются с использованием параллельных вычислений, позволяет существенно сократить время проектирования и повысить размерность решаемых задач [2-4].

Производительность параллельной программы зависит от большого числа параметров: архитектуры вычислительной системы, непосредственно самой задачи, библиотеки поддержки выполнения, коммуникационной среды. Разработка параллельной программы требуют наличия мощного набора инструментов, позволяющих с той или иной степенью автоматизации и визуализации эффективно проводить расчеты. Графические визуализации становятся все более мощными и удобными инструментами не только для отладки и получения характеристик производительности системы, но и для понимания поведения параллельной программы во время её выполнения. Поэтому рассмотрение различных методов визуализации тех или иных аспектов поведения параллельной программы, а также построение графического отображения этих аспектов, является актуальным и представляет интерес.

В данной работе представлены несколько моделей визуальных представлений, разработанных для отображения результатов оптимизации, полученных в результате использования параллельных вычислений в системе логической оптимизации FLC-2 [2-4].

II. ПАРАЛЛЕЛЬНЫЙ ЗАПУСК ПРОГРАММ ОПТИМИЗАЦИИ В СИСТЕМЕ FLC2

Особенно трудоемким этапом синтеза, является проведение технологически независимой оптимизации. В отдельных случаях необходимо нахождение эффективных решений для проекта, составленного из большого числа элементов. Разбиение описания логической схемы на подмножество элементов преследует цели: понижение размерности объекта оптимизации, получение оптимального решения для всех элементов, и получение оптимального решения для всего проекта. Применение некоторых проектных операций системы FLC-2 позволяет разбить проект логической схемы на совокупности не зависящих друг от друга блоков.

Разбиение на части описания проекта производится с применением ряда проектных процедур, реализующих различные методы декомпозиции. По мере необходимости проводятся проектные операции по выявлению и устранению информационных зависимостей между полученными в результате разбиения частями исходного проекта, что приводит к их преобразованию в «независимые» объекты. Независимость частей проекта, в контексте проведенных процедур декомпозиции означает, что входные аргументы для работы с одной частью проекта не определяются результатами вычислений с другими частями проекта. В итоге появляется возможность проведения параллельных вычислений без блокировок и барьерной синхронизации при выполнении проектных операций оптимизации с частями проекта. В системе FLC2 [2-4] реализована концепция организации параллельных вычислений на многоядерной вычислительной системе в стиле, называемом параллелизмом задач (task-based parallel) [5]. Согласно этой концепции в каждой задаче task можно организовать выполнение проектной операции с отдельными частями проекта логической схемы.

Организация параллельной работы выполнена с использованием технологии OpenMP [6], открытого стандарта для реализации многопоточных приложений, представленного набором различных директив, функций соответствующей библиотеки и переменных окружения. Параллельная работа программ оптимизации на многоядерном процессоре основана на работе в каждом ядре, так называемых процессов исполнения. В частности, директивы OpenMP использованы для создания пула задач, которые в ходе работы параллельной программы распределяются между потоками исполнения. Для каждого ядра автоматически создаются очереди задач, потоки исполнения в ядрах по мере выполнения одних задач берут из очереди следующие. Реализация параллелизма задач в системе FLC2 основана на следующих положениях:

– Исполняемый код программы, запущенный для выполнения, образует процесс. Средствами OpenMP, размещенными в головной программе, можно осуществить запуск нескольких параллельных процессов с исполняемыми кодами программ оптимизации.

– В каждом из параллельных процессов выполняется внешняя программа, запуск которой производится посредством команды start, аргументы которой определяют имя и аргументы запускаемой внешней программы.

– Исходные объекты необходимые для работы программ, которые выполняются в параллельных процессах, созданных головной программой, должны быть независимыми друг от друга.

