значение всех хеш-блоков до этого момента. Хеш-значением всего сообщения является выход последнего блока.

Для вычисления хэша используются пять состояния (A, B, C, D, E). Для хранения этих переменных предусматривается буфер, который представляет собой набор 32-разрядных регистров.

В алгоритме хеширования SHA-1 используется раундовая система расчёта значений. Для каждого раунда определяются своя нелинейная раундовая операция F и раундовая константа K.

Блок сообщения преобразуется из 16 32-разрядных слов M(t) в 80 32-разрядных слов W(t) по определённому правилу. Далее значения сохраняются во временные регистры A, B, C, D, E и производятся запланированные преобразования. После этого текущие значения регистров прибавляются к их исходным значениям соответственно, далее идёт обработка и конечным значением будет объединение пяти 32-разрядных слов в одно 160-разрядное хеш-значение. В случае, когда сообщение состоит из одного SHA-1 блока, полученная в регистрах A, B, C, D, E сумма представляет собой финальный хеш. Для вычисления хеша блок обработки используется в цикле 80 раза, так что финальное значение получается за 80 тактов.

В докладе рассматриваются вопросы построения встраиваемого модуля формирования ключа на основе пароля для FPGA. Архитектурным вариантом аппаратной реализации алгоритма SHA-1 наиболее подходит итеративная архитектура, так как конвейерную архитектуру довольно сложно реализовать из-за последовательности выполнения операций в алгоритме SHA-1 при условии, что сообщение будет больше одного блока. Таким образом итеративная архитектура обеспечивает минимальное использование ресурсов FPGA, однако, и невысокое быстродействие.

Список использованных источников:

1. Т. В. Кузьминов. Криптографические методы защиты информации. Новосибирск: Сибирское предприятие РАН, 1998.
2. Х. К. А. ван Тилборг. Основы криптологии. Профессиональное руководство и интерактивный учебник. Мир, 2006.
3.Ященко В.В. Введение в криптографию. Издание 4-е, дополненное. Москва: Изд-во МЦНМО, 2012.

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ПРОЦЕССОР АДАПТИВНОГО ФИЛЬТРА ВИНЕРА НА РАСПРЕДЕЛЕННОЙ АРИФМЕТИКЕ ДЛЯ СИСТЕМ МУЛЬТИМЕДИА

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Андреев И.Д.

Петровский Н.А. – к.т.н.

В рамках данной работы было разработано VHDL-описание процессора адаптивного фильтра на распределенной арифметике для платформы Xinix Zynq 7010. Данная платформа представляет собой систему на кристалле со встроенным блоком ПЛИС, что значительно упрощает тестирование разрабатываемых аппаратных модулей. Процессор предназначен для подавления акустического эха и обеспечивает обработку аудио в режиме реального времени.

Одной из проблем, препятствующих комфортному общению людей посредством голосовой связи, является эхо – вместо того, чтобы слышать только собеседника, человек слышит еще и самого себя с некоторой задержкой. В таких условиях продолжение беседы порой невозможно.

Акустическое эхо является шумом. Удаление любых шумов из полезного сигнала в компьютерной технике осуществляется с помощью цифровой фильтрации. Наиболее эффективными с точки зрения соотношения производительность\энергопотребление являются аппаратные фильтры. Они позволяют работать в масштабе реального времени потребляя относительно небольшую мощность.

Основной операцией цифровой фильтрации является умножение с накоплением. Однако аппаратный умножитель занимает большую площадь кристалла процессора, а также оказывает негативное влияние на энергопотребление. Особо острой проблема питания становится в условиях полностью автономной работы вычислительной системы.

Поэтому был найден способ замены операции умножения на операцию суммирования, который взял на вооружение распределенную арифметику: числа в двоичной системе счисления представляются в виде суммы произведений разрядов числа и соответствующих их положению в числе степеней двойки. Тогда каждый из разрядов можно использовать в качестве адреса некоторой памяти, в которой хранятся все возможные суммы коэффициентов фильтра. С развитием технологий память стала дешевой, объемной и крайне энергоэффективной, что делает реализацию фильтра с использованием распределенной арифметики на памяти еще более привлекательной.

Проблема фильтрации эха заключается в том, что подобный шум не стационарен: его импульсная характеристика меняется во времени, а значит коэффициенты фильтра не могут быть постоянными. Задачу удаления таких сигналов призван решить адаптивный фильтр: его коэффициенты меняются на основании входных данных.

Таким образом необходимо синтезировать устройство, состоящее из двух компонентов: фильтра с конечной импульсной характеристикой и модуля пересчета коэффициентов.

Фильтр работает в соответствии с выражением[1]:

$$y[n] = \sum_{j=1}^{N-1} Q(b_n) \cdot 2^{-j} + 2^{-(N-1)} \cdot Q(0), \quad (1)$$

где *N* – количество разрядов слова входных данных.

Величина $Q(b_n)$определена следующим образом:

$$Q(b_n) = \sum_{i=0}^{K-1} \frac{A_i}{2} \cdot c_{nj} \quad (2)$$

где A_i –коэффициент фильтра;

K – порядок фильтра.

