Теоретические вопросы автоматики и информатики

УДК 681.518.5:004.052.32

Д. В. Ефанов, канд. техн. наук,
В. В. Дмитриев
Кафедра «Автоматика и телемеханика на железных дорогах»,
Петербургский государственный университет путей сообщения
Императора Александра I

БАЗИСЫ ДВОИЧНОЙ ЛОГИКИ

Дискретные устройства, синтезированные с использованием различных базисов двоичной логики, могут иметь различную сложность технической реализации в зависимости от технологии изготовления элементной базы. Какие из элементарных функций алгебры логики образуют базис, определяет теорема Поста – Яблонского. Однако возникает задача определения общего количества неизбыточных базисов, т. е. таких базисов, удаление из которых хотя бы одной функции нарушает их функциональную полноту. Эта задача интересна не только с математической точки зрения, но и с точки зрения разработчика – он может выбрать наиболее приемлемый способ технической реализации дискретного устройства из существующих способов. В данной работе рассматривается задача перечисления неизбыточных базисов классической двоичной логики. Приводятся формулы, определяющие соответствие между функциями основного базиса {И; ИЛИ; НЕ} и функциями неизбыточных базисов. Дается пример практического использования неизбыточных базисов для построения логических устройств автоматики.

булева функция; функционально полная система; базис; особенный класс функций

Введение

При построении дискретных устройств автоматики и вычислительной техники часто используются различные функционально полные системы (базисы). Применение разнообразных базисов позволяет синтезировать различные по сложности логические схемы, реализующие одинаковые функции. Параметр сложности, в свою очередь, непосредственно связан с такими важными характеристиками дискретного устройства, как энергопотребление, быстродействие, тестопригодность и др. [1–8].

В данной работе определяются все неизбыточные базисы и устанавливается соответствие между функциями основного базиса {И; ИЛИ; НЕ} и функциями, входящими в каждый неизбыточный базис. Статья может быть полезной проектировщикам дискретных систем, знакомых с различными технологиями изготовления тех или иных логических элементов. Например, логические элементы основного базиса, реализованные по КМОП-технологии, имеют большее количество транзисторов, чем логические элементы {И-НЕ} и {ИЛИ-НЕ}, сами по себе являющиеся базисами [9]. Дискретное устройство может описываться даже более сложным логическим выражением, чем выражение, записанное в базисе {И; ИЛИ; НЕ}, но при этом иметь более простую техническую реализацию. Выбор базиса для синтеза дискретного устройства остается за разработчиком. Важно при этом знать зависимости для преобразования логических выражений, записанных в общеизвестном базисе {И; ИЛИ; НЕ}, в выражения, записанные в выбранном базисе. Результаты исследования, отраженные в статье, позволяют сделать это для любого неизбыточного базиса.

1 Базисы двоичной логики

1.1 Функции алгебры логики

Набор функций {И; ИЛИ; НЕ} является функционально полным, или базисом. Этот факт вытекает из наличия у любой функции стандартной формы записи: дизъюнктивной совершенной нормальной формы (ДСНФ) и конъюнктивной совершенной нормальной формы (КСНФ). Кроме того, этот факт также следует из теоремы Поста – Яблонского [10–12].

На практике для построения логических схем автоматики и вычислительной техники могут использоваться также другие базисы [13]. Возникает задача поиска наиболее оптимального базиса для построения конкретной логической схемы. При этом интересным представляется рассмотрение всех неизбыточных базисов.

В табл. 1 перечислены все элементарные функции алгебры логики (ФАЛ) от двух переменных. Использованы следующие обозначения функций:

0 – константа нуля;

& – логическое умножение (конъюнкция);

Δ – запрет переменной;

x_i – повторение переменной;

$\oplus$ – сложение по модулю два (неравнозначность, исключающее ИЛИ);

$\vee$ – логическое сложение (дизъюнкция);

$\downarrow$ – функция ИЛИ-НЕ (стрелка Пирса);

$\sim$ – эквивалентность (равнозначность);

$\neg$ – логическое отрицание (инверсия);

$\rightarrow$ – импликация;

$|$ – функция И-НЕ (штрих Шеффера);

1 – константа единицы.

Таблица 1. Таблицы истинности элементарных ФАЛ

x_1	x_2	f_0 0	f_1 &	f_2 Δ	f_3 x_1	f_4 Δ	f_5 x_2	f_6 $\oplus$	f_7 $\vee$	f_8 $\downarrow$	f_9 $\sim$	f_{10} $\neg$	f_{11} $\rightarrow$	f_{12} $\neg$	f_{13} $\rightarrow$	f_{14} $\vert$	f_{15} 1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Все ФАЛ можно разделить на три группы:

1) функции, не зависящие от переменных, – это функции f_0 и f_{15};

2) функции, зависящие от одной переменной, – это функции f_3, f_5, f_{10} и f_{12};

3) функции, зависящие от обеих переменных, – это функции $f_1, f_2, f_4, f_6, f_7, f_8, f_9, f_{11}, f_{13}$ и f_{14}.