Псевдокод создания параллельных (task-based) процессов с использованием директив OpenMP приведен на рисунке 1.

```
…
# pragma omp parallel for
for(i=0; i<N; i++)        // цикл по N – числу запусков выполнения внешней программы
{
# pragma omp task …       // образование новой независимой задачи
{
# pragma omp critical 1 // критическая секция 1
{ … }                     // время начала задачи , занести в трассу задачи
start (new process(…exe,…argv[…]…)); // выполнение внешней программы
# pragma omp critical 2 // критическая секция 2
{ … }                     // время окончания задачи, занести в трассу задачи
}                         // завершение кода задачи
 }                        // конец цикла – N параллельных запусков внешней программы
…
```

Рисунок 1. Фрагмент псевдокода запуска параллельного выполнения внешних программ

Данный псевдокод, иллюстрирует возможность параллельного выполнения внешних программ, которые представлены только своим исполняемым кодом. Директивы #pragma omp task определяют явные (explicit) задачи, вызывают порождение отдельных задач (task), которые могут быть немедленно исполнены потоком, встретившим эту директиву, либо же помещены в очередь и быть выполненными позже одним из потоков исполнения. В критических секциях 1 и 2, определяемых директивой #pragma omp critical, формируется трасса отдельного параллельного процесса. Средствами библиотек OpenMP определяется: время начала процесса, время завершения процесса, время выполнения программы оптимизации в каждом процессе. Анализ временных событий важен, как для оценки эффективности проведенной логической оптимизации, так и для оценки быстродействия выполнения параллельной программы.

Алгоритм реализации параллельных вычислений следующий: в головной программе запускается множество параллельных процессов (задач task); задачи динамическим образом распределяются по всем ядрам (включая логические процессоры, если они есть); у всех процессов свои адресные пространства; процессы взаимодействует с памятью независимо, в каждом процессе внешняя программа работает со своими входными данными. Когда в головной программе производится запуск другой task, создается новый (дочерний) процесс. Работа дочерних процессов может осуществляться в синхронном и асинхронном режиме. Все дочерние процессы выполняют, как правило, один код. Этот код представлен командой start, которая производит запуск консольной реализации внешней программы [4]. Аргументами команды start являются параметры командной строки программы, запускаемой в этом процессе. В случае, если в головной программе работу дочерних процессов определяла только директива #pragma omp parallel for, то в конце параллельной области происходит неявная барьерная синхронизация, т.е. все потоки (процессы) останавливаются до тех пор, пока последний поток (процесс) не выполнит свою программу целиком. Но при вставке в цикл директивы #pragma omp task синхронизация потоков отменяется, создается набор задач task, число задач определяется размером цикла, все задачи помещаются в очередь. Из очереди задачу (процесс) может выбрать один из потоков исполнения. Задач может быть намного больше числа потоков исполнения (числа ядер). Планировщик реализует формирование очереди, проводит распределение задач по потокам исполнения, применяя стратегию work-stealing [7], суть которой заключается в том, что освободившийся поток при отсутствии ожидающих задач в своей очереди заимствует задачу из очереди одного из потоков исполнения.

III. СРЕДСТВА ВИЗУАЛИЗАЦИИ ПРИ ВЫПОЛНЕНИИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

При реализации средства визуализации могут использоваться самые различные методики сбора данных и связанный с ними способ организации работы визуального средства. Существует два различных подхода: первый и более простой в реализации – off-line анализ, а также более прогрессивный и информативный on-line анализ [8]. Off-line анализ подразумевает обработку данных после выполнения параллельной программы. Эта обработка основывается на том, что в процессе выполнения программы создается так называемая трасса (лог-журнал, содержащий информацию о ходе выполнения программы). В конкретных участках головной программы, запускающей параллельные процессы, псевдокод которой показан на рисунке 1, проводится сбор значений временных событий, происходящих в отдельных процессах. После завершения работы программы в отдельном процессе трасса этого процесса сохраняется. Для визуального анализа, реализованного в FLC2 подхода к организации параллельных вычислений, разработано программное средство визуализации [9]. Согласно данным off-line трассы, полученной в процессе работы параллельной программы, строится графическое отображение, которое дает наглядное представление о загрузке ядер процессора, о времени выполнения оптимизации для элементов проекта. Кроме временных отметок в трассе каждого процесса сохраняются: номер выполняемой в этом процессе задачи, идентификатор потока исполнения, имя выполняемой в процессе программы. Визуализация трассы параллельных процессов осуществляется путем формирования графического представления в виде модифицированных диаграмм Ганта. Перед началом работы программы визуализации формируется объединенный трассировочный off-line файл из трасс, получаемых в каждом процессе. В программе визуализации проводится: подготовка информации из трассы для графического отображения, а затем реализация графического отображения с использованием библиотеки QtCharts. Подготовка трассы заключается в организации многоуровневой сортировки хэш-таблицы, построенной по извлеченной трассировочной информации. В качестве инструмента визуализации данных трассы, используется модуль графической библиотеки QtCharts, который интегрирован с фреймворком Qt. Библиотека QtCharts является кроссплатформенной, обеспечивает высокую производительность [10], поддерживает широкий спектр разнообразных типов визуализаций (графиков, гистограмм, диаграмм и др.). Для графического отображения выбрана диаграммы в виде горизонтальной диаграммы с накоплением (HorizontalStackedBarChart). Этот тип диаграммы представляет данные трассы в виде

