ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ПОИСКА ОБРАЗУЮЩИХ ПОЛИНОМОВ

Мыцко Е.А., Аксенов С.В., Мальчуков А.Н.
Томский политехнический университет
634050, Россия, г. Томск, пр–т Ленина, 30
E-mail: EvgenRus70@mail.ru

Введение

При построении блоковых помехоустойчивых кодов (БЧХ-кодов, Рида-Соломона) используются полиномы Жегалкина. Полиномы участвующие в построении кода называются образующие полиномы. Выбор образующего полинома под конкретную задачу зависит от двух основных параметров блокового помехоустойчивого кода – длина блока данных m и количество исправляемых кодом независимых ошибок t. Отдельные коды, такие как коды Боуза-Чоудхори-Хоквингема (БЧХ-коды), имеют чёткие рекомендации выбора образующего полинома, но такие коды ограничены фиксированной длиной кодового слова (p – любое натуральное число) [1]. Остальные длины кодовых слов получаются путём укорачивания более длинных кодовых слов за счёт уменьшения разрядности информационного блока при сохранении количества контрольных разрядов. Укорачивание БЧХ-кодов вводит лишнюю избыточность, которая не несёт в себе дополнительных корректирующих возможностей, а лишь уменьшает эффективность кода.

Для поиска образующих полиномов, более эффективных, чем укороченные БЧХ-коды, необходимы большие вычислительные затраты. Это связано с тем, что мы знаем только необходимые начальные условия для поиска, такие как минимальные вес образующего полинома, минимальная длина кодового слова, а значит и минимальная длина образующего полинома. Для того, чтобы найти подходящий полином необходимо перебрать все полиномы, подходящие под выше озвученные необходимые условия и если среди них не найдётся полином, который пройдёт проверки на достаточные условия для образующего полинома, то поиск продолжится среди полиномов, длина которых на единицу больше, чем предыдущих. Среди достаточных условий образующего полинома для заданных значений m и t – являются два основных условия: 1) строящийся на основе этого образующего полинома код должен иметь расстояние Хэмминга не меньшее, чем d=2t+1; 2) остатки от деления всех комбинаций ошибок на образующий полином в рамках заданной t должны быть уникальными.

Для проверки на достаточные условия каждого претендента на образующий полином необходимо провести множество итерационных вычислений, что требует больших временных и вычислительных затрат. Например, для вычисления образующего полинома для n=32, t=3 на процессоре Intel XEON 5150 с частотой ядра 2.66 ГГц и объёмом оперативной памяти 1 Гб потребовалось 2 недели.

В данной работе рассматривается применение технологий параллельных вычислений для уменьшения времени, требуемого в задаче поиска образующего полинома.

Описание алгоритма поиска образующего полинома

Алгоритм поиска образующих полиномов заключается в полном переборе всех полиномов – претендентов, что является ресурсоемкой задачей. Таким образом, для увеличения быстродействия алгоритма поиска следует применить технологии параллельных вычислений.

Ниже представлен алгоритм поиска образующих полиномов [2].

Начало.

Шаг 1. Задаются m – разрядность информационного блока и t – желаемая корректирующая способность кодека.

Шаг 2. Вычисляется минимальная длина контрольного блока (старшая степень образующего полинома) по формуле для поиска «границы Хэмминга»[3]:

$$k = \left\lceil \log_2 \left(\sum_{i=0}^{t} C_n^i \right) \right\rceil,$$

где k – старшая степень образующего полинома, t – количество независимых ошибок, исправляемых кодом, n – разрядность кодового слова.

Шаг 3. Выбирается начальное значение веса полинома $w = d_m$. Минимальное расстояние кода, исправляющего t независимых ошибок, вычисляется по формуле [4]: $d_m=2t+1$.

Шаг 4. Выбирается полином из множества, состоящего из полиномов со старшей степенью, равной k, и весом, равным w.

Шаг 5. Строится помехоустойчивый полиномиальный код на основе выбранного образующего полинома и генерируются все кодовые слова.

Шаг 6. Вычисляется минимальное кодовое расстояние построенного кода (путём нахождения минимального веса среди всех кодовых слов, за исключением нулевого слова).

Шаг 7. Если вычисленное кодовое расстояние удовлетворяет заданным требованиям, то переходим на шаг 8, иначе переходим на шаг 10.

Шаг 8. Вычисляются синдромы ошибок путем деления всех комбинаций ошибок на образующий полином.

