Цифровые фильтры в полях Галуа

Мария ГУЛЕНКО
Дмитрий КАПЛУН

Введение

В операциях цифровой обработки сигналов особое внимание уделяется цифровой фильтрации, которая в среднем занимает до половины всего объема вычислений. В узком смысле цифровой фильтр — это частотно-избирательная цепь, обеспечивающая селекцию цифровых сигналов по частоте [1]. После выполнения цифровой фильтрации мы, как правило, получаем интересующий нас сигнал, то есть сигнал, несущий нужную нам информацию в виде, удобном для последующей обработки. Соответственно, к параметрам цифровых фильтров в современных системах цифровой обработки сигналов начинают предъявлять повышенные требования. Растет порядок значений фильтров, которые нередко представлены четырехзначным числом, постепенно возрастает и разрядность обрабатываемых данных. Это ведет к увеличению объема вычислений, а значит, и к резкому росту аппаратных затрат. При синтезе цифровых фильтров наибольшие затраты времени и оборудования приходятся на операции умножения и сложения [1]. Следовательно, от эффективности реализации этих арифметических операций (в первую очередь, операции умножения) зависят аппаратные и временные характеристики синтезируемого фильтра, а также практически все его основные параметры. Таким образом, задача минимизации времени вычислений и уменьшения аппаратных затрат сводится к оптимизации каждой из операций умножения и сложения, требуемых для вычисления очередного отфильтрованного отсчета.

Одним из решений поставленной задачи может стать реализация фильтров в конечных полях.

Поля Галуа

Полем называется множество с двумя операциями — сложением и умножением, которые удовлетворяют следующим аксиомам [2]:

1. Множество образует абелеву (коммутативную) группу по сложению.
2. Поле замкнуто относительно умножения, и множество ненулевых элементов образует абелеву группу по умножению.
3. Дистрибутивный закон выполняется для любых элементов поля.

Широко известны примеры полей с бесконечным числом элементов: множество вещественных чисел, множество комплексных чисел, множество рациональных чисел. Имеются также поля с конечным числом элементов.

Поле с q элементами, если оно существует, называется конечным полем, или полем Галуа (Galois Fields — GF) в честь французского математика Эвариста Галуа, и обозначается $GF(q)$ [2].

Конечные поля можно описывать с помощью таблиц сложения и умножения. Вычитание и деление однозначно определяются таблицами сложения и умножения. Приведем пример поля $GF(5) = \{0, 1, 2, 3, 4\}$:

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

•	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Для произвольного поля, как бесконечного, так и конечного, применимы почти все известные алгоритмы вычислений. Это происходит потому, что большинство процедур, используемых в полях вещественных и комплексных чисел, зависит только от даваемой определением (приведенным выше) формальной структуры поля и не зависит от частных характеристик конкретного поля. В произвольном поле F есть даже преобразование Фурье [2]:

$$V_k = \sum_{i=0}^{n-1} \omega^{ik} v_i, \; k = 0, \ldots, n-1,$$

где ω — корень степени n из единицы в поле F, а v и V — векторы длины n над полем F. Преобразование Фурье длины n в поле F существует тогда и только тогда, когда поле содержит корень степени n из единицы.

Теперь перейдем к непосредственному использованию конечных полей.

Китайская теорема об остатках

Любое неотрицательное целое число, не превосходящее произведения модулей (а это и будет нашим конечным полем), можно однозначно восстановить, если известны его вычеты по этим модулям. Этот результат был известен еще в Древнем Китае и носит название китайской теоремы об остатках [2].

Китайская теорема об остатках доказывается в два этапа. Сначала доказывается единственность решения, а затем его существование.

Теорема 1. Для заданного множества целых положительных попарно взаимно простых чисел $m_0, m_1, \ldots, m_k$ и множества неотрицательных чисел $c_0, c_1, \ldots, c_k$ при $c_i < m_i$ система:

$$c_i = c \pmod{m_i}, \; i = 0, \ldots, k$$

имеет не более одного решения в интервале $0 \le c \le \prod_{i=1}^{k} m_i$.

