www.esa-conference.ru

Обратимые вычисления

Самбурский Александр Ильич, студент 2 курса ф-та ВМК
МГУ

Описание проблемы направления

Современные технологии в сфере процессорного проектирования позволяют создавать интегральные схемы с огромным числом (несколько сотен миллионов) вентилей на рабочей поверхности. За последние десятилетия рост их количества имел высокий темп, однако со временем перед инженерами встала проблема реализации архитектуры вычислительных систем. Она связана с нагревом функционального элемента в случае, когда ему на вход подаётся бóльший поток информации, чем генерируется на выходе. Каждый потерянный таким образом бит информации приводит к рассеянию тепла. Это правило основано на принципе Ландауэра. В связи с тем, что отвод тепла, необходимого для работы, неосуществим в схемах с огромным числом теряющих информацию вентилей, производство более компактных и продуктивных процессоров терпит замедление.

В связи с обозначенным вопросом, необходимо сформулировать и реализовать архитектуру компьютеров, в которой минимизировано непроизводительное выделение энергии.

Рассуждения о предлагаемых решениях

Отвод тепла, как было упомянуто, не сможет решить эту проблему ввиду недостаточной мощности. Значит, допустимое решение должно быть построено на том, чтобы максимально возможно уменьшить количество выделяющегося тепла на транзисторах. Чарльз Беннет показал, что тепло не выделяется на вентиле, только если он обратим. В таком случае информационные потери на элементах схем отсутствуют, и принцип Ландауэра не ограничивает возможности построения схем процессора. Поэтому рассматриваемая задача сводится к методам построения логических систем, обладающих функциональной обратимостью. В работе рассмотрено несколько аспектов решения поставленной проблемы, среди них разобраны вопросы возможности синтеза схем, в частности, отказоустойчивых.

Фундаментальные элементы обратимых схем

Компьютер оперирует с числами, представляющими собой двоичные наборы. В простейшем случае над этими наборами можно проводить логические операции. Как известно из курса дискретной математики, с помощью определённого набора логических функций над единичными битами, можно задать любую логическую операцию. Ввиду этого сначала рассмотрим такой «базис», состоящий из отрицания и конъюнкции, и подумаем, что нужно доработать для архитектуры с заданными требованиями. Реализацию будем демонстрировать в виде участка схемы функциональных элементов, как и работают процессоры.

Конъюнкция не обладает свойством однозначности хотя бы потому, что осуществляет отображение $\mathbf{Z}_2 \rightarrow \mathbf{Z}_1$. Есть несколько способов сделать её обратимой. Ясно, что необходимо увеличить количество выходов и сделать таким же число входов. Причём при проверке понятно, что нельзя реализовать обратимую конъюнкцию и при 2 входах и выходах, так как под её действием «0» на одном из выходов (на котором выводится конъюнкция) элемента встречается 3 раза, а «1» - только 1, следовательно, один набор встречается дважды, и отображение не обратимо.

Рассмотрим гейт размера 3*3. Пусть на входы поданы значения A, B и C. Напомним, что мы стремимся к обратимой реализации конъюнкции. Если на его выходах мы потребуем вывод значений A, B и A*B, то не сможем по выходным данным восстановить третий входной сигнал C, и эта реализация не подходит. В то же время ясно, что на выходах должны фигурировать и A, и B, и A*B, иначе гейт не будет обратимым.

В таком случае видна обоснованность перехода к контролируемым вентилям. В чём их концепция: часть входов являются контролирующими и подаются на выход без изменений. Другая часть (на которой можно реализовать интересующие нас преобразования) – контролируемые. При определённых комбинациях контролирующих входов, они, не изменяясь, подаются на соответствующие выходы гейта, а при остальных – преобразуются посредством некоторого обратимого отображения (f).

Одной из наиболее удачных и оптимальных реализаций можно назвать Вентиль Тоффоли (TG) – 3*3 обратимый гейт (рис.1). В этом гейте входы A и B – управляющие, C – управляемый. Преобразование управляемого входа происходит посредством функции f(C)=¬C. Комбинация управляемых входов, при которых это преобразование осуществляется: A=1, B=1. При других значениях – на третий выход подаётся C.

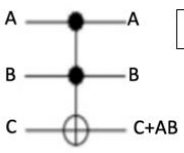

Рис. 1, TG

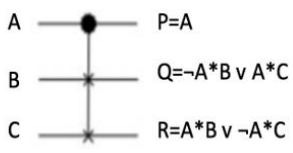

Рис. 2, FG

Другим примером является Вентиль Фредкина (FG) (рис.2). Вход A – управляющий, остальные – управляемые. При A=1 сигналы с входов B и C подаются на соответствующие выходы, поменявшись местами. Иначе - без изменений.

Технические полезные свойства Вентиля Фредкина и Тоффоли совпадают.

В чём их достоинства:

Во-первых, они самообратимы. Проверяется непосредственно. Важным следствием этого является возможность использования TG и FG для обратных вычислений без перестраивания их логики и местоположения в цепи.

ЕВРАЗИЙСКОЕ НАУЧНОЕ ОБЪЕДИНЕНИЕ
EURASIAN SCIENTIFIC ASSOCIATION
www.esa-conference.ru

Во-вторых, они логически универсальны. Вентилем Фредкина клонирование сигнала, not, and, or можно реализовать одним переключателем. nand, nor, xor и xnor - двумя. Однобитный сумматор требует четырёх вентилей. Однако это не означает оптимальность их исключительного использования.

В-третьих, эти элементы благодаря их реализации можно органично использовать при каскадном синтезе схем. Вентиль Тоффоли часто используется в вычислительных схемах, например полусумматора.

В обратимых схемах используют обобщения этих элементов с увеличенным числом контролирующих входов. Обобщённые гейты также сохраняют указанные замечательные свойства.

Помимо рассмотренных двух фундаментальных вентилей, существуют и множество других, для которых нужно определить область наиболее эффективного использования. Некоторые гейты логически универсальны и позволяют снизить сложность схемы, ряд вентилей сохраняют связь между входными и выходными данными, как сохранение чётности и веса Хэмминга, что полезно для тестирования. Некоторые используется исключительно для тестирования. Важно не столько собрать наименее затратную функциональную схему, сколько наделить её определёнными качествами. Для анализа схем и выбора наиболее оптимальных необходимо ввести некоторые числовые параметры, отображающие затраты на использования участвующих в этих схемах гейтов.

Среди этих параметров выделяют 5:

- Стоимость в гейтах (Gate cost) - число «элементарных» вентилей в схеме. К примеру, вклад в общую стоимость схемы 1*1 гейта можно взять как 1, вклад 2*2 гейта – как 2, а стоимость более сложных элементов – как суммарную стоимость композиции указанных простых гейтов;
- Квантовая стоимость (Quantum cost) – число элементарных квантовых гейтов (NOT, CNOT и т.д.) в реализации эквивалентной схемы;
- Служебные входы (Ancilla input) – число входов, не определяющих поведение схемы, на них подаются константы. Их вводят для того, чтобы схема вычисляла заданную логическую функцию;
- Число входов (Number of input);
- Мусорные выходы (Garbage output) – аналогично служебным входам мусорные выходы содержат информацию, не используемую в дальнейших вычислениях, но они необходимы для обратимости.

Для каждого гейта в отдельности можно рассматривать их собственные характеристики. Например, число реализуемых функций, максимальная глубина обратимости (MRD) – минимальное число одинаковых вентилей, соединённых последовательно для получения на выходе этой цепочки входных данных. MRD – одна из особо серьёзных характеристик эффективности использования гейтов. Чем она меньше, тем меньше будет сложность и затратность схемы. У обобщённых вентилей Тоффоли и Фредкина (MCT и MCF) эта характеристика равна 1, что делает их главными фаворитами в создании функциональных схем.

Для того, чтобы сравнить эффективность использования гейтов в синтезе, можно сконструировать сумматор и рассмотреть указанные выше свойства цепей для каждой его реализации из различных вентилей. Он взят для исследования, так как, очевидно, является одним из наиболее часто функционирующих элементов, а, значит, и в большей степени определяющих эффективность произвольной схемы. И в этом исследовании преимущество у MCT. Другие гейты показывают хорошие результаты по указанным пунктам, каждый в своей области.

В то же время, возвращаясь к возможности тестирования схем, важную роль играют консервативные (сохраняющие вес Хэмминга) и сохраняющие чётность гейты. Поэтому оставлять только MCT и MCF не всегда предусмотрительно.

Вопросы о сбоеустойчивости

Сбоеустойчивость – одно из самых важных свойств в любой вычислительной системе. Необходимо добиться этого качества в функциональных схемах. Сбои в функциональных схемах проявляются в самопроизвольном изменении рабочего сигнала (он может исчезнуть или появится, может всегда ошибочно инвертироваться или принимать постоянное значение). Этого невозможно избежать, но нужно уметь защищать вычислительный процесс от распространения ошибок. Начнём с наиболее простого способа реализовать эту идею [1, с. 7-12].

Суть реализации заключается в том, что каждый сигнал проходит схему по трём независимым каналам, и для каждой возможной операции в качестве аргументов подаются эти независимые тройки (без сбоев их значениями являются 000 для сигнала 0 или 111 для сигнала 1). Результатом являются тройки сигналов для каждого выходного значения, они получаются при обработке «среднего значения» входных данных. Для этого метода основную роль играет гейт, называемый Воутером. Он изображён на рис.3.

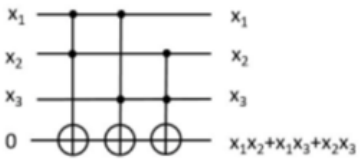

Рис. 3, Воутер

Легко проверяется, что функция «голосования» на 4 выходе Воутера позволяет определить сигнал, занимающий большинство на входах (3 или 2 из 3). Воутер «прикреплён» к каждому элементу схемы или к наиболее уязвимым из них.

Можно выделить следующие полезные свойства такого подхода:

- Каждый последующий элемент схемы можно рассматривать в качестве автоматического исправителя ошибок предыдущего элемента;
- При возникновении одного сбоя во входной тройке все выходные тройки не будут испорчены;

ЕВРАЗИЙСКОЕ НАУЧНОЕ ОБЪЕДИНЕНИЕ
EURASIAN SCIENTIFIC ASSOCIATION
www.esa-conference.ru

▪ При возникновении даже нескольких ошибок в одном из элементов, что, к слову, маловероятно, благодаря выводу сигнала через несколько каналов, в следующем элементе схемы эта ошибка исправится.

Таким образом, в условиях, когда сбои не происходят очень часто, число ошибок не увеличивается и не влияет на результат вычислений.

Но, конечно, за любое удобство приходится расплачиваться другим. Такая реализация сбоеустойчивости требует увеличение затрат на схемы. Если, к примеру, требуется максимально «защитить» сам Вентиль Тоффоли, то, как изображено на рис.4, потребуется 5*3+3=18 обычных Вентилей Тоффоли.

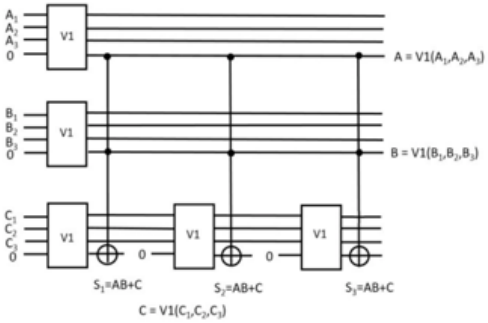

Рис. 4, Сбоеустойчивый Вентиль Тоффоли

Поэтому логичным решением при такой реализации будет защищать элементы от сбоев не «полностью». Где-то структуру сбоеустойчивости элемента лучше упростить, где-то вообще исключить. Этот вопрос требуют дополнительного исследования на предмет уязвимости используемых гейтов.

Последним замечанием к этому методу укажем следующее. В интерпретации сигналов можно сделать замену: 010 принимать за 0, а его инверсию – 101 – за 1. Расставив для каждой входной и выходной тройки на втором элементе NOT, мы сможем работать так же, как и до этого. Эта замена обоснована, так как при ней общая нагрузка на транзисторы будет более равномерна. Эта идея показывает для меня важность процедуры повсеместного проектирования. Конечно, здесь не демонстрируется связь между всеми уровнями вычислительной системы или факт нарушения общей эффективности из-за неэффективного использования одного из них. В то же время ясно, что для любой реализации нужно не просто определить её достоинства и сферу применения, но и возможности улучшения на смежных уровнях для большего выигрыша в эффективности.

Выше рассматривались принципы построения схем, способных исправлять ошибки в процессе вычисления. Как было замечено, высокий уровень исправления ошибок оборачивается серьёзным усложнением схемы, что тоже должно быть контролируемо. Рассмотрим теперь один из методов синтеза сбоеустойчивых обратимых схем, лишь обнаруживающие ошибки.

Обнаружить однократную ошибку достаточно просто: можно использовать контроль чётности. Вопрос обнаружения нескольких ошибок тоже не вызывает затруднений – можно использовать групповой код с проверкой на чётность [2]. И для тех, и для предшествующих видов сбоеустойчивости используются специальные гейты, так что на практике это, само собой, осуществимо. При фиксировании сбоя дополнительные схемы контроля должны обеспечить повтор вычислений на участке схемы, расположенном после последней успешной проверки. В качестве примера можно привести тестирующие системы, основанные на ETG-гейтах. Они основаны на том, что любой элемент схемы можно перестроить по определённому алгоритму. В результате без изменения структуры самой схемы к каждому элементу добавляется тестовый бит, по которому и можно обнаружить ошибку. В связи с этим заметно возрастает число гейтов в схеме. Подробности алгоритма перестроения можно найти в [3, с. 17].

Реализация схем, обнаруживающих ошибки, обходится дешевле схем, способных исправлять их. С другой стороны, при отсутствии исправления, необходимо быть готовым производить часть вычислений заново. Поэтому логично реализовывать схемы, обнаруживающие ошибки, на простых наборах операций, а исправляющие – в более сложных вычислениях. Переход между этими секциями легко организуем, поэтому возможен комбинированный подход реализации схемы.

Алгоритмы синтеза схем

Процесс обратимых вычислений необходимо настраивать и поддерживать на каждом уровне вычислительной системы, от физической реализации до алгоритмических языков и концепций программирования. Нарушение обратимости, скажем, в условном операторе, меняющем в своём теле константу условия, рушит все достижения на других уровнях. Скажем пару слов о реализации обратимых схем на более высоком уровне, нежели физический.

Все функции, с которыми мы работаем – логические – принимают на вход набор чисел множества {0, 1} и выдают на выходе те же объекты. По сути, они представляют из себя многозначные биективные функции многих переменных на множестве {0, 1}. Биективность (она же и обратимость) играет важную роль в формировании представления о множестве всех встречаемых логических функций. Быть может, при определённых свойствах этих множеств, возможно теоретическая оценка эффективности композиции тех или иных функций (для нашего практического вопроса – синтеза схем).

Начнём с того, что обратимость булевых функций гарантирует по определению наличие обратных к ним. Множество всех рассматриваемых нами функций с операцией композиции – есть группа: заданная операция замкнута, аксиомы группы выполнены (ассоциативность, существование нейтрального и об-

www.esa-conference.ru

ратного элементов). При дополнительных ограничениях на функции в перспективе можно ожидать ряд полезных теоретических свойств.

Перейдём к интересному вопросу: как получить наиболее оптимальную (по числу затраченных ресурсов, таких как число гейтов) схему для любой обратимой логической функции? Какие гейты лучше включить в декомпозицию?

Отметим, что одним из условий реализации синтеза является определение библиотеки гейтов – множества вентилей, из которых и будет состоять схема; она должна являться базисом любых функциональных схем. Библиотеку выбирают, с одной стороны, компактной, а с другой стороны – удобной. Задачей инженера является определение наиболее подходящей для конкретных целей.

Теория линейной алгебры может предложить способы декомпозиции логической функции в функциональную схему. Рассмотрим один из них (на рис.5).

Множество всех обратимых логических функций с одинаковым числом аргументов с введённой операцией XOR есть группа (**G**). В этой группе присутствует нормальный делитель (обозначим его как **H**). Из определения имеем, что для любого элемента *a* из **G** справедливо представление $a=b_1*a*b_2$, где b_1 и b_2 – некоторые элементы **H**. При произвольных значениях множителей правой части можно получить любой элемент **G** в левой. Более того, достаточно взять в правой части в качестве *a* один любой элемент группы **G** и, меняя b_1 и b_2, получить любой элемент группы **G**. На этом можно построить алгоритм декомпозиции.

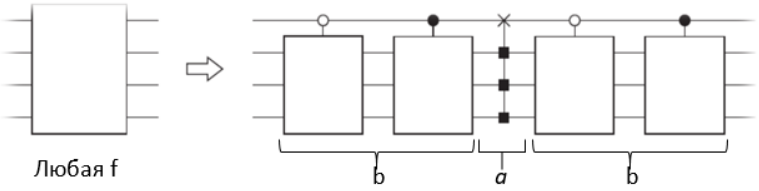

Рис. 5, Первая итерация декомпозиции схемы в гейты

На рис.5, элемент b1 осуществляет преобразование, не изменяющее первый входной сигнал. При его работе выполняется один логический блок, входящих в его состав. Определив компонент a и рекурсивно раскладывая каждый блок с таким, можно получить реализацию любой обратимой функции. Такая итерационная декомпозиция допустима, так как логическая интерпретация каждого из блоков есть элемент группы всех обратимых логических функций с количеством входов, на 1 меньшего, чем в исходной схеме, и там так же найдётся нормальный делитель. При таком подходе в библиотеку базисных элементов на группу обратимых функций с w входами (всего их (2w)!) включаются только 2w.

Подобным образом можно описать, пожалуй, один из наиболее оптимальных (с точки зрения стоимости получаемой схемы в гейтах) алгоритм декомпозиции. Его результатом является функциональная схема, полученная из логического блока (рис.6), функциональность которого можно проанализировать по его таблице истинности.

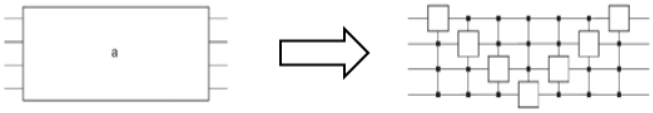

Рис. 6, Схема работы алгоритма декомпозиции

Алгоритму для работы необходима только таблица истинности функции. При работе она постепенно расширяется, это позволяет представить исходную логическую функцию как композицию более простых функций, каждую из которых можно реализовать с помощью одного обратимого гейта. Их таблицами истинности являются пары смежных столбцов расширенной таблицы истинности исходной функции. В итоге получается схема, схематично изображённая на рис.6 справа. Произвольная обратимая логическая функция с w входами представима гейтами простой структуры, количество которых в среднем зависит от количества входов как (2*w-1). Это невероятный результат. Доказан следующий факт: алгоритмы декомпозиции не могут показать результат в стоимости в гейтах полученной схемы лучше, чем (2*w-3) в общем случае. Поэтому этот алгоритм является очень близким к оптимальному. Подробное описание алгоритма приведено в [4, с. 58-61]. Таким образом, можно заменить нетривиальное преобразование исходной функции на композицию множества простых преобразований, каждое из которых реализуемо обобщённым элементом CNOT. На этом основан один из наиболее оптимальных способов синтеза схем.

Подведём небольшие итоги. В работе рассмотрены важные стороны проектирования обратимых интегральных схем, приведены алгоритмы их синтеза, обозначены их достоинства. В дальнейшем для определения эффективности какого-либо способа синтеза будет создан программный продукт, строящий схемы разными методами и выявляющий наиболее подходящий для конкретных целей.

Литература:

1. Кормаков Г. В., Гуров С. И.: Сбоеустойчивые обратимые схемы и метод их синтеза в пространстве Хемминга, 2018. – с. 7-12.
2. Помехоустойчивое кодирование: https://ie.tusur.ru/books/COI/page_08.htm
3. Гуров С. И., Жуков А. Е., Закаблуков Д. В., Кормаков Г. В.: Обратимые вычисления, 2019. – с. 17.
4. Alexis De Vos.: Reversible Computing, 2010. – с. 58-61.