УДК 004.75,004.272

АНАЛИЗ РЕАЛИЗАЦИЙ АЛГОРИТМА КРИПТОГРАФИЧЕСКОГО ХЭШИРОВАНИЯ SHA-1 НА КОНВЕЙЕРНЫХ СХЕМАХ

Е.В. ЛИСТОПАД

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 1 июля 2015

Проведен анализ аппаратных реализаций алгоритма криптографического хэширования SHA-1 на конвейерных схемах с различным количеством уровней логики. Рассмотрены аппаратные реализации алгоритма SHA-1 для приложений, требующих высокой производительности.

Ключевые слова: алгоритм криптографического хэширования SHA-1, конвейерные схемы, специализированный процессор.

Введение

Конвейерные схемы построения специализированных процессоров позволяют достигать максимальных показателей производительности и эффективности использования аппаратных ресурсов. Примерами являются аппаратные реализации алгоритма криптографического хэширования SHA-1 [1] (далее – алгоритм SHA-1) на конвейерных схемах [2]. Известные на сегодняшний день универсальные аппаратные реализации алгоритма SHA-1 [3, 4] имеют преимущественно итеративную архитектуру, использующую только один блок обработки данных, который реализует одну итерацию алгоритма SHA-1. Такая архитектура обеспечивает минимальное использование аппаратных ресурсов, однако, и минимальное быстродействие. Для определенного класса задач требуется максимальное быстродействие процесса хэширования, которое обеспечивается аппаратными реализациями алгоритма SHA-1, имеющими конвейерную архитектуру вычислений.

Исходные данные

Алгоритм SHA-1 используется в криптографических приложениях и протоколах для вычисления хэша фиксированной длины в 160 бит от входного сообщения максимальной длины 2^{64}–1 бит. Алгоритм SHA-1 работает с 512-разрядными блоками (далее – SHA-1 блоки), которые обрабатываются раздельно. SHA-1 блок разделяется на 16 32-разрядных слов (M_0, …, M_{15}). Для вычисления хэша используются пять переменных состояний (*A, B, C, D, E*). Для хранения этих переменных в аппаратных реализациях предусматривается буфер размером в пять 32-разрядных регистров. На начальном этапе вычислений данные регистры инициализируются шестнадцатеричными значениями (табл. 1).

Таблица 1. **Значения для инициализации регистров**

регистр A	0x67452301
регистр B	0xEFCDAB89
регистр C	0x98BADCFE
регистр D	0x10325476
регистр E	0xC3D2E1F0

Алгоритм хэширования SHA-1 выполняется в четыре этапа по 20 итераций в каждом. Определяются нелинейные операции и константы (1–4).

$$F_t(m,l,k) = (m \wedge l) \vee (\neg m \wedge k), \quad K_t = 0x5A827999, \quad 0 \le t \le 19, \tag{1}$$

$$F_t(m,l,k) = m \oplus l \oplus k, \quad K_t = 0x6ED9EBA1, \quad 20 \le t \le 39, \tag{2}$$

$$F_t(m,l,k) = (m \wedge l) \vee (m \wedge k) \vee (l \wedge k), \quad K_t = 0x8F1BBCDC, \quad 40 \le t \le 59, \tag{3}$$

$$F_t(m,l,k) = m \oplus l \oplus k, \quad K_t = 0xCA62C1D6, \quad 60 \le t \le 79. \tag{4}$$

Блок сообщения преобразуется из 16 32-битовых слов M_t в 80 32-битовых слов W_t (5, 6).

$$W_t = M_t, \quad 0 \le t \le 15, \tag{5}$$

$$W_t = W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16} << 1, \quad 16 \le t \le 79. \tag{6}$$

На рис. 1 приведена схема преобразований, осуществляемых одной итерацией алгоритма SHA-1.