Шаг 9. Если все синдромы ошибок уникальны для построенного кода, то полином принимается в

качестве образующего полинома и переходим на шаг 11, иначе переходим на шаг 9.

Шаг 10. Если проверены все полиномы из множества (множество задано на шаге 4), то переходим на шаг 11, иначе переходим на шаг 4.

Шаг 11. Если вес w равен k, увеличиваем k на единицу и переходим на шаг 4, иначе увеличиваем вес w и переходим на шаг 4.

Шаг 12. Образующий полином сохраняется в файл.

Технологии параллельных вычислений

Одной из самых популярных технологий параллельного программирования является технология OpenMP (Open Multi-Processing) [4]. OpenMP – это набор директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью (SMP-системах).

Значительная часть функциональности OpenMP реализуется при помощи директив компилятору. Формат директивы на C/C++:

#pragma omp directive-name [опция[[,] опция]...]

Объектом действия большинства директив является один оператор или блок, перед которым расположена директива в исходном тексте программы.

Для того, чтобы замерить время выполнения блока программы используется функция **omp_get_wtime(),** возвращающая в вызвавшей нити астрономическое время в секундах (вещественное число двойной точности), прошедшее с некоторого момента в прошлом. Для задания числа потоков используется функция **omp_set_num_threads(int).** Таким образом, используя директивы и функции OpenMP можно модифицировать алгоритм поиска образующего полинома с целью увеличения быстродействия. Данную технологию можно применять к отдельным блокам, функциям, циклам программы.

Применение технологии OpenMP для поиска образующего полинома

Для поиска образующего полинома задаются входные параметры, такие как: разрядность информационного блока (m) и количество исправляемых независимых ошибок (t). Выходным значением является образующий полином в двоичном представлении. Весь алгоритм поиска можно разбить на 3 основных этапа: формирование полинома-кандитата, вычисления расстояния Хэмминга для построенного помехоустойчивого кода, проверка синдромов ошибок на уникальность.

При анализе алгоритма поиска образующего полинома было установлено, что при больших значениях входного параметра m значительное время работы алгоритма занимает вычисление расстояния Хэмминга помехоустойчивого кода, а также при увеличении значений входного параметра t увеличивается время проверки синдромов на уникальность. Таким образом, если ускорить выполнение данных этапов, то можно добиться увеличения быстродействия поиска образующего полинома.

На рисунке 1 изображен график ускорения алгоритма поиска образующего полинома относительно последовательного алгоритма при различных входных параметрах m и t. По оси X приведены значения входных параметров, по оси Y – значения коэффициентов ускорений.

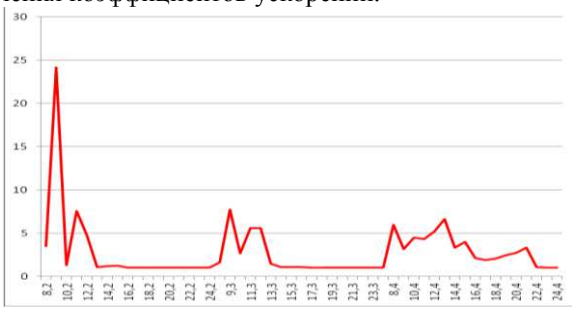

Рис. 1. График ускорений алгоритма с применением OpenMP относительно последовательного

Заключение

Используя технологии параллельных вычислений, был модифицирован последовательный алгоритм поиска образующего полинома по заданным входным параметрам: разрядность информационного блока (m) и кратность исправляемых ошибок (t). Результаты исследования быстродействия последовательного и распараллеленного алгоритмов показали, что применение технологии OpenMP увеличивает быстродействие алгоритма до 80-90% относительно последовательного варианта.

Литература

1. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования: методы, алгоритмы, применение: учебное пособие : пер. с англ. – М.: Техносфера, 2006. – 320 с.

2. Мальчуков А.Н.Алгоритм поиска образующих полиномов для системы проектирования кодеков помехоустойчивых кодов //Современные техника и технологии: Труды XIV Международной научно-практической конференции студентов, аспирантов и молодых ученых - Томск, 24-28 марта 2008. - Томск: ТПУ, 2008. - с. 340-341

3. Логвиненко Н.Ф. Оптимизация длин кодовых блоков и пакетов в системах защиты от ошибок с переспросом. // Управляющие системы и машины, 1995 – №3 – С.20-23.

4. Антонов А.С. Параллельное программирование с использованием технологии OpenMP //Издательство московского университета, 2009. – 78 С.