Величинаc_{nj} вычисляется по формуле:

$$c_{nj} = b_{nj} - \overline{b_{nj}}, \quad (3)$$

где b_{nj} – значение j-го разряда слова, поступившего на вход системы в момент времени n.

$\overline{b_{nj}}$ – инверсия значения этого разряда.

Наконец, величина$Q(0)$ в выражении (1) определена так:

$$Q(0) = \sum_{i=0}^{K-1} \frac{A_i}{2}. \quad (4)$$

В памяти фильтра хранятся всевозможные решения выражения (2), на основании которых производится вычисление результата.

Пересчет коэффициентов фильтра осуществляется в соответствии со следующим выражением[2]:

$$A_i[n+1] = A_i[n] + \mu \cdot e[n] \cdot x[n-i], \quad (5)$$

где μ – константа, определяющая шаг поиска;

$e[n]$– текущая величина ошибки;

$x[n-i]$ – слово входных данных, попавшее в систему в момент времени *n – i.*

Ошибка рассчитывается следующим образом:

$$e[n] = d[n] - y[n], \quad (6)$$

где $d[n]$ – нежелательный сигнал, который необходимо подавить.

Если подставитьвыражение (5) в (2), то получится, что память фильтра необходимо обновлятьследующим образом:

$$Q(b_n) = \sum_{i=0}^{K-1} \frac{A_i}{2} \cdot c_{nj} + \mu \cdot e[n] \cdot \sum_{i=0}^{K-1} x[n-i] \cdot c_{nj}. \quad (7)$$

Таким образом внутри модуля пересчета коэффициентов так же будет находиться некоторый элемент памяти, в котором хранятся всевозможные суммы слов входных данных, умноженные на шаг поиска. По

окончании каждого цикла работы фильтра, его память будет обновлена при помощи значений, хранящихся в памяти модуля пересчета коэффициентов и новой величины ошибки.

Списокиспользованныхисточников:

1. White, S.A. Application of distributed arithmetic to digital signaling processing: a tutorial review / S.A.White // IEEE ASP Mag. – 1989. – №6
2. Allred, D.J. LMS Adaptive Filters Using Distributed Arithmetic for High Throughput / D.J.Allred, H.Yoo, V.Kristman, W.Huang, D.V.Anderson // IEEE Transactions of Circuits and Systems – 2005 – Vol. 52

ПРОЦЕССОР ЦЕЛОЧИСЛЕННОГО ДИСКРЕТНОГО КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ НА ОСНОВЕ РАСПРЕДЕЛЁННОЙ АРИФМЕТИКИ НА СУММАТОРАХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Чернышёв В.С.

Петровский А.А. – д.т.н.

В рамках данной работы было разработано VHDL-описание процессора целочисленного дискретного косинусного преобразования (ДКП) на распределенной арифметике на сумматорах. Процессор предназначен для расчёта ДКП входных из 8 слов, представленных восьмью битами в беззнаковой арифметике.

Не прекращаемое развитие информационных технологий выдвигает на передний план проблему быстрой и качественной передачи разного рода информации по цифровым линиям связи, делая особым важным решение данной проблемы для современного мира.

Алгоритм дискретного косинусного преобразования (ДКП, англ. Discrete Cosine Transform, DCT) является ключевым в выполнении компрессии аудиоданных[1] и изображений[2].

Несмотря на то, что математически размер входных данных может быть любого размера, наиболее распространено использование входных векторов длинной 8 (MPEG-1, MPEG-2, JPEG) и 16 [3]. Ввиду важности алгоритма, многие работали над снижением его сложности, чтобы ускорить процесс вычислений и уменьшить затраты вычислительной мощности, результаты представлены в следующей таблице.

Алгоритм	Год	Необходимо умножений	Необходимо сложений
N. Ahmed, T. Natarjan and K. R. Rao. [4]	1974	64	64
W.-H. Chen H. Smith and S.C. Fralick. [5]	1977	16	26
W.-H. Chen H. Smith and S.C. Fralick. Fast	1977	13	29
Z. Wang. [6]	1984	13	29
B. Lee. [7]	1984	12	29
M. Vetterli and N. Nussbaumer. [8]	1984	12	29
N. Suheiro and M. Hatori. [9]	1986	12	29
H. S. Hou. [10]	1987	12	29
Y. Arai, T. Agui and M. Nakajima. [11]	1988	13	29
Loeffler, Ligtengerb and Moschytz. [12]	1988	12	32
Loeffler, Ligtengerb and Moschytz. Fast.	1989	11	29

Таблица 1 – Быстрые 8-точечные ДКП алгоритмы

Ввиду сложности реализации операции умножения, эффективность алгоритма во многом определяется наименьшим их количеством. При реализации ДКП часто берут за основу быстрый алгоритм Лоффлера, из-за малого количество умножений. Чтобы избавиться от оставшихся операция умножения, предлагается использовать распределённую арифметику на сумматорах, которая позволяет заменить операцию умножения на заранее известный коэффициент – операциями сложения и сдвигов. То есть, векторное умножение$Y = X \times A$можно предствавить как: