УДК 621.3.049.77:004.312

КВАЗИСАМОСИНХРОННАЯ РЕАЛИЗАЦИЯ УСТРОЙСТВА ДЕЛЕНИЯ И ИЗВЛЕЧЕНИЯ КВАДРАТНОГО КОРНЯ

Ю.А. Степченков, Ю.Г. Дьяченко, Ю.В. Рождественский, Н.В. Морозов, Д.Ю. Степченков

Аннотация

Представлены подходы к проектированию самосинхронной аппаратуры различных классов и рассмотрены условия внутрисистемной интеграции синхронных и самосинхронных устройств на примере разработки квазисамосинхронного вычислительного устройства, выполняющего функции деления и извлечения квадратного корня над числами одинарной и двойной точности в соответствии со стандартом IEEE 754.

1. Введение

Реализация высоконадежных устройств цифровой обработки данных, работоспособных в широком диапазоне условий эксплуатации, всегда являлась актуальной задачей. Постоянное усложнение задач, решаемых бортовыми комплексами мобильных и летательных аппаратов, ужесточение требований, предъявляемых к их надежностным характеристикам, заставляют искать новые архитектурные и схемотехнические решения, которые позволили бы расширить диапазон работоспособности цифровых устройств в условиях автономной работы. Одним из таких направлений, гарантирующим повышение надежности и снижение требований к условиям эксплуатации вычислительных устройств, является самосинхронная схемотехника [1, 2] – альтернатива синхронной схемотехники, обеспечивающая решение проблемы синхронизации обработки данных на основе принципов контроля окончания переходных процессов в элементах схемы.

Синхронизация – один из важнейших принципов работы цифровых систем, решающий проблему координации событий в аппаратуре и связанный, в основном, с обеспечением интерфейса между физическим и логическим (искусственным) временем.

Начиная с середины 1950-ых годов активно исследуются и развиваются два альтернативных метода синхронизации элементов в аппаратуре: синхронный (С) и самосинхронный (СС). При синхронном методе (С-методе) система абстрагируется от физического времени и связь между физическим временем и событиями в системе обеспечивается системными часами. Внешние часы, формирующие импульсы синхронизации, отделены от модели поведения системы и не имеют завершенного причинно-следственного отношения к событиям в системе.

Простота С-подхода стала причиной преобладания так называемых FS-систем (Fully Synchronous) с одним генератором (G) – см. рис. 1,а, для корректной работы

которых период синхроимпульсов выбирается из расчета на худший случай – максимально возможное время переключения элементов при неблагоприятных сочетаниях условий работы (температуры напряжения питания, и т.п.). Таким образом, цена корректной работы С-аппаратуры – недоиспользование ее возможностей по сравнению с номинально возможным быстродействием. Рис. 2 из [3] иллюстрирует временные потери, типичные для С-системы.

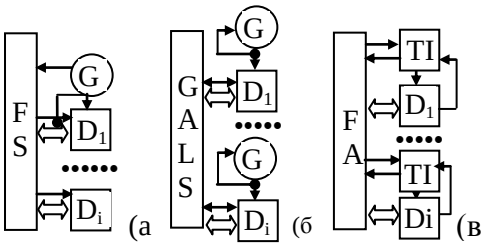

Рис. 1. Типы организации синхронизации

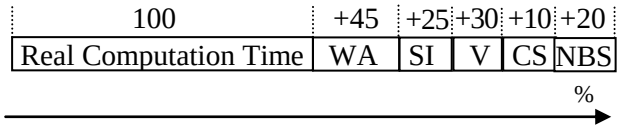

Рис. 2. Временные потери в синхронном ВУ

Вынужденная ориентация на худший случай работы WA (Worst Average), устранение перекоса сигналов CS (Clock Skew), потери из-за несбалансированности ступеней конвейера NBS (Non Balanced Stages), учет разброса параметров технологии V (Variability) и необходимость сохранения правильной формы сигналов SI (Signal Integrity) вынуждает ухудшать быстродействие С-аппаратуры в 2.3 раза для обеспечения правильной работы в узком диапазоне напряжений (± 10% от номинала).

По мере уменьшения проектных норм, увеличения степени интеграции и повышения быстродействия недостатки С-подхода становятся все более очевидными: система доставки синхроимпульсов к их приемникам становится менее эффективной. Отсюда актуален переход к GALS-системам (Global Asynchronous/Local Synchronous), которые переводят решение проблем синхронизации устройств (D, Device) в локальные области (см. рис. 1,б).

Альтернатива FS-подходу – полностью асинхронный (FA, Fully Asynchronous), где внешние часы заменяются причинно-следственными связями между событиями (см. рис. 1,в). Связь между логическим и физическим временами (TI-интерфейс, Timing Interface) обеспечивается механизмами индикации завершения переходных процессов в элементах системы. Отказ от использования часов требует решения ряда проблем высокой сложности и, в том числе, необходимости согласования СС-аппаратуры с С-окружением.

В работе [4] предложена методология GALA-систем (Global Asynchronous / Local Arbitrary) где проектировщик может использовать широкую гамму решений – от реализации полной самосинхронности до введения параллельной задержки и использования локального генератора. Все зависит от целей проектируемого изделия.

Несмотря на теоретически доказанные и практически подтвержденные преимущества СС-схем, до сих пор преобладают С-системы. В основном это связано с тем, что изначальная ориентация СС-метода на динамическую (поведенческую) модель схемы предопределяет б***о***льшую трудоемкость проектирования по сравнению со статической моделью аналогичной синхронной схемы.

В настоящее время ситуация кардинально меняется. В частности, переход к субмикронной технологии и необходимость ориентации перспективных САПР СБИС на поведенческий стиль проектирования приводит к выравниванию сложности проектирования

всех типов СБИС и необходимости использования моделей и результатов, полученных для СС-схем, в САПР СБИС общего назначения. Основной аргумент, предопределивший в середине 50-годов ориентацию на С-метод проектирования, перестал действовать.

Естественно, эволюционный путь развития С-схемотехники еще не исчерпан. Инженерная смекалка и современные технологические возможности позволяют решать текущие схемотехнические проблемы в рамках С-метода. Однако накопленный опыт теоретических и практических изысканий в области СС-метода позволяет утверждать, что современные проблемы разработки СБИС-систем могут быть решены более эффективно при переходе на синхронно-самосинхронный стиль проектирования.

Основным преимуществом СС-схем [1, 2] является стабильная работа при любых конечных задержках элементов. Это позволяет им работать при напряжениях питания и условиях окружающей среды, близких к граничным, определяемым физическими свойствами полупроводниковых активных элементов – транзисторов, если это не критично с точки зрения быстродействия устройства и всей системы в целом. Платой за это является увеличение аппаратурных затрат – количества транзисторов в схеме. Однако это усложнение нивелируется отсутствием схемы синхронизации и в ряде случаев оказывается незначительным, например, при реализации последовательностных устройств. Кроме того, достигаемое при этом расширение области работоспособности, в первую очередь, по напряжению питания, является достаточной компенсацией усложнения схемы в устройствах, предназначенных для работы с источниками питания с ограниченными ресурсами.

В данной статье представлена разработка квазисамосинхронного вычислительного устройства (в дальнейшем – ВУ), выполняющего функции деления и извлечения квадратного корня и реализованного по КМОП технологии с проектными нормами 0.18 мкм. Обрабатываемыми операндами служат числа одинарной и двойной точности в соответствии со стандартом IEEE 754 [5] с некоторыми упрощениями:

- на вход поступают только нормализованные операнды,
- результатом является также нормализованное число,
- результат, не представимый в виде нормализованного числа, вызывает исключительную ситуацию.

В настоящее время известно множество алгоритмов вычисления частного от деления и корня квадратного [6-9]. Однако не все они могут быть эффективно реализованы в виде СС-схемы. Особенность СС-схем – запрос-ответное взаимодействие между соседними в цепочке обработки информации блоками, – делает наиболее эффективной многостадийную реализацию любого вычислительного алгоритма, когда управление передается от предыдущей стадии последующей асинхронно, в соответствии с реальным быстродействием каждой стадии. Деление и извлечение квадратного корня не

являются в этом отношении исключением. Особенность предлагаемого решения – совмещение в одном устройстве функций деления и извлечения квадратного корня.

2. Алгоритм деления и извлечения корня

В соответствии со стандартом IEEE 754 обрабатываемые операнды имеют формат числа с плавающей точкой и состоят из бита знака, поля порядка и поля мантиссы. Длины полей порядка и мантиссы зависят от типа представления обрабатываемого операнда: одинарной или двойной точности. Алгоритм вычисления одинаков для обеих точностей: мантиссы и порядка обрабатываются отдельно. Вычисление порядка результата сводится к вычитанию порядка делителя из порядка делимого при выполнении деления или к сдвигу в сторону младших разрядов при извлечении квадратного корня. Основную же сложность представляет собой вычисление мантиссы результата. В дальнейшем под вычислением результата мы будем понимать вычисление мантиссы.

Современные быстрые алгоритмы деления используют оценку ожидаемого значения частичного результата (одного или нескольких битов мантиссы результата), начиная со старших разрядов, на основе значений нескольких старших разрядов промежуточных данных. Среди множества алгоритмов деления можно выделить два основных класса: безвозвратный и с возвратами в случае ошибочного предсказания частичного результата. Алгоритмы первого класса предусматривают возможность коррекции накапливаемого результата вычисления на последующих шагах (итерациях), если оценка на текущем шаге оказалась неточной и остаток от деления оказался отрицательным. Алгоритмы второго класса требуют получения корректной оценки на каждом шаге и в случае обнаружения ее некорректности возвращаются к ее вычислению, уточняя до тех пор, пока получаемый промежуточный результат (остаток от деления) не будет неотрицательным. Наиболее быстродействующие алгоритмы основаны на использовании умножения, например, метод Ньютона-Рефсона (Newton-Raphson), или табличных методов. Однако они требуют реализации аппаратного умножителя или относительно больших объемов статической памяти или ПЗУ, которые при реализации в СС-базисе приводят к значительным аппаратурным затратам и оказываются не столь эффективными. Поэтому в качестве базового был выбран безвозвратный алгоритм деления.

Он как бы реализует традиционное деление "в столбик" и основан на использовании канонической формулы:

$$P_{i+1} = r \cdot P_i - D \cdot q_i, \qquad i=0,\ldots n-1, \tag{1}$$

где P_i, P_{i+1} – промежуточные остатки от деления на i-ом и (i+1)-ом шагах алгоритма; r – основание алгоритма (radix); D – делитель; q_i – полученный на i-ом шаге

частичный результат; n – число шагов алгоритма, зависящее от размерности обрабатываемых операндов N и основания алгоритма: $n=]N/(2^{r-1})[$. В качестве P_0 используется исходное делимое. Окончательный результат формируется из частичных результатов путем их конкатенации:

$$Q = \{q_0 q_1 q_2 \ldots q_{n-1}\}.$$

Основание алгоритма r может принимать значения из двоичного ряда {1, 2, 4, 8,…}. Чем больше основание алгоритма, тем за меньшее число шагов формируется результат операции. При этом частичный результат содержит один или несколько битов окончательного результата.

SRT алгоритм, названный так по начальным буквам фамилий его разработчиков (Sweeney, Robertson, Tocher), предложивших его независимо друг от друга [10], ускоряет вычисление частного от деления, вынося часть операций за пределы основного цикла, но вносит одновременно некоторую избыточность в массив формируемых и обрабатываемых промежуточных операндов.

В настоящее время существуют реализации SRT алгоритма для различных оснований, вплоть до r = 512. Однако в нашем случае, для реализации блока вычисления частного и квадратного корня, наиболее целесообразным является SRT алгоритм с основанием r = 2, поскольку он позволяет объединить функции деления и извлечения квадратного корня в одном блоке за счет минимальных дополнительных затрат [11, 12], что является немаловажным фактором при реализации цифровых устройств в СС-базисе.

Действительно, каноническая формула для вычисления квадратного корня в безвозвратном алгоритме с основанием r = 2 очень похожа на формулу для операции деления и выглядит следующим образом:

$$P_{i+1} = 2 \cdot P_i - (2 \cdot Q_{i-1} + q_i \cdot 2^{-i}) \cdot q_i, \qquad (2)$$

где Q_{i-1} – накопленный к i-ому шагу алгоритма результат извлечения корня. В качестве P_0 используется исходное подкоренное выражение, а $Q_{-1}=0$. Таким образом, формулы (1) и (2) отличаются только тем, что в случае деления из удвоенного остатка вычитается взвешенный частичным результатом q_i делитель, а при вычислении квадратного корня – вспомогательный операнд, определяемый "на лету" из накопленного к текущему шагу алгоритма результата Q_{i-1}. При этом результат накапливается в обоих случаях одинаковым образом.

Свойство самосинхронности ВУ придают парафазная дисциплина кодирования всех информационных сигналов, индицирование моментов окончания всех переходных процессов и организация запрос-ответного взаимодействия между соседними блоками в ВУ. Аналогичные средства уже использовались в работах [11], [12]. Но они не удовлетворяют требованиям строгой самосинхронности; заложенные в них схемы запрос-

ответного взаимодействия не рассчитаны на возможность появления большого разброса в задержках однотипных элементов из-за деградации характеристик активных структур или локальных отклонений технологических параметров.

Ниже раскрываются особенности реализации описанного алгоритма. Для простоты изложения формулы и функции приводятся в нотации для инфазного (однобитного) представления информационных сигналов. Парафазная реализация формул и функций получается путем применения известных правил кодирования [2].

3. Функциональная реализация ВУ

Структурная схема ВУ, реализующего один шаг алгоритма совмещенного вычисления частного от деления и корня квадратного (см. рис. 3) содержит следующие блоки:

БНР – блок накопления результата,

МВ – мультиплексор вычитаемого (второго члена формул (1) и (2)),

ССП1, ССП2 – сумматоры с сохранением переноса, первый из которых реализует формулу (1) для случая $q_i = -1$, а второй – для случая $q_i = +1$,

ВЧР – выбор частичного результата q_i,

МС – мультиплексор суммы,

МП – мультиплексор переноса.

Блок БНР формирует "на лету" одновременно как сам результат вычисления Q, так и его дополнение R, равное результату минус единица текущего разряда. Это позволяет при значении частичного результата q = –1 просто подменять текущий результат Q на R, не выполняя вычитания, в соответствии с формулами:

$$Q_i = (1-|q|)\cdot Q_i + \frac{(|q|-q)}{2}\cdot(R_i + P_i) + \frac{(|q|+q)}{2}\cdot(Q_i + P_i),$$

$$R_i = (1-|q|)\cdot(R_i + P_i) + \frac{(|q|-q)}{2}\cdot R_i + \frac{(|q|+q)}{2}\cdot Q_i,$$

где Q_i, R_i – i-ый бит результата и его дополнения; q – частичный результат, полученный на предыдущем шаге и принимающий одно из значений {–1,0,+1}; P_i – флаг активности разряда; i=0,..,55 – номер бита результата.

Флаг активности разряда P_i равен 1, если его индекс i соответствует номеру текущего формируемого бита результата. В остальных случаях он равен нулю. Результат формируется побитно, начиная со старшего разряда. Начальные значения операндов Q и R – нулевые. Как видно из схемы на рис. 3, выходы блока БНР используются только на следующем шаге алгоритма. Следовательно, блок БНР не входит в критический путь данной стадии, если его собственная задержка не превышает задержки критического пути.

Мультиплексор вычитаемого формирует второй член формул (1) и (2) на основе

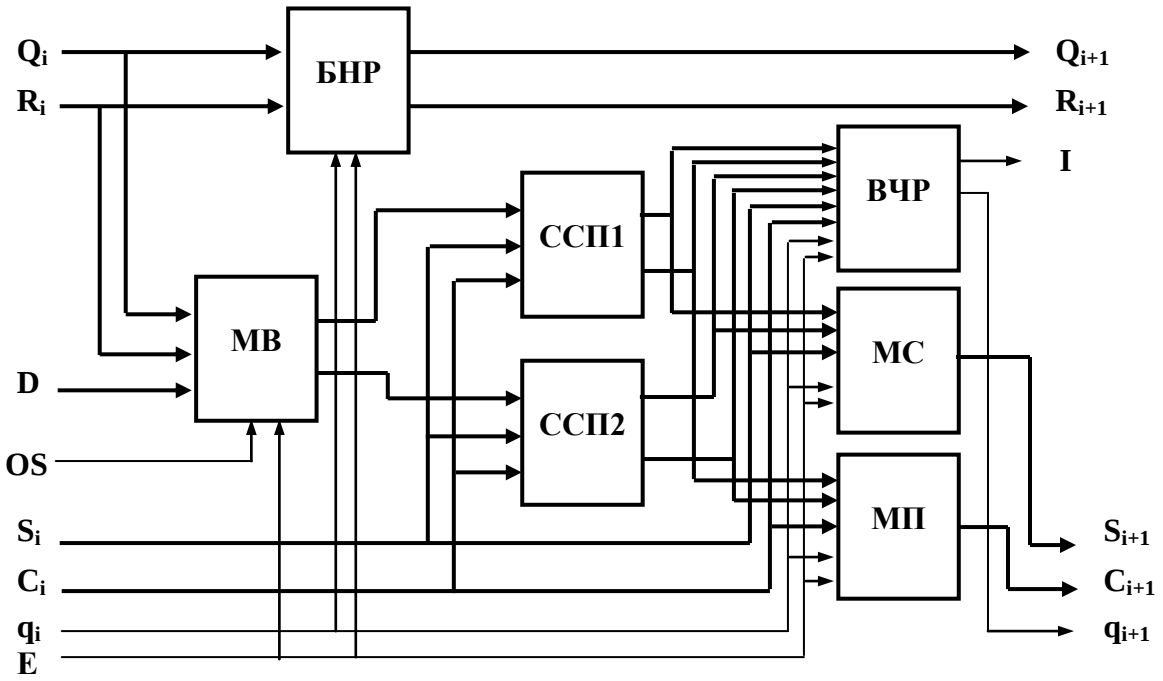

Рис. 3. Структурная схема, реализующая один шаг алгоритма

накопленного промежуточного результата Q и R или входного операнда – делителя D. Поскольку вычитание выполняется в дополнительном коде как суммирование с инверсным представлением операнда с добавлением единицы в младший разряд, делитель представлен одновременно своим исходным и инверсным значениями.

Вычитаемое для случая q = –1 определяется по формуле:

$$A_i = (R_i + P_i + P_{i-1}) \cdot \overline{OS} + D_i \cdot OS,$$

где P_i, P_{i-1} – флаги активности текущего и предыдущего разрядов; OS – признак выполняемой операции – деление (OS=1) или извлечение квадратного корня (OS=0); D_i – i-ый бит делителя. Вычитаемое для случая q=+1 формируется в соответствии с формулой:

$$S_i = (\overline{Q_i} \cdot \overline{P_i}) \cdot \overline{OS} + \overline{D_i} \cdot OS.$$

Сумматоры с сохранением переноса работают параллельно, хотя на каждом шаге алгоритма фактически требуется не более одного из них. Сигнал переноса, формируемый в каждом разряде, передается в соседний по старшинству разряд сумматора не текущей стадии, а следующей. Информация при этом не теряется за счет хранения результата суммирования в избыточном представлении: и суммы, и переноса для каждого разряда. Для получения безызбыточного представления такой суммы требуется использовать сумматор с распространением переноса. В данном алгоритме это делается в конце циклических вычислений для формирования итогового остатка и округления результата.

Использование двух сумматоров с сохранением переноса дополнительно повышает быстродействие алгоритма, уменьшая количество каскадов на критическом пути обработки

данных; иначе пришлось бы добавить перед сумматором мультиплексор для выбора нужного вычитаемого в соответствии со значением частичного результата на предыдущем шаге алгоритма.

Анализ показывает, что наибольшей задержкой формирования результата обладает блок выбора частичного результата, так как его критический путь включает трехразрядный сумматор с распространением переноса и мультиплексор с выходной логикой. Поэтому частичный результат q_{i+1} формируется последним из всех выходных сигналов, показанных на рис.3, и к моменту его формирования остальные входные сигналы для блоков БНР, МС, МП и ВЧР будут уже готовы. Тем самым обеспечивается минимальное время выполнения одного шага алгоритма при многостадийной организации вычислений.

Блок ВЧР принимает решение на основе значений четырех старших разрядов остатка. Этого достаточно для эффективной сходимости SRT алгоритма с основанием 2. При этом три старших разряда формируют операнды для трехразрядных сумматоров с распространением переноса, а самый младший из четырех старших служит только для своеобразного округления получающегося результата предварительной оценки. В итоге формируются три предварительные оценки, выбор между которыми осуществляется мультиплексором 3:1 на основе значения частичного результата q_i, полученного на предыдущем шаге, как показано на рис.4. Сумматоры предвычисляют остаток в старших разрядах на текущем шаге алгоритма для различных значений q_i заранее, минимизируя тем самым общую длину критического пути алгоритма.

Таким образом, блоки МВ, ССП1 и ССП2 и сумматоры блока ВЧР осуществляют

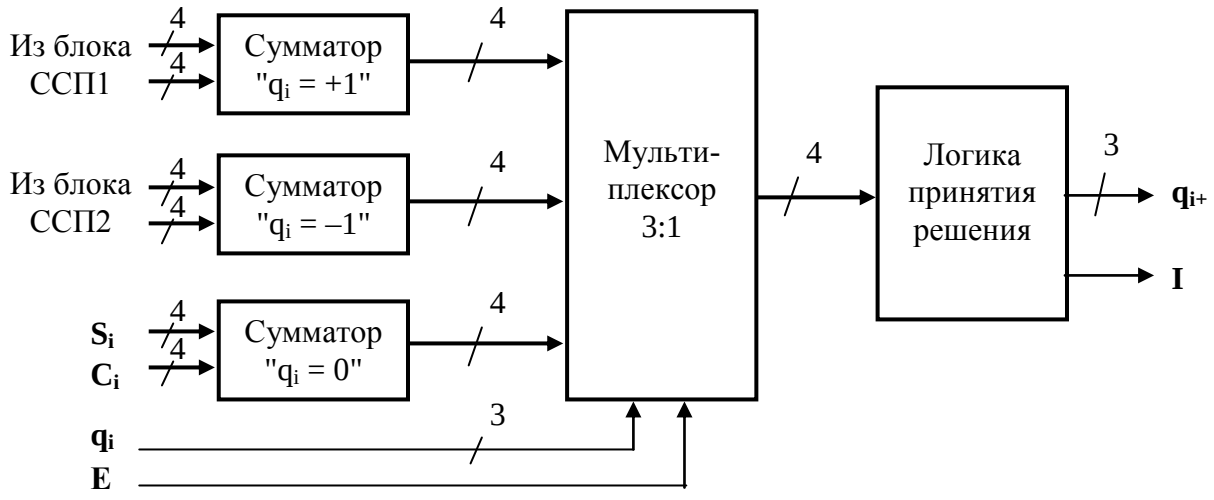

Рис. 4. Структурная схема блока ВЧР

предвычисление вспомогательных данных для принятия решения о выборе очередного частичного результата q_{i+1}, что позволяет существенно ускорить работу алгоритма при его многостадийной, конвейерной реализации.

Преимущества СС-устройств проявляются при их конвейерной реализации, причем при числе ступеней конвейера не менее трех. Это обусловлено регламентом их работы:

● наличием двух фаз работы любого СС-устройства – рабочей и спейсерной,

● использованием запрос-ответных отношений между соседними в тракте обработки данных устройствами.

Первое условие необходимо для реализации контроля за окончанием переходных процессов в каждом устройстве – ступени конвейера. Второе условие обеспечивает строгую последовательность переключений соседних ступеней конвейера и гарантирует достоверность данных на входе каждой ступени в любой момент времени и работоспособность СС-схемы при любых конечных задержках составляющих схему элементов.

Реализация индикации окончания переходных процессов зависит от требований, предъявляемых к разрабатываемому проекту. Если в С-конвейерах отсутствует фиксация окончания переходных процессов и в памяти (FF, Flip-Flop), и в CL (Combination Logic) – см. рис. 5,а, то в квазисамосинхронных конвейерах окончание переходных процессов в FF фиксируется, а окончание переходного процесса в CL моделируется встроенной задержкой DL (DeLay) – рис. 5,б. В СС-схемах индицируется окончание переключения каждого элемента для фиксации появления неисправности в схеме и остановки вычисления (см. рис. 5,в). Реализация этого принципа обеспечивает независимость работоспособности схемы от соотношения задержек составляющих ее элементов. На практике реализация полномасштабной индикации связана с большими накладными расходами – до 50 % от сложности самой схемы. Поэтому иногда руководствуются принципом "ожидаемого соотношения задержек элементов, реализованных на одном кристалле". В соответствии с ним до момента появления критических дефектов на кристалле можно гарантировать следующее:

● однотипные элементы в кристалле будут иметь близкие задержки переключения;

● чем больше входов у элемента, тем меньше его среднее быстродействие;

● более нагруженные цепи имеют и б***о***льшие задержки.

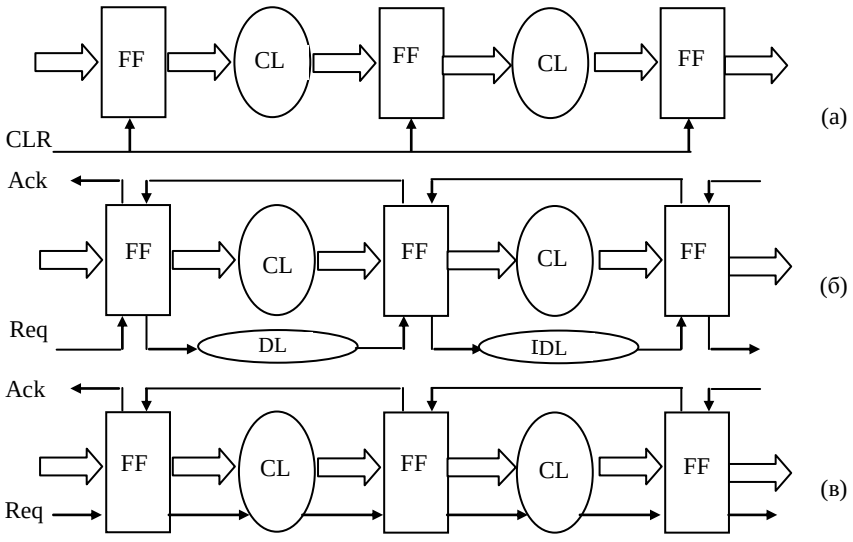

Рис. 5. Варианты реализации конвейеров

Такой подход позволяет существенно сократить затраты на индикацию схемы и ускорить ее работу. В большинстве практических случаев, не рассчитанных на работу в экстремальных условиях, этого оказывается достаточно для обеспечения бессбойной работы схемы в широком диапазоне условий эксплуатации. В этих подходах неизменным остается одно – организация запрос-ответного взаимодействия между соседними блоками.

СС-схема не может начать переключение в очередную фазу до тех пор, пока предшествующее устройство не перейдет в аналогичную фазу, а последующее устройство – в противоположную фазу работы. Чем больше ступеней конвейера, тем меньше суммарные непроизводительные затраты времени, когда каждое устройство в общей системе простаивает в ожидании разрешения перехода в следующую фазу работы от соседних устройств. Практически данное взаимодействие реализуется с помощью гистерезисных триггеров (Г-триггеров) [1] или С-элементов [2], входами которых являются индикаторные выходы предшествующего и последующего устройств, фиксирующие окончание переключения соответствующего устройства в очередную фазу работы.

С учетом особенностей функционирования ступеней конвейера и базиса их реализации в качестве входов индикаторных элементов могут использоваться и входные управляющие сигналы. На рис. 6 показана схема взаимодействия трех соседних стадий в СС-конвейере описываемого ВУ на базе использования Г-триггера. Управляющий сигнал E для каждой стадии формируется Г-триггером на основе индикаторного выхода I последующей стадии и управляющего сигнала предшествующей стадии. Он разрешает переключение стадии в соответствующую фазу работы. Это до некоторой степени аналог синхросигнала в традиционной С-схемотехнике. Индикаторный выход I фиксирует окончание переключения стадии в соответствующую фазу работы. В результате обеспечивается строгая последовательность переключений соседних стадий, несмотря на то, что индикаторный выход стадии учитывается только в Г-триггере предыдущей стадии.

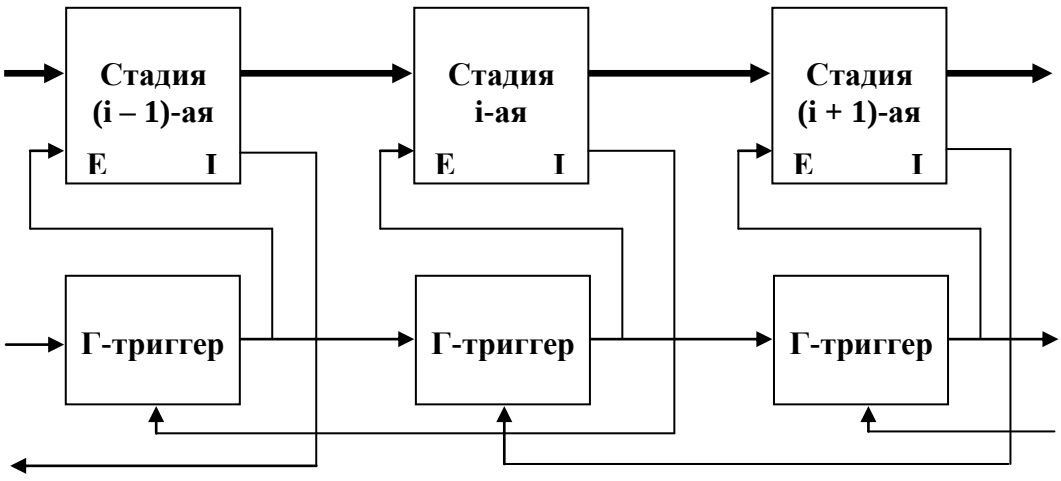

Рис. 6. Организация запрос-ответного взаимодействия между стадиями

Текущая стадия не может перейти в рабочую фазу или в спейсер, пока следующая стадия находится в аналогичной фазе. Этого не дает сделать индикаторный выход I следующей стадии. Тем самым обеспечивается безошибочность обмена данными между стадиями: пока новые данные не будут восприняты и зафиксированы следующей стадией, текущая стадия не перейдет в спейсер. По отношению к предыдущей стадии такой строгой последовательности не требуется. Текущая стадия может начать переход в рабочую фазу одновременно с предшествующей, поскольку завершить этот переход она сможет не раньше, чем на выходах предыдущей стадии появится новое рабочее состояние. Следовательно, последовательность переключения стадий будет соблюдена.

Из-за ограничений по площади топологической реализации (со стороны заказчика) было принято решение ограничиться частичной реализацией принципа самосинхронности в ВУ. Однако использованные принципы взаимодействия блоков ВУ и псевдодинамические элементы позволили обеспечить строгую последовательность переключения ступеней конвейера.

Одна из проблем реализации многостадийных вычислений – хранение промежуточного результата. В синхронных реализациях она традиционно решается с помощью регистров, разделяющих соседние стадии. В СС-схемах все стадии работают автономно, переключаясь по своим собственным задержкам, зависящим от типа обрабатываемых данных. Поэтому соседние стадии могут иметь разные задержки переключения. Но моменты окончания этих переключений индицируются, сигнализируя о готовности результата на выходе каждой стадии. Этого достаточно для организации четкого обмена данными между стадиями. Проблема хранения промежуточного результата может быть решена за счет использования элементов с памятью. В последнем случае для реализации ВУ эффективным оказывается применение элементов с динамической логикой [11]. Парафазная дисциплина информационных сигналов упрощает перевод таких элементов в спейсер и предохраняет от появления на выходе ложных кратковременных рабочих состояний при переключении в рабочую фазу.

Количество стадий в конвейерной реализации алгоритма может быть произвольным, но практически оно ограничено габаритами топологической реализации и мощностью потребления. Теоретический анализ показывает [11], что оптимальное (с точки зрения быстродействия) число стадий в таком алгоритме равно 4 или 5. Именно в этом случае конвейер оказывается сбалансированным наилучшим образом.

Исходя из указанных предпосылок, заданных габаритных ограничений и требований к быстродействию ВУ, был выбран четырехстадийный вариант его реализации. Каждая стадия реализует ступень вычислительного конвейера, структурная схема которой показана на рис.3. Общая структурная схема ВУ представлена на рис.7.

Помимо четырех однотипных стадий С1 – С4, она содержит следующие блоки:

- входной регистр операндов и признаков операции (ВРО),
- входной мультиплексор операндов (МО),
- индикаторную схему (ИС), включающую и схему управления (СУ),
- схему обработки экспонент (СОЭ),
- блок постобработки мантиссы (БПО),
- выходной регистр результата (ВРР).

Размерности операндов, показанные на рис. 7, соответствуют асинхронному варианту реализации ВУ. Для СС-варианта размерности внешних сигналов сохраняются, поскольку данное ВУ предназначено для работы с синхронным окружением. Размерности же внутренних шин должны быть увеличены вдвое, так как традиционная самосинхронная реализация предполагает использование парного (парафазного) представления для информационных сигналов [1]. Это позволяет кодировать два рабочих и одно спейсерное состояние для каждого информационного сигнала. При этом спейсер может быть нулевым (00) или единичным (11), а рабочие состояния (01) и (10) кодируют соответственно низкий и высокий уровень информационного сигнала. В данной реализации ВУ используется нулевой спейсер для всех внутренних шинных сигналов.

Схема ВУ работает следующим образом. Обрабатываемые операнды OP1 и OP2 (для операции извлечения квадратного корня только OP1) и признаки операции OF (одинарная или двойная точность, тип операции, тип округления результата)

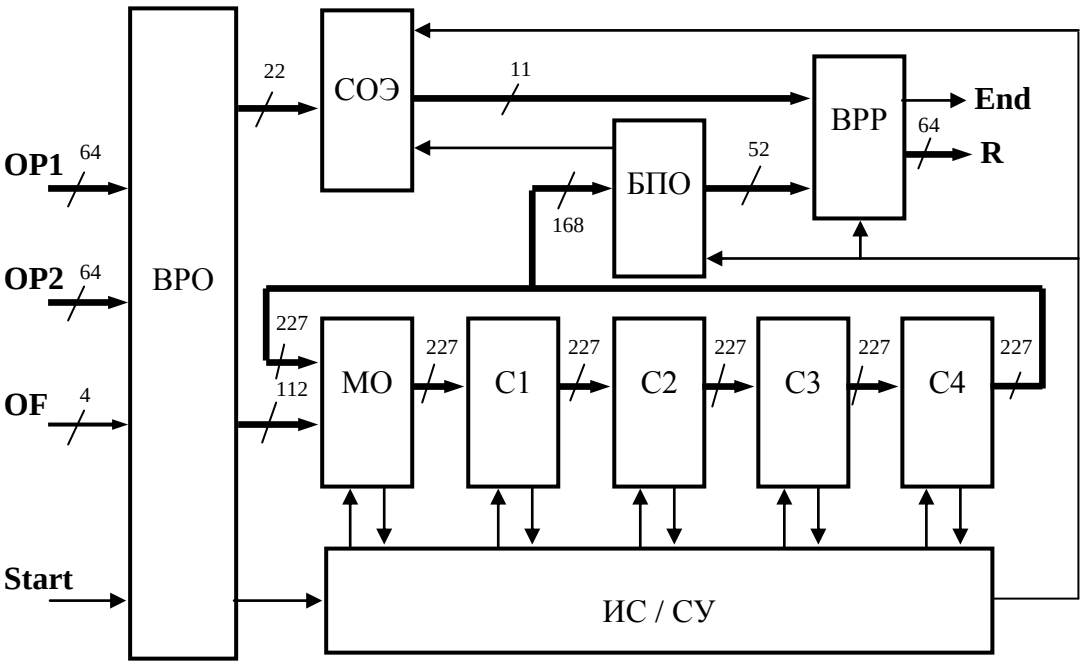

Рис. 7. Структурная схема ВУ

записываются во входной регистр ВРО синхронно по сигналу Start. Экспоненты операндов обрабатываются в блоке СОЭ. На значение результирующей экспоненты может повлиять и мантисса результата, так как стандарт IEEE 754 использует нормализованное представление для всех чисел, в котором мантисса характеризует только дробную часть числа, а обязательная целая единица присутствует неявно. Поэтому, если результат обработки мантисс при делении оказывается меньше единицы, мантисса сдвигается в сторону старших разрядов, а экспонента уменьшается на единицу.

Знаковый разряд обрабатывается в схеме управления, после чего передается в выходной регистр ВРР и в блок постобработки БПО. На вычисление экспоненты результата операции он не оказывает влияния. Но при округлении к минус или плюс бесконечности его нужно учитывать.

Мантиссы обрабатываются в кольце, включающем четыре однотипные стадии С1–С4 и входной мультиплексор, который необходим для выполнения следующих функций:

- ввода в кольцо исходных значений обрабатываемых операндов,
- начального обнуления промежуточных операндов,
- замыкания кольца.

За один цикл в кольце вычисляются четыре бита мантиссы результата. Максимальный размер мантиссы исходного операнда равен 52 битам. В процессе вычисления формируется мантисса результата, состоящая из 56 бит. Четыре младших бита мантиссы результата используются для округления и возможной нормализации. Таким образом, для вычисления мантиссы числа с двойной точностью требуются 14 циклов работы кольца. Для чисел с одинарной точностью достаточно 7 циклов, так как длина мантиссы в этом случае равна 23 и для получения правильного результата достаточно пяти дополнительных бит.

По окончании всех циклов мантисса и остаток передаются в блок постобработки БПО, который выполняет функции:

- вычисления окончательного остатка на основе его избыточного представления,
- округления мантиссы по четырем младшим битам,
- нормализации результата (при необходимости).

Сформированные мантисса и экспонента результата фиксируются в выходном регистре ВРР. С-окружение оповещается о готовности результата установкой флага End=1.

Особенностью данного проекта является отказ от использования промежуточных регистров внутри ВУ. Это оказалось возможным благодаря применению псевдодинамических элементов с памятью на слабых транзисторах.

Высокую эффективность реализации многостадийного ВУ обеспечивают

псевдодинамические элементы. Пример псевдодинамического элемента, реализующий функцию мультиплексора 2:1, приведен на рис.8. Он имеет парафазные информационные входы {A, AB}, {B, BB}, инфазные сигналы выборки EA, EB, инфазный управляющий сигнал и парафазный выход{Y, YB}. Переход в спейсер обеспечивается подачей входного управляющего сигнала E=0. В этом случае N-часть схемы отключается от выходов и последние переключаются в состояние Y=YB=0. За счет использования слабых транзисторов WT0 и WT1 выходы могут неограниченно долго поддерживать спейсерное состояние после перевода управляющего сигнала в состояние E=1, до тех пор, пока на информационные входы и на входы выборки не придет рабочая комбинация, переводящая один из выходов мультиплексора в состояние логической 1.

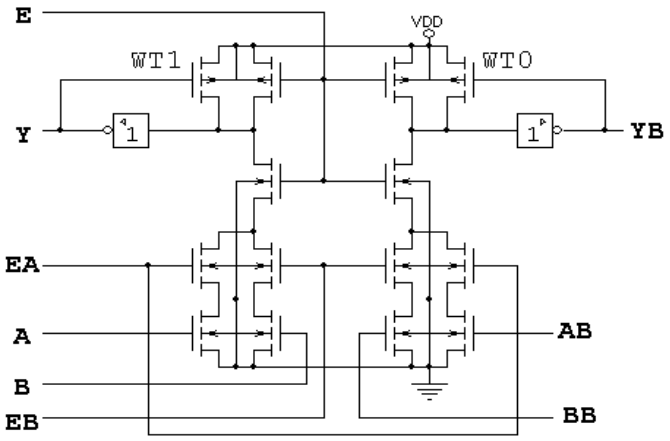

Рис. 8. Принципиальная схема псевдодинамического мультиплексора

Хранение рабочего состояния на выходах не поддерживается, но оно соответствует нулевому потенциалу на входах инверторов, формирующих выходы мультиплексора, и за счет паразитных емкостей может храниться достаточно долго, поскольку утечка заряда в данном случае не страшна. Кроме того, работа схемы организована таким образом, что управляющий вход переключается в рабочее состояние E=1 незадолго до формирования рабочей комбинации на прочих входах.

В наиболее критичных, с точки зрения помехоустойчивости, узлах схемы (блоках БНР, МС и МП на рис.3, мультиплексоре блока ВЧР, входном мультиплексоре МО на рис.7), формирующих информационные сигналы, которыми обмениваются между собой стадии ВУ, используются элементы с памятью на слабых инверторах. Пример такого элемента, реализующего мультиплексор 3:1, показан на рис.9. Он также имеет парафазные информационные входы {A, AB}, {B, BB}, {C, CB}, парафазный выход {Y, YB}, инфазные сигналы выборки EA, EB, EC и управляющий сигнал E. Слабые инверторы D0 и D1 обеспечивают надежное хранение на выходах {Y, YB} информации.

Наличие управляющего сигнала E и минимальное количество последовательно

соединенных транзисторов в схемах элементов на рис. 8 и 9 ускоряет их переключение в спейсер, повышая тем самым быстродействие и всего ВУ. Электрическое моделирование показывает, что средняя задержка переключения между фазами работы у такого элемента меньше, чем у его аналога, реализованного стандартной КМОП схемой без слабых инверторов.

СС-взаимодействие между соседними стадиями реализовано в полном соответствии с принципами построения СС-схем и с учетом особенностей работы применяемых элементов с памятью. При этом входной мультиплексор также включен в СС-конвейер. Окончание циклов вычисления фиксируется схемой управления, после чего результат с выходов последней стадии подается в блок постобработки.

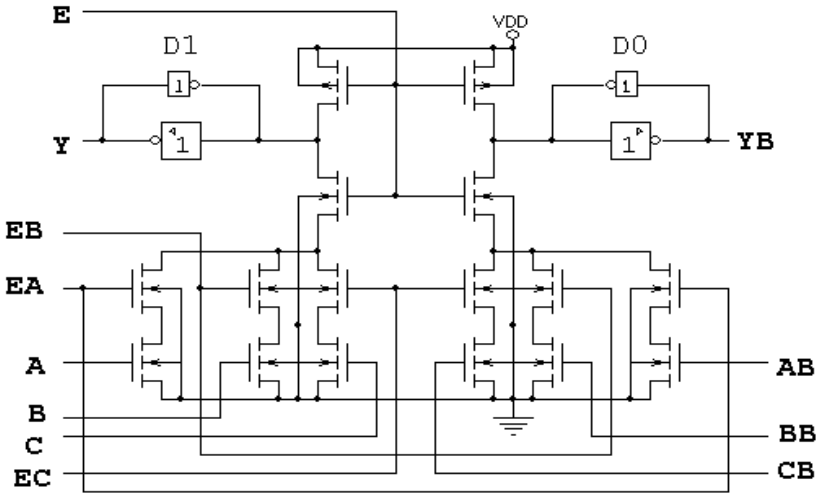

Рис. 9. Принципиальная схема мультиплексора 3:1 с памятью

4. Практическая реализация ВУ

Разработка ВУ состояла в последовательном выполнении стадий традиционного нисходящего маршрута проектирования БИС на основе библиотеки стандартных элементов фирмы Artisan [13]. Однако для реализации элементов с памятью и псевдодинамических элементов готовых решений в составе библиотеки не оказалось. Поэтому маршрут проектирования ВУ включал в себя следующие стадии:

- разработку и отладку алгоритма работы ВУ в соответствии со стандартом IEEE 754;
- функционально-логическое проектирование ВУ, определение набора необходимых дополнительных библиотечных элементов;
- разработку схемотехнических и топологических представлений новых библиотечных элементов;
- характеризацию новых библиотечных элементов и составление их логических моделей на языке Verilog;
- логическое моделирование ВУ и верификацию его функционирования работы на

статистически полном наборе примеров, с использованием системы Modelsim;

● разработку и верификацию топологии ВУ;

● вторичное моделирование ВУ с учетом извлеченных из топологии паразитных емкостей и резисторов с помощью программы Ultrasim;

● схемотехническую и топологическую оптимизацию проекта ВУ по результатам вторичного моделирования.

Отладка алгоритма работы ВУ проводилась с использованием тестовой программы, разработанной в Институте системного программирования РАН. Эта программа позволяет проверить работу алгоритма на произвольных сочетаниях обрабатываемых операндов, в сложных случаях округления и при прочих особенностях реализации стандарта IEEE 754.

Для реализации ВУ потребовалось разработать дополнительно 27 элементов – псевдодинамических и с памятью. Они реализованы топологически в стиле стандартных библиотечных элементов фирмы Artisan [13]. Для включения новых элементов в систему логического моделирования была проведена их характеризация.

Характеризация необходима для извлечения временных и энергетических параметров элементов – характеристик, которые используются системами логического моделирования. В рамках данной работы характеризация проводилась только по задержкам в зависимости от напряжения питания $U_{ип}$ и температуры T при трех условиях функционирования: типовых (Uип=1.8 В, T=25^0C), наихудших (Uип=1.62 В, T= 125^0C) и наилучших (Uип=1.98 В, T=–60^0C).

Характеризация элементов осуществлялась с помощью программы SignalStorm, обеспечивающей автоматическую характеризацию с погрешностью получаемых параметров не более 5%, что является приемлемым для первичного моделирования без учета паразитных параметров. Часть элементов характеризовалась в автоматическом режиме. Другую часть по различным причинам не удалось характеризовать автоматически. Основные причины этого:

● не проходит этап построения векторов с сообщением “illegal net” или “loop node”;

● не проходит этап электрического моделирования;

● этап выдачи результата завершается аварийно с сообщением “can’t write output function”, иногда дополняемым сообщением “function too complicated”;

● этап выдачи результата проходит, однако функциональное описание элемента в alf-файле не появляется, либо имеет непонятный вид;

● этап выдачи результата заканчивается успешно, но команда получения модели элемента в виде Verilog-файла выдает сообщение “segmentation violation”;

● команда получения Verilog-модели элемента отрабатывается успешно, однако

функциональное описание элемента оказывается неправильным – отсутствует название выходного контакта в описании отдельных функций, либо указаны непонятные имена контактов, либо просто явно неверное описание.

Эти проблемы почти для всех элементов удалось решить при помощи:

- построения gate-файла, задающего структуру принципиальной схемы элемента;
- использования команды db_diff, детализирующей описание выводов элемента;
- составление функционального описания элемента в alf-файле вручную;
- ручной правки Verilog-модели.

Тем не менее, характеризацию некоторых элементов пришлось провести вручную при помощи программы электрического моделирования SPECTRE. Для верификации работы системы характеризации SignalStorm была проведена ручная характеризация некоторых простых элементов, успешно прошедших автоматическую характеризацию. Полученные обоими способами результаты оказались фактически идентичными.

Следует отметить, что при ручной характеризации можно более точно описать условия переключения для сложных элементов, поскольку задержка переключения выхода элемента зависит иногда от состояния всех его входов, а не только изменяемого и, даже при одном и том же состоянии входов, от потенциала внутренних узлов. В то же время только в рамках автоматической характеризации проще получить задержки типа setup и hold (предустановки) и зависимости.

На основе результатов характеризации были построены модели элементов, использованные при отладке аппаратной модели алгоритма делителя. Сравнительный анализ показал, что полученные таким образом модели оказались достаточно близкими по параметрам к реальным схемам. Разница в задержках между логическим моделированием (Modelsim), учитывающим параметры, полученные при характеризации, и электрическим моделированием (Ultrasim) без учета топологии оказалась менее 10%.

Описываемое ВУ было реализовано в составе микросхемы спецвычислителя по стандартной 0.18-мкм КМОП технологии с шестью слоями металлизации. В соответствии с техническим заданием площадь блока ВУ не должна была превышать 0.36 мм^2. В результате тщательного ручного топологического проектирования в САПР CADENCE блок ВУ занял 0.35 мм^2 и имеет форму прямоугольника с соотношением сторон 1.4 : 1. Общий вид топологической реализации блока делителя представлен на рис.10.

Вторичное моделирование – с реальными паразитными емкостями и резисторами, извлеченными из топологии делителя – показало, что паразитные параметры оказывают существенное влияние на временные характеристики устройства. Это объясняется большим количеством сигналов в схеме и, соответственно, плотной трассировкой в

топологической реализации делителя. Соседние трассы, идущие параллельно, оказывают сильное взаимное влияние, приводящее к увеличению задержки.

При типовых условиях задержка выполнения операции деления уменьшается до 50.3 нс. Для асинхронных и СС-схем тактовая частота значения не имеет. Поэтому не требуется рассчитывать заранее на наихудший случай. В реальных условиях эксплуатации наихудший случай получается далеко не всегда, поэтому оценка быстродействия СС-схем рассчитывается обычно по типовым условиям эксплуатации.

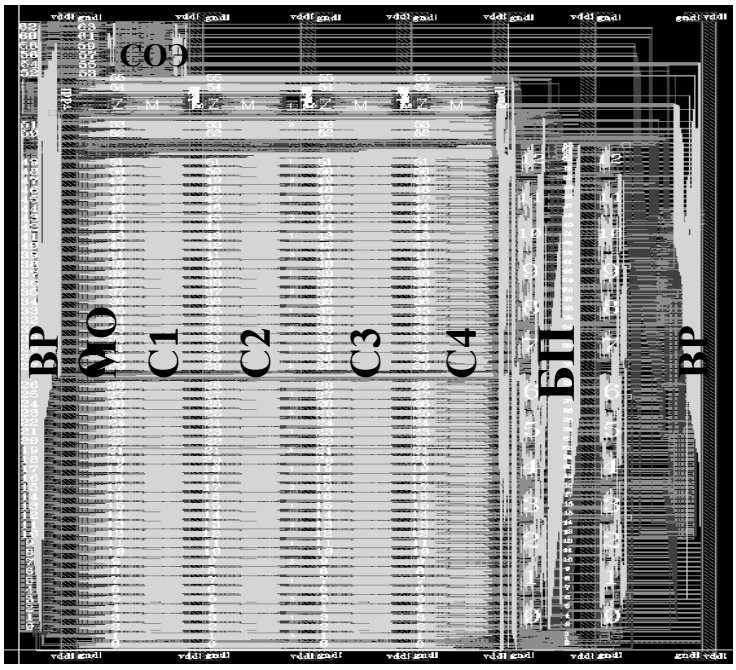

Рис. 9. Топологическая реализация 4-стадийного делителя

В данном случае при частоте синхронного окружения 250 МГц время выполнения операции деления с двойной точностью составит примерно 13 тактов синхросигнала. Такой же задержкой обладает и синхронная реализация операции извлечения квадратного корня. Распределение задержки по этапам алгоритма выглядит так: 4.2 нс – запись операндов и режимов операции во входные регистры и подготовка вычислений мантиссы в цикле; 37.1 нс – циклическое вычисление мантиссы (0.66 нс на один бит результата); 9.0 нс – постобработка и запись результата в выходной регистр. Как видно, вычисление мантиссы в цикле выполняется достаточно быстро. Относительно большая задержка получается в блоке постобработки за счет использования 56- и 52-разрядных сумматоров с распространением переноса для вычисления полного остатка и окончательного результата, а также из-за организации строго самосинхронного контроля завершения формирования окончательного результата.

Особенностью выбранного варианта ВУ с совмещенным алгоритмом деления и извлечения квадратного корня является также и примерно одинаковое время выполнения

обеих операций. Такой результат получается за счет минимальных дополнительных аппаратурных затрат на реализацию квадратного корня в дополнение к операции деления.

В таблице приведены усредненные данные по временам выполнения операций деления и извлечения квадратного корня по результатам моделирования ограниченного случайного набора операндов и режимов округления. Тем не менее, в этот набор попали все сложные случаи округления в соответствии с требованиями стандарта IEEE 754.

В строках 1–3 таблицы приведены результаты моделирования начального варианта ВУ, разработанного путем формального применения методологии проектирования СС-схем. В остальных строках показаны результаты моделирования оптимизированного по быстродействию варианта ВУ, полученного из начального варианта с помощью умощнения драйверов сильно нагруженных цепей и модификации управления конвейером.

Таблица. Время выполнения операций

№ п/п	Условия моделирования на Ultrasim ($U_{пп}$, T^0C)	Деление, нс		Извлечение корня, нс	
		точность вычислений			
		одинарная	двойная	одинарная	двойная
1	Лучшие по ТЗ (1,98 В, - 60^0C)	26.6[1)]	44.4[1)]	25.8[1)]	44.2[1)]
2	Типовые по ТЗ (1.8 В, 25^0C)	32.8[1)]	50.3[1)]	30.8[1)]	52,5[1)]
3	Худшие по ТЗ (1.62 В, +125^0C)	48.3[1)]	81.0[1)]	46.8[1)]	80.5[1)]
4	Лучшие по ТЗ (1,98 В, - 60^0C)	23.4[2)]	37.8[2)]	23.2[2)]	37.2[2)]
5	Типовые по ТЗ (1.8 В, 25^0C)	29.4[2)]	44.7[2)]	28.1[2)]	45.5[2)]
6	Худшие по ТЗ (1.62 В, +125^0C)	42.0[2)]	68.2 [2)]	42.0 [2)]	68.0 [2)]
7	0.9 В, +125^0C	-	219[2)]	-	-
8	0.8 В, +125^0C	-	300[2)]	-	-
9	0.7 В, +125^0C	-	480[2)]	-	-
10	0.6 В, +125^0C	-	858[2)]	-	-
11	0.5 В, +125^0C	1293[2)]	2100[2)]	-	-
12	0.4 В, +125^0C	4656[2)]	-	-	-
13	0.35 В, +125^0C	12920[2)]	-	-	-
14	0.32 В, +125^0C	15760[2)]	-	-	-

[1)] параметры начального варианта ВУ
[2)] параметры оптимизированного ВУ

Данные в таблице подтверждают, что времена выполнения ВУ обеих операций практически совпадают и для типового режима эксплуатации имеют порядок 50 нс. Для более объективного сравнения синхронной и СС- реализаций ВУ по быстродействию необходимо руководствоваться понятием его реального быстродействия; соответствующего фактическим задержкам его элементов, определяемым реальными условиями работы (температуры, напряжения питания и т.д.). Из таблицы видно, что быстродействие ВУ по условиям, заданным в ТЗ изменяется в широких пределах: лучшее и худшее времена исполнения отличаются почти в два раза.

ВУ, как и любое СС-устройство, характеризуется устойчивой работоспособностью

при пониженном напряжении питания, например, при падении напряжения батареи за допустимые нормы (см. строки 7 – 14 таблицы). Для тех приложений, где определяющим фактором является сохранение работоспособности даже за счет существенного падения быстродействия устройства, применение СС-аппаратуры является актуальным.

Несмотря на то, что реализация данного ВУ является примером квазисамосинхронного исполнения, основной вычислительный тракт практически отвечает требованиям самосинхронности и характеризуется бестестовой самопроверяемостью относительно константных неисправностей. Это свойство, наряду с другими, присущими СС-реализациям, предопределяют высокую эффективность создания надёжных изделий, в том числе и отказоустойчивых.

5. Заключение

На основе описанного алгоритма была разработана программа деления и извлечения квадратного корня на языке C++. Ее тестирование на статистически значимом наборе входных данных для различных типов операций, точностей представляемых операндов, режимов округления продемонстрировало безошибочность работы алгоритма в различных ситуациях.

Разработка СС-ВУ деления и извлечения квадратного корня показала:

- использование методологии проектирования СС-схем позволяет получить жизнеспособное эффективное решение даже для такого в значительной степени комбинационного устройства, как делитель, хотя традиционно комбинационные схемы наименее "удобны" для реализации в СС-базисе;
- эффективное решение ВУ в базисе СС-схем возможно только на основе многостадийной – конвейерной – организации вычислительного тракта; в данном случае – с помощью четырех однотипных стадий, каждая из которых вычисляет один бит результата;
- предложенная реализация обеспечивает одинаковое быстродействие устройства при выполнении обеих операций – деления и извлечения квадратного корня;
- СС-реализация вычислительного тракта позволяет отказаться от использования регистров для хранения промежуточных результатов и за счет этого снизить энергопотребление схемы в целом;
- большое количество информационных сигналов из-за их парафазного кодирования предъявляет особые требования к технологии (число слоев разводки, паразитное влияние соседних трасс и т.д.), по которой предполагается изготавливать устройство, реализующее данный алгоритм. Кроме того, оно автоматически увеличивает плотность трассировки СС-схемы, что негативно сказывается на временных

характеристиках устройства в гораздо большей степени, чем для обычных С-реализаций;

● "узким местом" многоразрядных СС-схем являются сигналы управления, общие для всех разрядов; они имеют самую большую нагрузку и в КМОП технологии должны формироваться мощными устройствами-усилителями для сокращения вносимых ими задержек;

● моделирование показывает, что ВУ имеет быстродействие на уровне лучших синхронных аналогов; блоки пред- и постобработки вносят существенный вклад в общую задержку выполнения операции деления и извлечения квадратного корня (до 20 %), и в этом заключается значительный резерв повышения быстродействия устройства.

Вычислитель, разработанный в соответствии с данным алгоритмом, реализован в составе тестовой схемы по КМОП-технологии с проектными нормами 0.18 мкм. Ее изготовление и тестирование позволит оценить эффективность предложенного решения по сравнению с синхронными аналогами и работоспособность схемы в диапазоне изменяющихся напряжения питания и температуры окружающей среды.

6. Литература

1. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах. Под ред. В.И. Варшавского. М.: Наука, 1986, 400 с.
2. *Varshavsky V., Kishinevsky M., Marakhovsky V.* et al. Self-timed Control of Concurrent Processes, Ed. by V.Varshavsky - Kluver Academic Publishers, 1990. – 245 p.
3. P. Beerel, J. Cortadella, and A. Kondratyev, “Bridging the gap between asynchronous design and designers (Tutorial),” in VLSI Design Conference, (Mumbai), 2004.
4. V. Varshavsky , V. Marakhovsky, GALA (Globally Asynchronous - Locally Arbitrary) Design, Concurrency and Hardware Design, Advances in Petri Nets, p.61-107, January 2002
5. IEEE Standard for Binary Floating-Point Arithmetic / IEEE Std. 754. New York ANSI—1985, Aug.
6. S. E. McQuillan and J. V. McCanny, “Fast VLSI algorithms for division and square root”, J. VLSI Signal Processing, vol. 8, pp. 151–168, Oct. 1994.
7. P. Montuschi, L. Ciminiera, and A. Guistina, “Division unit with Newton-Raphson approximation and digit-by-digit refinement of the quotient,” IEEE Proceedings: Computers and Digital Techniques, vol. 141, pp. 317-324, 1994.
8. Lang T. and Montuschi P. Very-high radix combined division and square root with prescaling and selection by rounding / In Proceedngs of the 12th IEEE Symposium on Computer Arithmetic, IEEE, 1995, July.- pp. 124–131.
9. J. Arjun Prabhu and Gregory B. Zyner. 167 MHz Radix-8 Divide and Square Root Using Overlapped Radix-2 Stages / In Proceedings of the 12th Symposium on Computer Arithmetic (ARITH '95), 1995, pp.155-162.
10. Ajay N., Atul D., Warren J., Debjit D. S. 1-GHz HAL SPARC64® Dual Floating Point Unit with RAS Features / In Proceedings of the 15th IEEE Symposium on Computer Arithmetic (ARITH’01), 2001.
11. T.E.Williams. M.A.Horowitz. A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider / IEEE Journal of Solid-State Circuits, Vol. 26, No.11, pp.1651-1661 (Nov. 1991).
12. G. Matsubara, N. Ide, H. Tago, S. Suzuki and N. Goto. 30-m 55-b Shared Radix 2 Division and Square Root Using a Self-TimedCircuit / In Proceedings of the 12th Symposium on Computer Arithmetic (ARITH '95), 1995, pp.98-105.
13. Chartered Semiconductor 0.18µm IB Process 1.8-Volt SAGE-XTM. Standard Cell Library Databook / Artisan Components, February 2003, Release 1.0.