горизонтально сложенных столбцов, сгруппированных по следующим категориям: по вертикали представлено распределение процессов по ядрам процессора; по горизонтали представлена шкала времени выполнения параллельной программы. В горизонтальном столбце «последовательное сложение» блоков разной длины определяется порядком выполнения процессов в заданном ядре. Размер каждого блока определяется временем выполнения оптимизации в одном процессе. Это время вычисляется при анализе временных событий в трассе отдельного процесса.

IV. ИСПОЛЬЗОВАНИЕ ГИСТРОГРАММ И ТАБЛИЦ СОВМЕСТНО С МОДЕЛЯМИ ДАННЫХ

Для организации комплексного визуального анализа, суть которого заключается в реализации разных вариантов визуализации одних и тех же данных, используется архитектура шаблона Model/View (Модель/Представление), которая является модификацией шаблона MVC [11]. Архитектура Model/View имеет более простую структуру, при этом механизм хранения данных и работа с ними отделены от их представления. Такое разделение позволяет отображать одни и те же данные в нескольких разных представлениях и реализовывать новые типы представлений без изменения базовых структур данных. На рисунке 2 представлена архитектура шаблона, который использует одну Модель (источник данных) для одновременного представления данных из модели двумя способами: в виде таблицы и в виде диаграмм. Согласно схеме работу по обновлению данных между источником данных и Моделью берет на себя Представление.

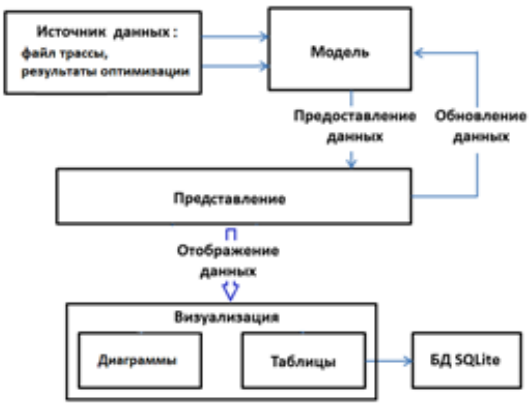


Рисунок 2. Архитектура шаблона M/V при работе с трассой параллельных процессов

Процесс работы в этой архитектуре выглядит следующим образом. Модель взаимодействует с источниками данных (трасса параллельных процессов или, например, статистика, предоставляемая FLC2 по результатам работы процедур оптимизации), обеспечивая интерфейс для других компонентов архитектуры. Представление через Модель может взаимодействовать с данными, в частности, может получать данные и отображать их. Для отображения в Представлении применяется программа визуализации, описанная выше. Когда данные трассы изменяются, программа визуализации напрямую взаимодействует с Моделью, уведомляя ее об изменении данных. Модель с данными, можно отображать как на гистограмме, так и в таблице. Пример визуализации двух представлений (в виде диаграммы и в виде таблицы) приведен на рисунке 3.

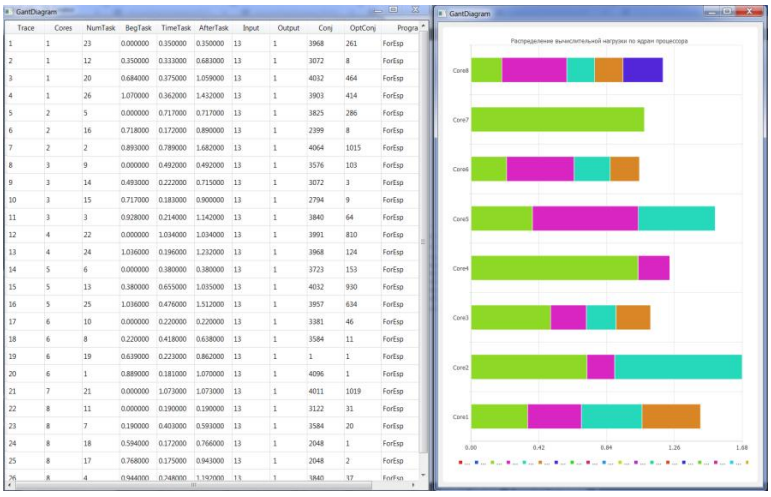


Рисунок 3. Визуализация Модели в виде диаграммы и в виде таблицы (для схемы из 26-ти блоков)

Эти представления визуализации получены после проведения оптимизации логической схемы, которая разделена на 26 независимых блоков. Представление в виде таблицы содержит информацию трасс всех процессов с указанием событий, начиная от идентификатора процесса, номера ядра, номера задачи и всех временных меток начала (столбец BegTask), длительности (столбец TimeTask) и завершения каждой задачи (столбец AfterTask). Кроме того, таблица дополнена следующей информацией, полученной при проведении оптимизации логической схемы: для каждого блока приводятся структурные параметры (число входов, число выходов, число конъюнкций) до и после оптимизации, имя программы, которая проводила оптимизацию с этой частью схемы. При более подробном рассмотрении диаграммы и таблицы можно отметить, что расположение данных в этих визуальных представлениях синхронизированы друг с другом.

Интеграция в FLC2 средств комплексного визуального анализа в значительной степени способствует решению проблемы сохранения полученных при экспериментах с разными проектами результатов и их дальнейшей структуризации. Данные можно хранить как в модели, так и выгружать в базу данных. Программа визуализации позволяет выгружать данные, сформированные Моделью в виде табличного представления, в такие же таблицы реляционной базы данных SQLite посредством всего одного запроса. Архитектура M/V и средства платформы Qt предоставляет возможностью работы с SQLite [10]. SQLite – это компактная встроенная система управления базами данных (БД), для которой не нужен отдельный сервер, запускать и работать с ней можно в системе, где проводится проектирование. Все данные эта БД хранит в одном файле, доступ к ним осуществляется локально. SQLite работает по реляционной модели, то есть все данные в базе данных связаны в виде таблиц. SQLite используют язык структурированных запросов SQL для хранения и обработки информации. Интеграция Модели базы данных и Модели визуальных табличных представлений позволяют организовать сериализацию данных из табличного представления в таблицы базы данных. После создания и подключения БД одним SQL-запросом выгружается все содержание таблицы, представленной визуальным образом. Текущее изменение БД SQLite в зависимости от выбранной стратегии обновления определяется обновлениями данных Модели в программе визуализации. Таким образом, таблицы в БД создаются, обновляются и добавляются согласно запросам, которые формируются той или иной стратегией обработки данных, полученных при проведении оптимизации.

V. ЗАКЛЮЧЕНИЕ

Рассмотрена методика реализации средств визуального анализа данных в системе логической оптимизации FLC2, которые позволяют провести интеграцию нескольких подходов: визуализации результатов, полученных при проведении оптимизации, в виде табличного представления и визуального представления off-line трассы, формируемой при выполнении параллельных процессов. Визуализация трассы осуществляется путем формирования графического представления в виде горизонтальной диаграммы с накоплением (диаграммы Ганта), которая дает представление о распределении задач по ядрам многоядерного процессора.

Методика получения двух визуальных представлений на основе единой модели данных и связывания визуального табличного представления с встроенной базой данных SQLite помогает проанализировать и оценить эффективность распараллеливания в различных маршрутах логического проектирования, осуществляет автоматическую сериализацию в таблицах реляционной базы данных SQLite результатов выполнения различных оптимизационных процедур. Сохранение в БД всей информации из визуального табличного представления в определенной мере автоматизирует сбор и накопление статистики о результатах оптимизации логических схем в разных маршрутах проектирования. Применение методов машинного обучения при извлечении информации из этих табличных данных предоставляет возможность выбора наиболее эффективной программы оптимизации для логической схемы с теми или иными параметрами. Выбор в результате обучения на табличных данных БД SQLite наиболее эффективной программы оптимизации приведет к получению наилучшего решения логико-комбинаторных задачи оптимизации, как с точки зрения увеличения быстродействия, так и нахождения оптимальных характеристик логической схемы.

ЛИТЕРАТУРА

[1] Черемисинов, Д.И. Использование параллельных вычислений при автоматизированном проектировании СБИС / Д.И. Черемисинов, Л.Д. Черемисинова // Проблемы разработки перспективных микро- и наноэлектронных систем – 2016. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. – М.: ИППМ РАН, 2016. Часть – С. 32-39.

[2] Бибило, П.Н. Система логической оптимизации функционально-структурных описаний цифровых устройств на основе продукционно-фреймовой модели представления знаний /П. Н. Бибило, В. И. Романов // Проблемы разработки перспективных микро- и наноэлектронных систем - (МЭС-2020) : IX

Юбилейная Всероссийская научно-техническая конференция : Сборник трудов ; под общей ред. А. Л. Стемпковского. – Москва : ИППМ РАН, 2021 Москва : ИППМ РАН, 2020-.С. 9–16.

[3] Логинова, И.П. Использование технологий параллельных вычислений для систем с общей памятью на примере задачи минимизации булевых функций / И.П. Логинова // Информационные технологии в промышленности, логистике и социальной сфере (ITI*2019) : тезисы докладов X Международной научно-технической конференции (23 – 24 мая 2019 года, Минск). – Минск : ОИПИ НАН Беларуси, 2019. – С. 130 – 133.

[4] Логинова, И.П. Анализ быстродействия при реализации параллельных вычислений для решения задач оптимизации в системе логического проектирования / И. П. Логинова // IX Международная научно-практическая конференция «BIG DATA and Advanced Analytics» (BIG DATA 2023): Сборник научных статей в 2 ч. Ч. 1, (Республика Беларусь, Минск, 17-18 мая 2023 г.). – Минск: БГУИР, 2023. – С. 365–378.

[5] J. Ciesko, S. Mateo, X. Teruel, X. Martorell, E. Ayguadé, J. Labarta, A. Duran, B. R. de Supinski, S. Olivier, K. Li, and A. E. Eichenberger, Towards Task-Parallel Reductions in OpenMP. Cham: Springer International Publishing, 2015, pp. 189–201.

[6] Антонов, А. С. Параллельное программирование с использованием технологии OpenMP / А. С. Антонов. – Москва: Изд-во МГУ, 2009. 78 с.

[7] Кучумов, Р.И. Реализация и анализ Work-Stealing планировщика задач // Стохастическая оптимизация в информатике. 2016. Т. 12, № 1. С. 20–39.

[8] Васёв П.А, Бахтерев М.О., Манаков Д.В., Поршнев С.В. Элементы систем научной визуализации // ГрафиКон 2022: 32-я Международная конференция по компьютерной графике и машинному зрению, 19-22 сентября 2022 г., Рязанский государственный радиотехнический университет им. В.Ф. Уткина, Рязань, Россия, С. 304–315.

[9] Логинова, И. П. Визуализация трассы параллельной программы в системе проектирования / И. П. Логинова // Информационные технологии и системы 2024 (ИТС 2024) – Information Technologies and Systems 2024 (ITS 2024): материалы междунар. науч. конф., Минск, 20 нояб. / редкол.: Л. Ю. Шилин [и др.]. – С. 89–90.

[10] Осипов, Д.Л. Программирование на C++ в среде Qt / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2026. – 816с.

[11] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объектно-ориентированного проектирования. – СПб.: Питер, 2020. – 448 с.

ORGANIZATION OF COMPLEX VISUAL ANALYSIS IN PARALLEL EXECUTION OPTIMIZATION PROCEDURES IN THE SYSTEM OF LOGICAL OPTIMIZATION OF FUNCTIONAL AND STRUCTURAL DESCRIPTIONS OF DIGITAL CIRCUITS

I. Loginova

United Institute of Informatics Problems of NAS of Belarus, irilog@email.ru

Abstract: to analyze the results of logical optimization performed using parallel computing in the system of logical optimization of functional–structural descriptions FLC-2, it is proposed to use a comprehensive approach involving visualization tools such as Gantt charts and tabular representations. The visual tabular representations obtained after completing the design procedure of optimization using parallel computing are automatically uploaded to the SQLite database and are used to evaluate the execution time and efficiency of the optimization.

Keywords: computer-aided design, parallel computing, digital circuit, diagram, model, database, Open MP.