Доказательство приведено в [2].

Теорема 2. Пусть $M = \prod_{i=1}^{k} m_i$ — произведение попарно взаимно простых положительных чисел, пусть $M_i = M/m_i$ и пусть для каждого I N_i удовлетворяют равенствам $N_i \times M_i + n_i \times m_i = 1$. Тогда единственным решением системы сравнений:

$$c_i = c \pmod{m_i}, \; i = 0, \ldots, k$$

является:

$$c = \sum_{i=0}^{k} ciNiMi \pmod{M}$$

Доказательство второй теоремы также приведено в [2].

Китайская теорема об остатках является основой представления целых чисел. При таком представлении достаточно просто выполняется операция умножения. Допустим, что надо выполнить умножение $c = a \times b$. Пусть $a_i = R_{m_i}[a]$, $b_i = R_{m_i}[b]$, $c_i = R_{m_i}[c]$ для каждого $i = 0, \ldots, k$. Тогда $c_i = a_i \times b_i \pmod{m_i}$, и это умножение вычислить легче, так как a_i и b_i являются малыми целыми числами. Аналогично, при сложении $c = a + b$ имеем $c_i = a_i + b_i \pmod{m_i}$ для всех $i = 0, \ldots, k$. В обоих случаях для получения окончательного ответа c должно быть восстановлено по вычетам в соответствии с китайской теоремой об остатках.

Вообще переход к системе вычетов позволяет разбить целые числа на маленькие кусочки, которые легко складывать, вычитать и умножать. Если вычисления состоят только из этих операций, то такое представление является альтернативной арифметической системой. Если вычисления достаточно просты, то переход от естественной записи целых чисел к записи через систему остатков и обратное восстановление ответа в целочисленном виде могут свести на нет все возможные преимущества при вычислениях. В случаях же, когда объем вычислений достаточно велик, такой переход может оказаться выгодным. Это происходит потому, что при

вычислениях все промежуточные результаты можно сохранять в виде системы остатков, выполняя обратный переход к целочисленному виду только при окончательном ответе.

Реализация цифровых фильтров в конечных полях

Как известно, классический цифровой фильтр описывается выражением [3]:

$$y(n)=\sum_{i=0}^{N} b_i x(n-i)-\sum_{i=1}^{N} a_i y(n-i), \quad (1)$$

где $y(n)$ — сигнал на входе фильтра; $x(n)$ — сигнал на выходе фильтра; a_i, b_i — коэффициенты фильтра.

Все цифровые фильтры делятся на два обширных класса: нерекурсивные — фильтры с конечной импульсной характеристикой (КИХ) и рекурсивные — фильтры с бесконечной импульсной характеристикой (БИХ) [3]. Мы будем рассматривать только КИХ-фильтры, фазовая характеристика которых, в отличие от БИХ-фильтров, линейна. Для КИХ-фильтров выражение (1) принимает следующий вид [3]:

$$y(n)=\sum_{i=0}^{N} b_i x(n-i). \quad (2)$$

Таким образом, задача синтеза КИХ-фильтра сводится к вычислению коэффициентов b_i такого фильтра.

Теперь, зная китайскую теорему об остатках, перейдем к реализации КИХ-фильтра k-го порядка. Возьмем за основу стандартную структуру (рис. 1).

Для построения алгоритма цифровой фильтрации в полях Галуа воспользуемся китайской теоремой об остатках. Как уже было сказано, она позволяет сохранять при вычислениях все промежуточные результаты в виде системы остатков, выполняя обратный переход к целочисленному виду только при окончательном ответе, что при достаточно большом объеме вычислений может оказаться выгодным. А именно, все вычисления в фильтре будут производиться и сохраняться в системе остатков, и лишь окончательный результат будет восстановлен в привычном виде. В связи с этим структура фильтра будет преобразована к виду, представленному на рис. 2.

