Алгоритм минимизации сложности аппаратной реализации сбалансированных S-блоков

Е. А. Курганов[1]

В докладе рассматривается авторский алгоритм, позволяющий получить аппаратную реализацию произвольной системы из m булевых функций от n переменных и его применение к сбалансированным S-блокам. После этого в плане сложности сравниваются аппаратные реализации S-блоков, полученные при помощи нового алгоритма и других известных методов.

Ключевые слова: S-блок, аппаратная реализация, оптимизация сложности схем, потоковые шифры, блочные шифры.

1. Введение

S-блок — это нелинейное преобразование, принимающее на вход n бит и возвращающее m бит. Такое преобразование проще всего представлять как таблицу подстановок размером $n \times m$. Чаще всего в криптографии используют только сбалансированные S-блоки (это значит, что отображение, задаваемое S-блоком, является биекцией, т.е. число входных битов равно числу выходных битов).

В настоящем докладе рассматривается аппаратная реализация сбалансированных S-блоков. Один из определяющих параметров производительности таких реализаций — сложность схемы, т.е. общее число элементов схемы. Рассматривается базис из конъюнкции, дизъюнкции и отрицания (игнорируется при вычислении сложности).

2. Алгоритм минимизации сложности системы булевых функций

Рассмотрим систему из m булевых функций от n переменных $f_1(x_1, \ldots, x_n)$, $\ldots, f_m(x_1, \ldots, x_n)$. Каждой из этих функций можно поставить в соответствие ее вектор значений $p_1, \ldots, p_m$ длины 2^n, где

[1] *Курганов Евгений Александрович* — ведущий разработчик программного обеспечения, ФГАУ НИИ «Восход», e-mail: kuev@yandex.ru.

Kurganov Evgenii Alexandrovich — senior software developer, Scientific and Research Institute Voskhod.

$p_{i_j} \in \{0,1\}$, $i = 1,\dots,m$; $j = 1,\dots,2^n$. Причем $p_{i_j} = 1$ тогда и только тогда, когда в таблице истинности функции f_i на j-м месте стоит 1. Будем рассматривать только системы из попарно различных векторов.

Введем обозначение: для каждого вектора p_i, $i = 1,\dots,m$

$$N_\vee(p_i) = \begin{cases} wt_H(p_i) - 1, & \text{если } wt_H(p_i) > 1, \\ 0, & \text{если } wt_H(p_i) \leqslant 1, \end{cases}$$

где $wt_H(v)$ означает количество координат, равных единице (т.н. вес Хэмминга), для вектора v.

Для пары векторов p_i, p_j, $i, j = 1,\dots,m, i \neq j$ определим вектор $p_i \& p_j$ следующим образом: $(p_i \& p_j)_k = p_{i_k} \& p_{j_k}$.

Основная идея алгоритма состоит в следующем:

1) Разбить исходное множество векторов на пары $(p_i, p_j)_t$, $t = 1,\dots,n/2$, так, что $N_\vee(v_t = (p_i \& p_j)_t)$ максимально.
2) Для каждой пары $(p_i \& p_j)_t$ (где l — номер шага алгоритма)
   - Добавить в изначально пустое множество $P_{l,2}$ векторы $p_i \& \bar{v}_t, p_j \& \bar{v}_t$,
   - Добавить в изначально пустое множество $P_{l,3}$ вектор v_t,
3) Повторить шаги 1-2 для множества $P_{l,2}$ до тех пор, пока $N_\vee(v_t) > 0$,
4) Повторить шаги 1-2 для множества $P_{l,3}$ до тех пор, пока $N_\vee(v_t) > 0$.

Описанный процесс легко представить в виде бинарного дерева. Оно строится по следующим правилам:

1) Корень дерева — исходное множество векторов $p_1,\dots,p_m$,
2) Левый потомок данного узла — множество $P_{l,2}$,
3) Правый потомок данного узла — множество $P_{l,3}$,
4) Узел не имеет потомков, если содержит только один вектор или $N_\vee(v_t) = 0 \, \forall \, t = 1,\dots,n/2$.

Далее для синтеза логической схемы строится дешифратор, после чего для корня дерева вызывается рекурсивная процедура, описанная в алгоритме 2.1. Более подробное описание алгоритма можно найти в статье [1].

Теорема 1. *Обозначим $H = \min\{2m-1, 2^{n-1}\}$. Тогда данный алгоритм имеет сложность $O\big(m^2\, 2^{H+n-1}\big)$.*

Algorithm 2.1 Синтез схемы по бинарному дереву

procedure SYNTHESIZE($node$)
  Для каждого вектора $p_i, i = 1, \ldots, n$
  if $left = 0$ и $right = 0$ **then** ▷ Данный узел — лист
    Синтезировать логическую схему LC_i для p_i
  else
    SYNTHESIZE($right$)
    SYNTHESIZE($left$)
    Синтезировать схему $LC_i = LC_{left_i} \vee LC_{right_{i/2}}$
  end if
end procedure

Теорема 2. *Схема, полученная в результате работы алгоритма 2.1, примененного к результату алгоритма построения бинарного дерева, реализует все m требуемых функций.*

Следствие 1. *Для любого фиксированного $n \geqslant 2$ для каждого S-блока $n \times n$ бит бинарные деревья, получаемые в результате работы алгоритма построения бинарного дерева, изоморфны.*

Следствие 2. *Для каждого S-блока $n \times n$ бит алгоритм построения бинарного дерева работает со сложностью $O(n^2\,2^{3n-2})$.*

3. Практические результаты

Для сравнения результатов была написана программа на языке C++. На вход подается сама подстановка (записанная в файл) и метод, которым надо сгенерировать схему. На выходе получается файл со схемой на языке Verilog. Поддерживаются следующие шесть методов синтеза: наивный, улучшенный наивный, Шеннона, Лупанова, на основе минимальной ДНФ и авторский. Подробнее о методах см. в [1, 2].

S-блок	**Шифр**	**Размер**
S_1	Шифр Кузнечик	8×8 бит
S_2	Шифр AES	8×8 бит
S_3	Шифр ZUC (первый S-блок)	8×8 бит
S_6	Шифр KASUMI (первый S-блок)	7×7 бит
S_7	Шифр KASUMI (второй S-блок)	9×9 бит

Для того, чтобы понять, насколько эффективен приведенный выше метод, были сгенерированы схемы для некоторых S-блоков, использующихся в криптографических алгоритмах (см. таблицу выше). Также данные S-блоки были синтезированы на программе Logic Friday (LF) [3].

Таблица 1. Сложность реализации S-блоков

S-блок	**Наив.(ул.)**	**Шеннон**	**Лупанов**	**мДНФ**	**LF**	**Авт.**
S_1	1319(1068)	680	677	973	823	838
S_2	1319(1068)	680	677	973	780	838
S_3	1319(1068)	680	677	960	800	838
S_6	604(512)	372	353	463	461	416
S_7	2878(2371)	1359	1340	1846	–	1750

В дополнение к результатам статьи [1] в докладе скорректирована оценка сложности алгоритма построения дерева и показано, что авторский метод является развитием идеи улучшенного наивного.

Список литературы

[1] Курганов Е.А., "Об аппаратной реализации сбалансированных S-блоков", *Программная инженерия*, **12**:1 (2020), 8–20.

[2] Яблонский С.В., *Введение в дискретную математику*, «Наука», Москва, 1986, 384 pp.

[3] *Logic Friday*, Программа в сети Интернет [Электронный ресурс], Режим доступа: https://download.cnet.com/developer/logic-friday/i-10268041

An algorithm for minimizing the complexity of hardware implementation of balanced S-blocks

Kurganov E.A.

The paper considers the author's algorithm, which allows obtain a hardware implementation of an arbitrary system of m Boolean functions from n variables and its application to balanced S-blocks. This is followed by a comparison of hardware implementations of S-blocks obtained in terms of complexity.

Keywords: S-box, hardware implementation, circuit complexity optimization, stream ciphers, block ciphers.

References

[1] Kurganov E.A., "On Hardware Implementation of Balanced S-boxes", *Programmnaya Ingeneria*, **12**:1 (2020), 8–20 (in Russian)

[2] Jablonskii S.V., *Introduction to discrete mathematics*, «Nauka», Moscow, 1986 (in Russian), 384 pp.

[3] *Logic Friday*, available at: https: // download.cnet.com/developer/logic-friday/i-10268041