Записывая ДСНФ или КСНФ для каждой элементарной ФАЛ, можно получить ее выражение через функции основного базиса. Если логическая схема реализована в базисе {И; ИЛИ; НЕ}, то при переходе к технической реализации ее в каком-либо другом базисе требуется выражение функций этого базиса через функции И, ИЛИ и НЕ. Решение этой задачи не является столь очевидным, как кажется на первый взгляд [14].

1.2 Особенные классы функций алгебры логики

Определим, сколько существует неизбыточных базисов. Для этого обратимся к особенным классам ФАЛ.

Существует 5 особенных классов ФАЛ:

1. Функции, сохраняющие ноль ($K0$), – к таким функциям относятся те функции, которые на нулевом наборе аргументов принимают значение, равное нулю: $f(0,0,...,0)=0$.

2. Функции, сохраняющие единицу ($K1$), – к таким функциям относятся те функции, которые на единичном наборе аргументов принимают значение, равное единице: $f(1,1,...,1)=1$.

3. Линейные функции (L) – те функции, которые могут быть представлены в виде $f(x_1, x_2, ..., x_n) = C_0 \oplus C_1x_1 \oplus C_2x_2 \oplus ... \oplus C_nx_n$, где C_i – постоянные, принимающие значения 0 либо 1.

4. Монотонные функции (M) – функции, у которых для любых пар входных наборов $A = \{a_1, a_2, \ldots, a_n\}$ и $B = \{b_1, b_2, \ldots, b_n\}$, таких, что $a_i \le b_i$, имеет место неравенство: $f(A) \le f(B)$.

5. Самодвойственные функции (S) – это функции, для которых выполняется равенство: $f(x_1, x_2, \ldots, x_n) = \overline{f(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})}$.

Особенные классы ФАЛ эффективно используются при построении надежных систем автоматики и вычислительной техники [15–31].

В табл. 2 сведены все элементарные ФАЛ, знаком «×» отмечены те особенные классы, в которые входят соответствующие функции.

Таблица 2. Принадлежность элементарных функций к особенным классам

Особенный класс функций алгебры логики	f_0 0	f_1 &	f_2 Δ	f_3 x_1	f_4 Δ	f_5 x_2	f_6 $\oplus$	f_7 $\vee$	f_8 $\downarrow$	f_9 $\sim$	f_{10} $\neg$	f_{11} $\rightarrow$	f_{12} $\neg$	f_{13} $\rightarrow$	f_{14} $\mid$	f_{15} 1
Сохраняющие 0	×	×	×	×	×	×	×	×								
Сохраняющие 1		×		×		×		×		×		×		×		×
Линейные	×			×		×	×			×	×		×			×
Монотонные	×	×		×		×		×								×
Самодвойственные				×		×					×		×			

На пяти особенных классах ФАЛ базируется теорема о функциональной полноте [10–12, 32].

Теорема Поста – Яблонского. Для того чтобы система функций $\{f_1, f_2, \ldots, f_m\}$ была функционально полной, необходимо и достаточно, чтобы она содержала хотя бы одну функцию, не принадлежащую каждому из особенных классов ФАЛ, т. е. хотя бы одну функцию:

– не сохраняющую константу нуля;
– не сохраняющую константу единицы;
– нелинейную;
– немонотонную;
– несамодвойственную.

Будем рассматривать последовательно все неизбыточные базисы – от тех, которые содержат минимум ФАЛ, до содержащих максимальное их количество. Известно, что максимум ограничен четырьмя функциями [12].

1.3 Неизбыточные базисы

Используя табл. 2 и теорему Поста – Яблонского, получим все не избыточные базисы (табл. 3).

Таблица 3. Функционально полные неизбыточные наборы ФАЛ

№	Набор функций	Принадлежность к особенному классу функций
1	$\downarrow$	$\downarrow \notin K0, K1, L, M, S$
2	$\mid$	$\mid \notin K0, K1, L, M, S$
3	$\rightarrow, 0$	$\rightarrow \notin K0, L, M, S$; $0 \notin K1$
4	$\rightarrow, \Delta$	$\rightarrow \notin K0, L, M, S$; $\Delta \notin K1$
5	$\rightarrow, \oplus$	$\rightarrow \notin K0, L, M, S$; $\oplus \notin K1$
6	$\rightarrow, \neg$	$\rightarrow \notin K0, L, M, S$; $\neg \notin K1$
7	$\Delta, \sim$	$\Delta \notin K1, L, M, S$; $\sim \notin K0$
8	$\Delta, \neg$	$\Delta \notin K1, L, M, S$; $\neg \notin K0$
9	$\Delta, 1$	$\Delta \notin K1, L, M, S$; $1 \notin K0$
10	$\neg, \vee$	$\& \notin S, L$; $\neg \notin K0, K1, M$
11	$\neg, \&$	$\& \notin S, L$; $\neg \notin K0, K1, M$
12	$\oplus, \&, \sim$	$\oplus \notin K1, M, S$; $\& \notin L$; $0 \notin K0$
13	$\oplus, \&, 1$	$\oplus \notin K1, M, S$; $\& \notin L$; $0 \notin K0$
14	$\oplus, \vee, \sim$	$\oplus \notin K1, M, S$; $\vee \notin L$; $0 \notin K0$
15	$\oplus, \vee, 1$	$\oplus \notin K1, M, S$; $\vee \notin L$; $0 \notin K0$
16	$\sim, \&, 0$	$\sim \notin K0, M, S$; $\& \notin L$; $0 \notin K1$
17	$\sim, \vee, 0$	$\sim \notin K0, M, S$; $\vee \notin L$; $0 \notin K1$