Входная последовательность $x(n)$ распараллеливается на k потоков. В каждом потоке каждый отсчет берется по соответствующему модулю (m_i), а потом поступает на блок “block i”. При этом в каждом из блоков будет реализована классическая структура фильтра, изображенная на рис. 1, но все вычисления в нем будут производиться по определенному модулю. Произведение всех модулей соответствует размеру конечного поля:

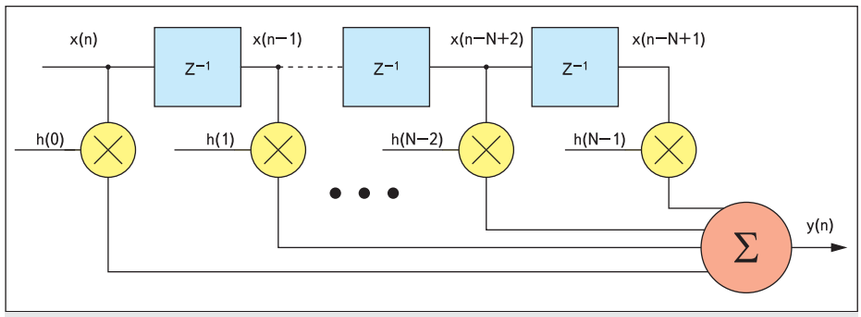

Рис. 1. Структура КИХ-фильтра

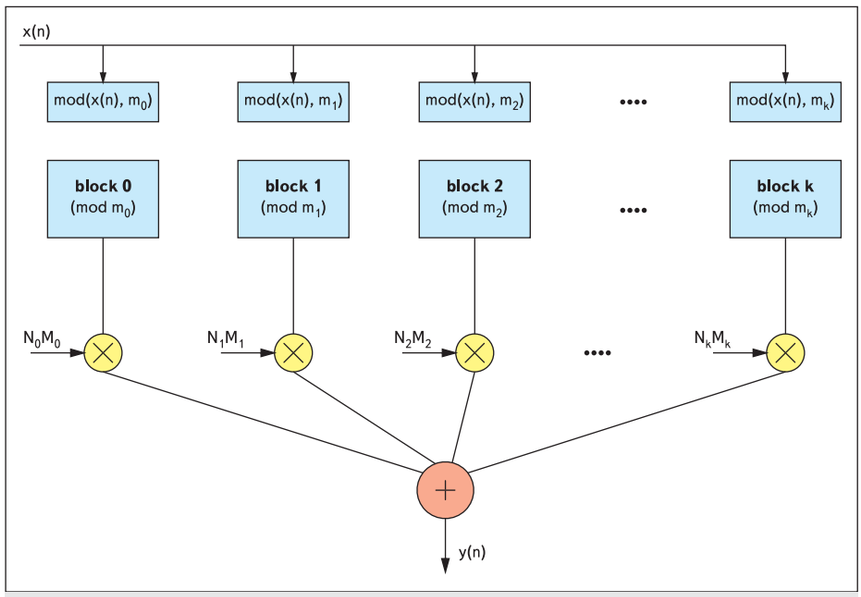

Рис. 2. Структура фильтра в конечных полях

$$M=\Pi_{i=1}^{k} m_i.$$

Далее отсчеты с выходов блоков восстанавливаются по вышеприведенной формуле, а именно каждый из них перемножается на соответствующий множитель $N_i \times M_i$ (где N_i является решением уравнения $N_i \times M_i + n_i \times m_i = 1$, а $M_i = M/m_i$), и все эти произведения суммируются. Так образуется окончательный выходной отсчет $y(n)$ в целочисленном виде. Заметим, что при внутренних вычислениях в эквивалентном фильтре в обычной системе исчисления значения отсчетов не должны превосходить размер поля M.