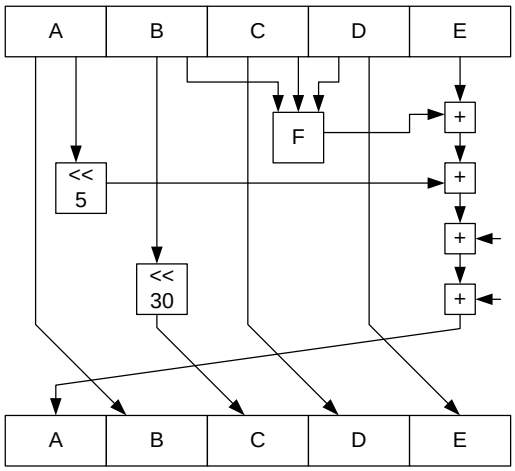

Рис. 1. Схема выполнения одной итерации алгоритма SHA-1

Первоначально значения регистров *A, B, C, D, E* сохраняются во временных переменных. Затем, на каждом шаге $i = 0,...,79$ выполняются необходимые преобразования (7–12).

$$temp = (A << 5) + F_t(B,C,D) + E + W_t + K_t, \tag{7}$$

$$E = D, \tag{8}$$

$$D = C, \tag{9}$$

$$C = B << 30, \tag{10}$$

$$B = A, \tag{11}$$

$$A = temp. \tag{12}$$

После преобразований текущие значения регистров *A, B, C, D, E* прибавляются к их начальным значениям соответственно. Осуществляется переход к следующей итерации. Результирующим значением будет конкатенация пяти 32-битовых слов в одно 160-битное хэш-значение. Каждая итерация изменяет значение регистров *A, B, C, D, E*, используя функцию *F*, которая использует значения регистров *B, C, D* в качестве входных переменных и формирует на выходе 32-разрядное значение. В выражениях (6, 7, 10) знак << обозначает операцию циклического сдвига влево. В целом алгоритм SHA-1 выполняется за 80 итераций. При этом

первые 20 итераций образуют раунд 1, следующие 20 – раунд 2 и т. д. Раунды различаются используемой функцией *F*, и константой *K*, как это определено в выражениях (1–4), где *t* – это номер шага. Как отмечалось выше, SHA-1 блок включает только 16 32-разрядных слов, которые по определенным правилам согласно выражениям (5,6) преобразуются в 80 32-разрядных слов, каждое из которых один раз используется на соответствующей итерации алгоритма. После выполнения всех 80 итераций полученные в регистрах *A, B, C, D, E* значения прибавляются по модулю 2^{32} к значениям временных переменных *A, B, C, D, E*, вычисленных для предыдущего SHA-1 блока текущего обрабатываемого сообщения. В случае, если сообщение состоит из одного SHA-1 блока, полученные в регистрах *A, B, C, D, E* значения после конкатенации дадут финальный хэш. В противном случае значения в указанных регистрах будут использоваться при обработке следующего SHA-1 блока.

Как видно из выражений (7–12), алгоритм SHA-1 имеет последовательную природу. При аппаратной реализации возможности параллельного выполнения операций ограничены имеющимися в алгоритме SHA-1 зависимостями по данным.

Методика эксперимента

Для анализа реализаций алгоритма SHA-1 на конвейерных схемах рассматривается три варианта построения специализированных процессоров. Следует отметить, что для экспериментов строились специализированные процессоры с обработкой входного сообщения длиной не более 160 бит (20 символов в кодировке ASCII).

Первый вариант построения подразумевает использование 82-ступенчатой конвейерной схемы (рис. 2), в которой одна ступень конвейера используется для фиксации исходных данных во входных регистрах, восемьдесят ступеней используются для вычисления всех восьмидесяти итераций алгоритма SHA-1 (при этом одна ступень конвейера вычисляет одну итерацию алгоритма), и одна ступень используется для фиксации результата в выходном регистре процессора.

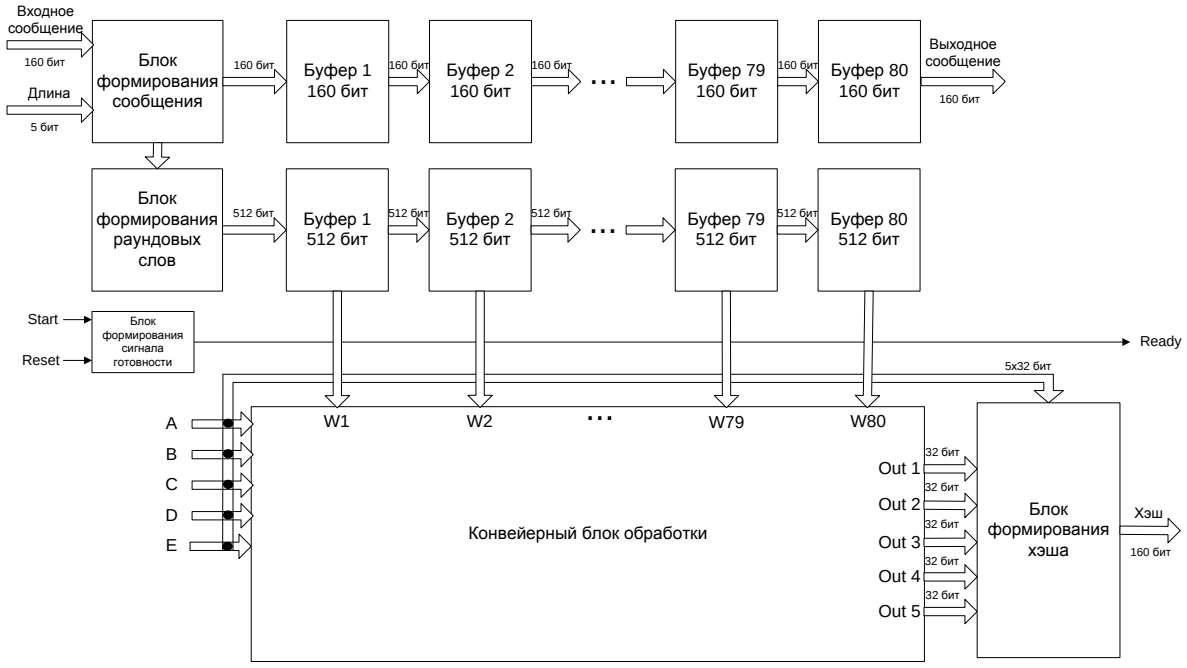

Рис. 2. 82-ступенчатая конвейерная схема построения специализированного процессора

Входное сообщение поступает на блок формирования сообщения. Задачами данного блока являются:

1) передача входных данных в неизменном виде в цепочку буферов, реализованных на сдвиговых регистрах;

2) модификация входных данных в соответствии с требованиями алгоритма (входное сообщение длиной 160 бит дополняется служебной информацией до 512 бит);

3) передача модифицированных входных данных в блок формирования раундовых слов.

Блок формирования раундовых слов представляет 512-битный вектор входных данных в виде 16 32-битовых слов M_t, которые в этом же блоке преобразуются в 80 32-битовых раундовых слов W_t в соответствии с выражениями (5,6). Далее, подготовленные раундовые слова передаются в цепочку буферов, реализованных на сдвиговых регистрах. Впоследствии в ходе вычислительного процесса раундовые слова из каждого буфера на каждом такте процессора поступают в конвейерный блок обработки (рис. 3), который изначально инициализируется фиксированными значениями (табл. 1).

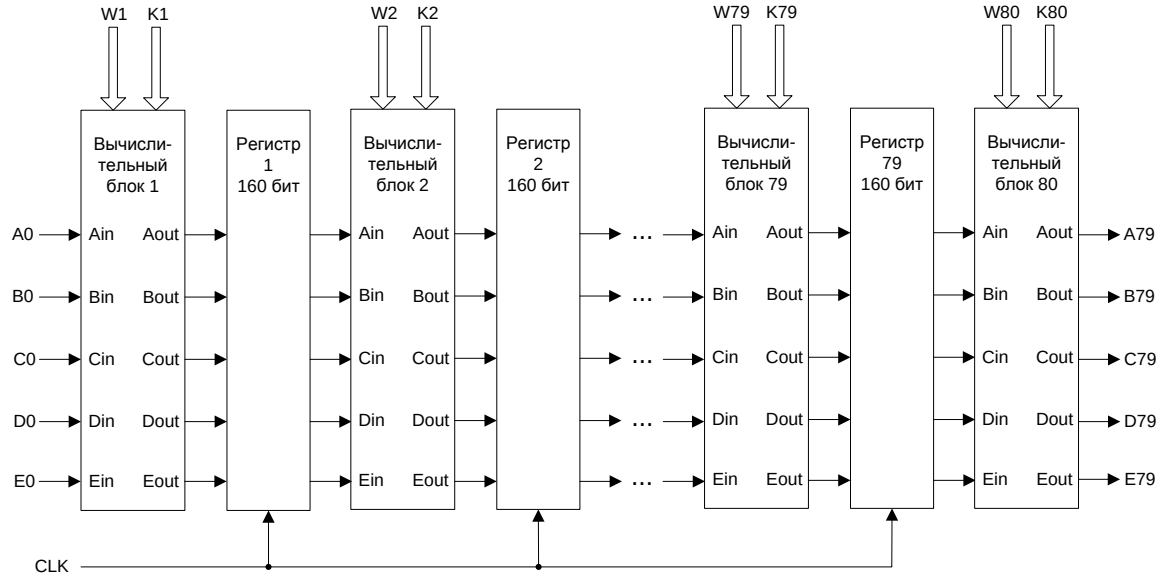

Рис. 3. Структура конвейерного блока обработки для 82-ступенчатой схемы построения специализированного процессора

Структура каждого вычислительного блока соответствует схеме на рис. 1. Таким образом, на каждой ступени конвейера формируются текущие значения переменных *A*, *B*, *C*, *D*, *E* (производится вычисление переменных *A*, *C*, а остальные передаются с перестановкой значений на следующий шаг алгоритма). Вычисление значений переменных *A*, *C* выполняется в соответствии с выражениями (7, 12).

Следует отметить, что описанная конвейерная схема имеет критический путь сигнала, включающий цепочку из четырех сумматоров (для вычисления переменной *A* на рис. 1). В других вариантах построения специализированных процессоров будут использоваться подходы, направленные на минимизацию уровней логики, в том числе и цепочки сумматоров.

Второй вариант построения специализированного процессора подразумевает использование той же конвейерной схемы (рис. 2) с таким же конвейерным блоком обработки (рис. 3), однако с другой структурой вычислительных блоков внутри конвейера. Вычислительный блок со структурой, приведенной на рис. 1, представляется целесообразным модифицировать таким образом, чтобы цепочка из четырех сумматоров оказалась разбитой пополам посредством регистров (рис. 4). Однако использование модифицированных вычислительных блоков требует определенной коррекции структуры конвейерного блока обработки. В частности, модифицированных вычислительных блоков требуется использовать 79 вместо 80, и необходимы дополнительные включения в структуру конвейера блока предварительных вычислений (рис. 5, *а*) и блока финальных вычислений (рис. 5, *б*). Таким образом, конвейерный блок обработки для данного варианта будет иметь 81 ступень, а сама конвейерная схема построения специализированного процессора будет 83-ступенчатой (за счет использования входных и выходных регистров).

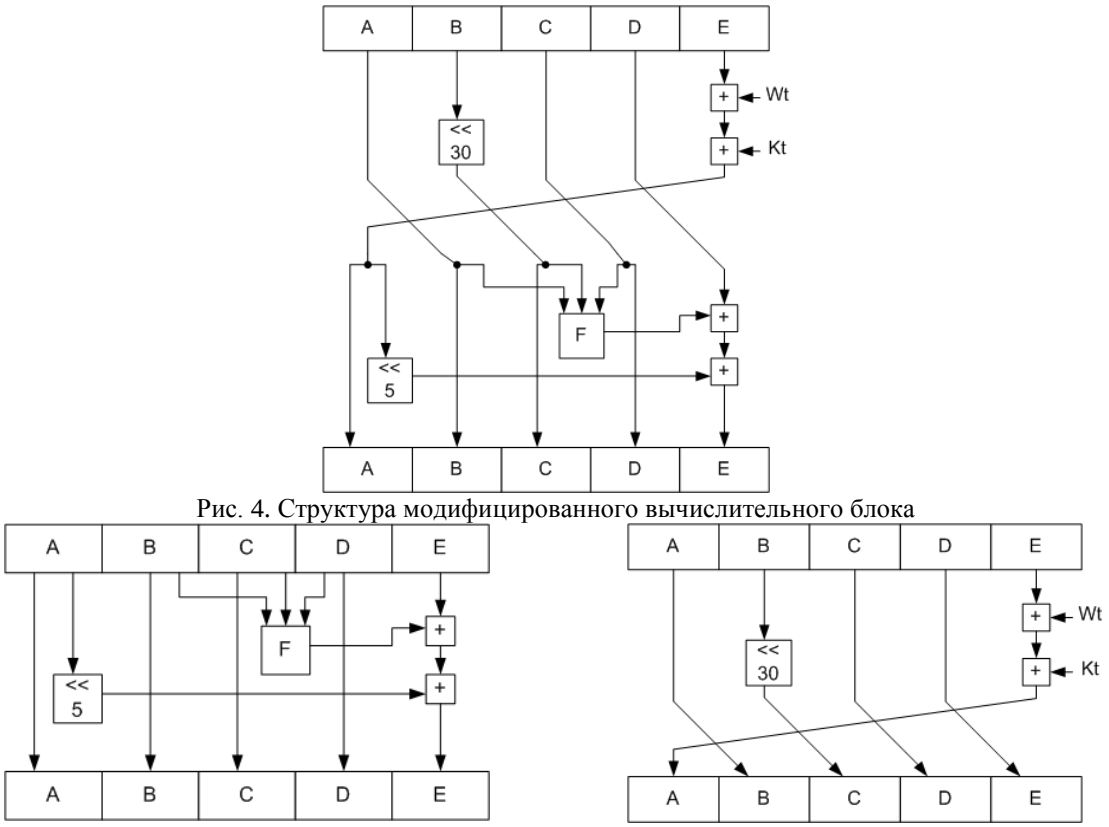

Рис. 4. Структура модифицированного вычислительного блока

а *б*

Рис. 5. Структуры дополнительных блоков:
а – блока предварительных вычислений, *б* – блока финальных вычислений

Третий вариант построения специализированного процессора подразумевает использование 162-ступенчатой конвейерной схемы (рис. 6). Такая схема отличается от 82-ступенчатой конвейерной схемы использованием в своем составе расширенного конвейерного блока обработки, в котором одна итерация алгоритма вычисляется на двух ступенях конвейера.

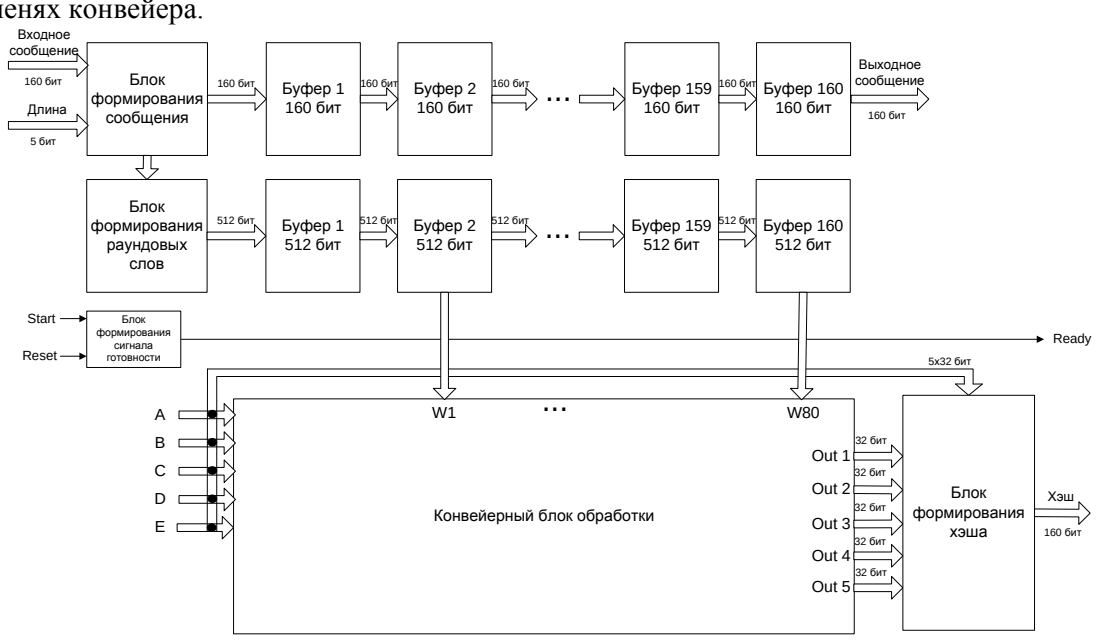

Рис. 6. 162-ступенчатая конвейерная схема построения специализированного процессора

Конвейерный блок обработки (рис. 7) данной схемы состоит из 160 вычислительных блоков двух типов. Первый тип (нечетные блоки) имеет структуру, показанную на рис. 5*а*. Второй тип (четные блоки) имеет структуру, показанную на рис. 5, *б*.

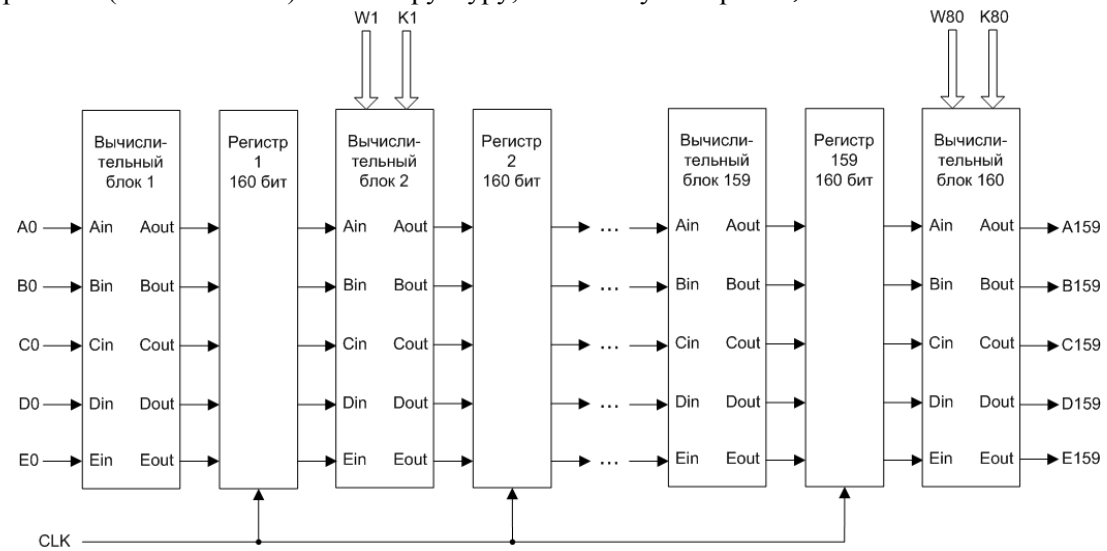

Рис. 7. Структура конвейерного блока обработки для 162-ступенчатой схемы построения специализированного процессора

В данном варианте построения специализированного процессора раундовые слова из исходного сообщения используются только на половине ступеней конвейера, однако необходимо буферизировать эти слова на каждой ступени для сохранения целостности данных. Увеличение числа ступеней позволяет уменьшить количество уровней логики в конвейерной схеме, что, в свою очередь, увеличивает пропускную способность специализированного процессора. Однако при этом необходимо удвоить количество буферных регистров, использующихся для хранения переменных, раундовых слов и исходного сообщения, что приводит к увеличению затрачиваемых аппаратных ресурсов. Для каждого из рассмотренных вариантов построения специализированных процессоров было разработано VHDL-описание и выполнен синтез для FPGA-кристалла XC5VLX110.

Результаты исследований

В результате анализа возможных реализаций алгоритма SHA-1 было построено 3 тестовых специализированных процессора, характеристики которых приведены в табл. 2.

Таблица 2. **Характеристики конвейерных процессоров алгоритма SHA-1, реализованных на кристалле Xilinx XC5VLX110**

№	Вариант построения	Аппаратные затраты (Slices)	Производительность, (Гбит/с)	Частота, МГц
1.	82-ступенчатый конвейер	6869 (39 %)	93,98	197,083
2.	83-ступенчатый конвейер	7547 (43 %)	84,02	176,211
3.	162-ступенчатый конвейер	8215 (47 %)	105,01	220,213

Вариант процессора, построенного с использованием 82-ступенчатой конвейерной схемы, требует минимальных затрат ресурсов кристалла, и при этом опережает по показателям производительности и частоты тот вариант процессора, который построен с использованием 83-ступенчатой конвейерной схемы. Вариант процессора, построенного с использованием 162-ступенчатой конвейерной схемы, требует максимальных затрат ресурсов кристалла, однако демонстрирует лучшие показатели производительности и частоты. Ввиду работы рассматриваемых процессоров с входными данными ограниченной длины, эффективно использоваться такие процессоры могут только при решении конкретных задач, требующих высоких скоростей (до 100 Гбит/с) обработки входных сообщений ограниченной длины. По причине указанной особенности рассматриваемых процессоров, они не могут использоваться в качестве универсального вычислительного средства для нахождения хэша от входного сообщения произвольной длины (ограниченной только требованиями самого алгоритма

SHA-1), следовательно, будет не совсем корректным сравнение их характеристик с известными реализованными универсальными процессорами (табл. 3) алгоритма SHA-1, построенными в том числе с использованием элементов конвейера.

Таблица 3. **Характеристики известных аппаратных реализаций алгоритма SHA-1**

№	Платформа	Аппаратные затраты (Slices)	Производительность, (Мбит/с)	Источник
1.	Xilinx XC2V1000	4258	3 541	[2]
2.	Xilinx XC2S100	423	212	[3]
3.	Xilinx 4VSX35FF668-12	2494	-	[4]
4.	Xilinx XCV1000	1480	1024	[5]
5.	Altera Flex10KE50	1191	61	[6]

Необходимо отметить, что известные универсальные аппаратные реализации алгоритма чаще имеют итеративную архитектуру (а не конвейерную), поскольку количество выполняемых итераций алгоритма зависит от длины входных данных и заранее неизвестно.

Заключение

Проведен анализ возможностей аппаратных реализаций алгоритма криптографического хеширования SHA-1 на конвейерных схемах. По результатам анализа построено 3 тестовых специализированных процессора. Исследована их производительность и возможность использования для решения конкретных задач. Целесообразно продолжать исследования по данной тематике, поскольку усматриваются возможности дальнейшего увеличения количества ступеней конвейера для аппаратной реализации алгоритма SHA-1 на базе выбранного кристалла FPGA. В качестве сферы применения исследуемых аппаратных реализаций стоит рассматривать сферу защиты информации, в частности такие ее элементы как «идентификация пользователя» и «верификация целостности данных».

ANALYSIS OF IMPLEMENTATIONS OF THE CRYPTOGRAPHIC HASH FUNCTION SHA-1 BASED ON PIPELINE CIRCUITS

E.V. LISTOPAD

Abstract

An analysis of hardware implementations of the cryptographic hash function SHA-1 based on pipeline circuits with different numbers of logic levels was made in this paper. The hardware implementations of the hash function SHA-1 for high performance applications were described.

Keywords: cryptographic hash function SHA-1, pipeline circuit, specialized processor.

Список литературы

1. FIPS PUB 180-2:1996, Secure Hash Standard (SHA-1), National Institute of Standards and Technology.
2. *Lee Y., Chan H., Verbauwhede I.* // Proc. of ASAP, 2006, P. 354–359.
3. Guoping Wang. An Efficient Implementation of SHA-1 Hash Function [Электронный ресурс]. – 2009. – Режим доступа: http://xa.yimg.com/kq/groups/19985088/1445823352/name/5.pdf. – Дата доступа: 25.06.2015.
4. *Murat Aşkar, Tuğba Şıltu Çelebi* // ICSTurkey, 2007, P. 85–89.
5. Roar Lien, Tim Grembowski, Kris Gaj. A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512 [Электронный ресурс]. – 2004.– Режим доступа: http://www.ece.iastate.edu/~zambreno/classes/cpre583/documents/LieGre04A.pdf. – Дата доступа: 25.06.2015.
6. Mohamed KhaJil Hani, Ahmad Zoo Sha'ameri, Chong Wei Sheng. Pipeline Implementation of Secure Hash Algorithm (SHA-l) for Cryptographic Application [Электронный ресурс]. – 2000. – Режим доступа: http://eprints.utm.my/10992/1/MohamedKhalilHani2000_PipelineImplementationofSecureHash.pdf. – Дата доступа: 25.06.2015.