Рассмотрим пример синтеза КИХ-фильтра 10-го порядка (11 коэффициентов) в среде Simulink пакета математического моделирования MATLAB. Разрядность входного сигнала равна 10. Разрядность коэффициентов фильтра также будет равна 10. Рассчитаем с запасом необходимый размер поля M. Максимальное значение 10-разрядных входного сигнала и коэффициента равно 1023, всего 11 коэффициентов. Следовательно, максимальное значение отсчета не должно превзойти 1023×1023×11=11 511 819. Исходя из этого, подберем подходящий размер поля. Возьмем число M = 11 741 730 = 2×3×5×7×11×13×17×23. Числа $m_0 = 2$, $m_1 = 3$, $m_2 = 5$, $m_3 = 7$, $m_4 = 11$, $m_5 = 13$, $m_6 = 17$ и $m_7 = 23$ являются взаимно простыми, что удовлетворяет условиям китайской теоремы об остатках. Далее, в соответствии со структурой на рис. 2, каждый входной отсчет будет взят по модулю 2, 3, 5, 7, 11, 13, 17 и 23 и поступит на фильтры, в которых все вычисления производятся по соответствующим модулям.

Пример фильтра по модулю $m_2 = 5$ приведен на рис. 3. Напомним, что в этом фильтре все вычисления производятся по модулю 5. Он состоит из 10 элементов задержки, 11 таблиц, в которые записаны векторы со всеми возможными значениями входного сигнала, умноженными на соответствующий коэффициент (по модулю 5) и 10 таблиц сложения для $GF(5)$. И, таким образом, вместо операции умножения (и сложения) будет осуществляться операция выборки по адресу из таблицы умножения (сложения). Адресом же и будут служить перемножаемые числа. Все остальные блоки “block 0”, “block 1”, ..., “block 7” имеют аналогичную структуру.

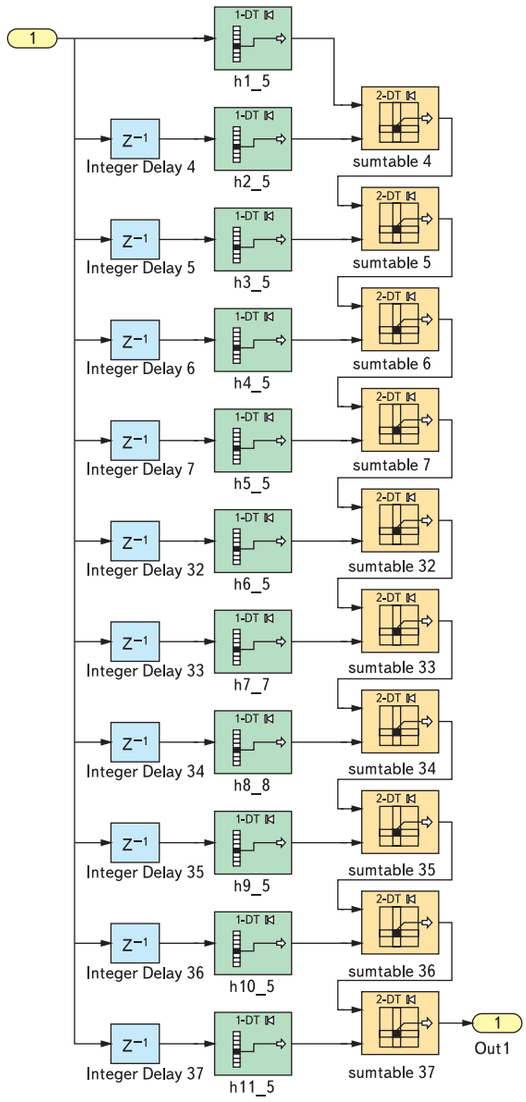

Рис. 3. Фильтр по модулю $m_2 = 5$ в Simulink

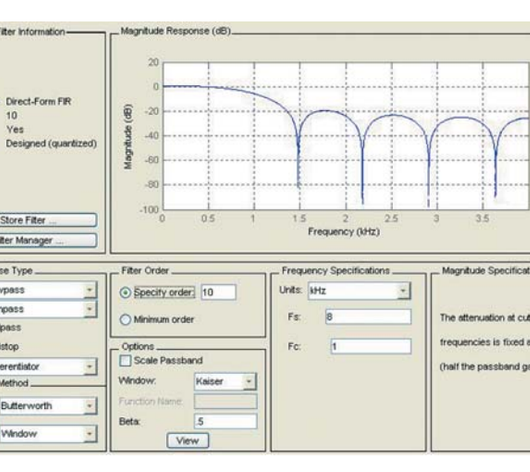

Рис. 4. АЧХ синтезированного фильтра

Для восстановления результата в целочисленном виде нам потребуется найти числа N_i и M_i:

$$M_i = M/m_i,$$
$$M_0 = 11741730/2 = 5870865,$$
$$M_1 = 11741730/3 = 3913910,$$
$$M_2 = 11741730/5 = 2348346,$$
$$M_3 = 11741730/7 = 1677390,$$
$$M_4 = 11741730/11 = 1067430,$$
$$M_5 = 11741730/13 = 903210,$$
$$M_6 = 11741730/17 = 690690,$$
$$M_7 = 11741730/23 = 510510.$$

Воспользовавшись алгоритмом Евклида, получаем: $N_0 = N_2 = N_3 = N_4 = 1$, $N_1 = -1$, $N_5 = 3$, $N_6 = -6$, $N_7 = -11$. Итак, $N_i M_i$ = [5870865 –3913910 2348346 1677390 1067430 27096630 –4144140 –5615610]. Значения коэффициентов получаем с помощью утилиты Filter Design среды MATLAB, задав следующие параметры: фильтр низких частот, метод расчета — Kaizer Window, порядок фильтра — 10, частота дискретизации — 8 кГц, частота среза — 1 кГц, разрядность входных данных — 10. АЧХ такого фильтра изображена на рис. 4.

Аппаратная реализация цифровых фильтров

Классический цифровой КИХ-фильтр состоит из элементов задержки, умножителей и сумматора, на выходе которого мы получаем выходной отсчет (рис. 1). При этом если фильтр k-го порядка, то потребуется $(k+1)$ умножение. А как известно, операция умножения требует больше всего времени и оборудования. Например, при реализации на ПЛИС, при n-разрядном сигнале для одного умножения потребуется n^2 элементарных операций, то есть n^2 логических элементов, следовательно, для фильтра k-го порядка только на умножение уйдет $k \times n^2$ элементов.

Порядки современных фильтров могут достигать десятков тысяч при 10–16-разрядном сигнале. Таким образом, только для умножения в подобном фильтре необходимо несколько миллионов логических элементов.

При аппаратной реализации КИХ-фильтра в конечных полях сокращение количества требуемых логических элементов достигается благодаря особенностям реализации операций сложения и умножения. Поскольку в каждом из блоков (рис. 2) используются модулярные операции, для высокой эффективности нужно применять специально спроектированные для таких систем сумматоры и умножители.

Существует большое количество подходов к реализации сумматоров по модулю m. Далее будут рассмотрены наиболее типичные и простые схемы модулярного суммирования [4].

Первая из них вычисляет модульную сумму $x+y$ с помощью таблицы подстановок (LUT) размером $n \times 2^{2n}$ (рис. 5). Для двух со-

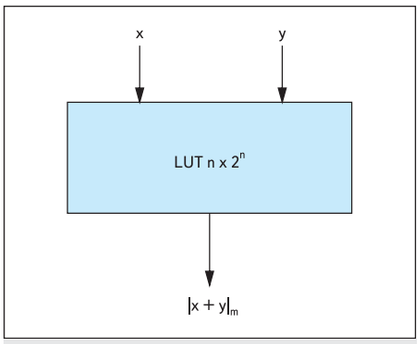

Рис. 5. Суммирование по модулю с помощью большой LUT-таблицы

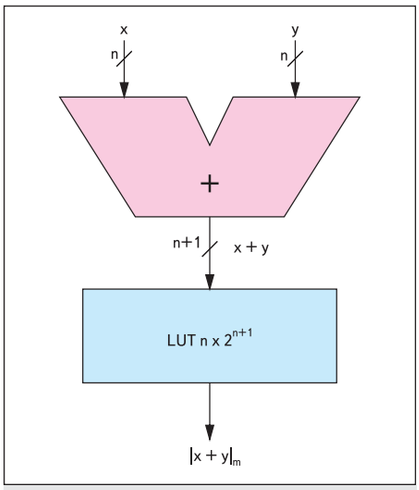

Рис. 6. Суммирование по модулю с предварительным обычным суммированием

ответствующих элементов просто выбирается ответ из большой таблицы. Это решение очень хорошо подходит для случаев, когда длина слова мала.

Для больших модулей память таблицы подстановок (LUT) была бы значительного размера, и другие схемы для суммирования оказываются в этом случае более предпочтительны. Следующее предложение основывается на обычном суммировании $x+y$ и одной таблице, содержащей все возможные значения для $x+y$ по модулю m. При этом существенно сокращается размер подстановочной таблицы — с $n\times2^{2n}$ до $n\times2^{n+1}$, что дает возможность расширять набор модулей при необходимости бо́льшего динамического диапазона или избыточных модульных каналов для коррекции ошибок (рис. 6).

Третья схема суммирования — самая распространенная, она наиболее предпочтительна в большинстве случаев. В этой схеме используются два сумматора и мультиплексор для выбора результата в соответствии с выражением (рис. 7):

$$|x+y|_m = \begin{cases} x+y, & 0 \le x+y \le m \\ x+y-m, & m \le x+y \end{cases}.$$

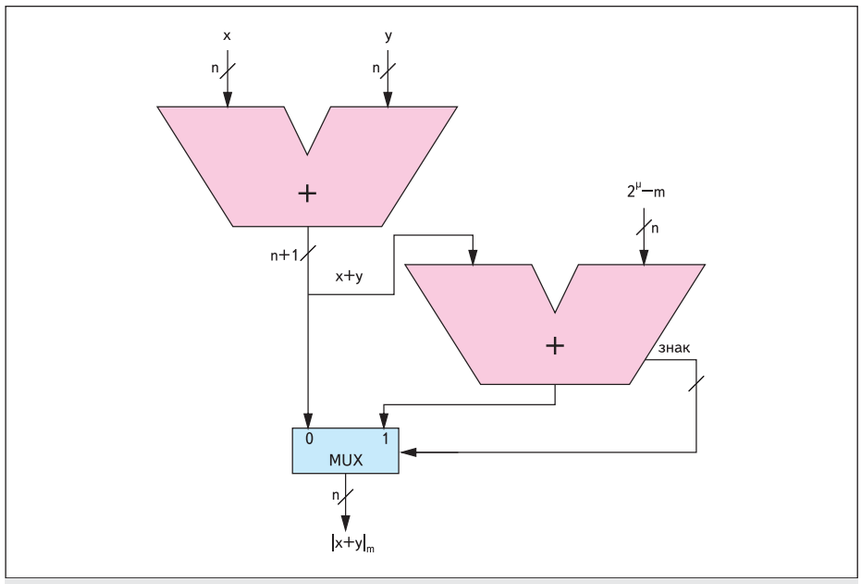

Рис. 7. Без использования LUT-таблиц: μ — аппаратная разрядность сумматора

Теперь перейдем к реализации умножителей. Одним из вариантов реализации модулярного умножителя является, как и при сложении, использование большой таблицы подстановок, когда для двух соответствующих элементов просто выбирается ответ из этой таблицы. Это решение, как и при сложении, хорошо подходит, когда длина слова мала.

Другим вариантом является использование умножителей, основанных на арифметике указателей. Их использование ограничено простыми модулями и базируется на осуществлении преобразования в степенную форму (так называемое степенное исчисление), в котором умножение может более быстро осуществляться посредством операции суммирования.

Метод работы этого умножителя связан с математическими свойствами полей Галуа. Все ненулевые элементы поля Галуа могут быть получены путем многократного возведения в степень примитивного элемента — порождающего поле $GF(p)$ элемента g_j. Это свойство полей Галуа можно использовать для умножения в $GF(m_j)$ благодаря использованию изоморфизма между мультипликативной по модулю m_j группой $Q = \{1, 2, \ldots, m-1\}$ и аддитивной по модулю (m_j-1) группой $I = \{0, 1, \ldots, m-2\}$. Этот изоморфизм может быть установлен следующим образом:

$$\forall q_n \in Q \, \exists i_n \in I: \; q_n = g^{i_n}$$

и умножение над полем $GF(m)$ может производиться по формуле:

$$\left|q_j q_k\right|_m = g^{\left|i_j + i_k\right|_{m-1}}.$$

Таким образом, умножение двух чисел q_j и q_k можно производить, вычисляя модулярную сумму соответствующих указателей i_j и j_k, а затем проводя обратное преобразование из степенного пространства в исходный вид. Необходимо специально обрабатывать случаи, когда один из операндов на входе умножителя равен нулю, и в этом случае назначать нулевой результат произведения. Это происходит потому, что не определен элемент в степенном пространстве, соответствующий нулевому элементу группы Q. Степени i_j и j_k для q_j и q_k, соответственно, могут быть заранее вычислены и помещены в LUT. Сложение степе-

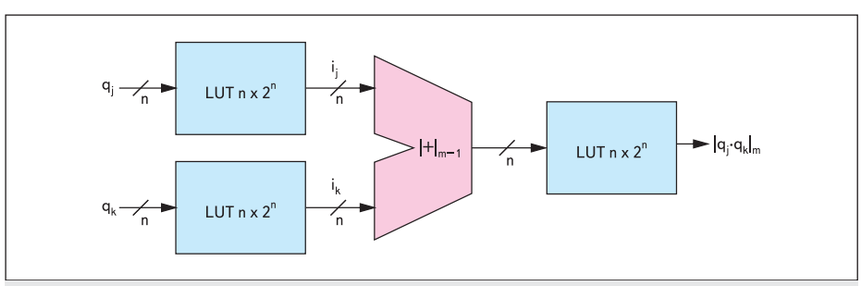

Рис. 8. Умножитель, основанный на исчислении степеней (умножитель Галуа)

ней выполняет сумматор по модулю m_j–1. Обратное преобразование из степенного представления i_j и j_k в исходное q_j и q_k также может быть выполнено с помощью предварительно вычисленных LUT. Такой умножитель показан на рис. 8.

Итак, преимущества при реализации цифровых фильтров в конечных полях достигаются благодаря замене умножителей и сумматоров эквивалентными схемами, которые при определенных условиях позволяют существенно сэкономить аппаратные ресурсы и реализовать фильтры с улучшенными параметрами. В частности, при реализации на ПЛИС вместо аппаратных умножителей и сумматоров используются структуры, которые могут строиться на основе памяти типа ROM (Read Only Memory). Стоимость же такой памяти на порядок меньше стоимости ПЛИСа, который бы мог потребоваться для реализации эквивалентного по параметрам фильтра.

Кроме того, сама структура, изображенная на рис. 2, имеет ряд неоспоримых преимуществ:

1. Независимость каждого канала по отдельному модулю обеспечивает значительную гибкость при планировке и топологическом проектировании кристалла.
2. Реализация таких устройств на основе ПЛИС, обладающих меньшими вентильными ресурсами, может быть легко перепланирована и размещена в несколько кристаллов.
3. Трассировочные межсоединения распространяются только внутри отдельного вычислительного канала, что исключает наличие длинных трасс и, как следствие, обеспечивает некоторое уменьшение потребляемой мощности и уменьшение задержек по критическим путям.

Литература

1. Солонина А. И., Улахович Д. А., Арбузов С. М., Соловьева Е. Б., Гук И. И.. Основы цифровой обработки сигналов: Курс лекций. СПб.: БХВ-Петербург, 2003.
2. Блейхут Р. Быстрые алгоритмы цифровой обработки сигналов: Пер. с англ. М.: Мир, 1989.
3. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов. М.: Мир, 1978.
4. Червяков Н. И., Дьяченко И. В. Принципы построения модулярных сумматоров и умножителей. Сборник научных трудов. Зеленоград: 2006.