Для получения неизбыточных базисов удобно последовательно перебирать те наборы элементарных ФАЛ, которые входят как можно в меньшее количество особенных классов.

1. *Функции-базисы.*

Существует две функции, которые не входят ни в один особенный класс ФАЛ, – это функции И-НЕ и ИЛИ-НЕ. Каждая из них в отдельности образует базис. Добавление хотя бы одной из функций к этим базисам приведет к тому, что набор станет избыточным.

2. *Функции, входящие только в один особенный класс ФАЛ.*

Таких функций, так же как и функций-базисов, всего две – это запрет и импликация. Если предположить, что первой функцией базиса является импликация, то достаточно добавить к ней всего одну ФАЛ из перечисленных в табл. 2 функций, чтобы обе функции образовали базис. Вторая функция не должна входить в особенный класс, в который входит импликация. Таких функций (исключая И-НЕ и ИЛИ-НЕ) четыре: константа нуля, запрет, сложение по модулю два и инверсия.

Аналогично для функции запрета имеем следующие функции, с которыми она образует функционально полные наборы: экивалентность, инверсия, импликация и константа единицы.

3. *Функции, входящие в два особенных класса ФАЛ.*

Функций, входящих в два класса ФАЛ, три: сложение по модулю два, эквивалентность и инверсия. Поиск функций, дополняющих их до базиса, более затруднен. Анализируя таблицу 2, мы убеждаемся, что добавление к функциям сложения по модулю два, эквивалентности и инверсии по одной функции не дает базиса, так как любые из не рассмотренных нами функций принадлежат хотя бы одному классу из тех, в которые входит рассматриваемая функция. Исключение составляет только инверсия, которая вместе с дизъюнкцией и конъюнкцией образует два базиса: $\{\neg, \vee\}$ и $\{\neg, \&\}$.

Инверсия – самодвойственная ФАЛ. Не входящими в этот класс из нерассмотренных являются только константа нуля, константа единицы, сложение по модулю два и эквивалентность. Анализ показывает, что нет новых неизбыточных базисов с функцией инверсии и четырьмя функциями, приведенными выше.

Рассмотрим функцию сложения по модулю два. Данная функция является линейной. Возьмем из не рассмотренных ранее все нелинейные функции. Это только логическое сложение и логическое умножение. Пусть вторая функция базиса – это логическое умножение, тогда имеем следующие неизбыточные базисы: $\{\oplus, \&, \sim\}$ и $\{\oplus, \&, 1\}$. Если вторая функция к функции сложения по модулю два – это функция логического сложения, тогда имеем базисы: $\{\oplus, \vee, \sim\}$ и $\{\oplus, \vee, 1\}$.

Аналогично рассмотренной выше функции сложения по модулю два рассмотрим двойственную к ней функцию эквивалентности. Она также линейна. Выберем из нерассмотренных ФАЛ все нелинейные – это также логическое сложение и логическое умножение. К каждой из них для получения базиса необходимо прибавить функцию, не сохраняющую единицу, – это функции константы нуля и сложения по модулю два. Функция сложения по модулю два рассмотрена выше. Имеем еще два неизбыточных набора функций: $\{\sim, \&, 0\}$ и $\{\sim, \vee, 0\}$.

1.4 Выражения функций основного базиса через функции неизбыточных базисов

Приведем формулы, устанавливающие соответствие между функциями стандартного базиса и другими известными неизбыточными базисами.

При установлении соответствий часто будут использованы формулы де Моргана:

$$\begin{aligned} &\overline{x_1 \vee x_2} = \overline{x_1}\,\overline{x_2};\\ &\overline{x_1 x_2} = \overline{x_1} \vee \overline{x_2}, \end{aligned} \tag{1}$$

а также закон двойного отрицания:

$$\overline{\overline{x}} = x. \tag{2}$$

1. *Функции-базисы*.

К данным функциям, как отмечено ранее, относятся функции И-НЕ и ИЛИ-НЕ. Выразим функции основного базиса в данных базисах:

$$\begin{cases} \overline{x} = x \mid x;\\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\overline{x_1} \cdot \overline{x_2}} = \overline{(x_1 \mid x_1)(x_2 \mid x_2)} = (x_1 \mid x_1) \mid (x_2 \mid x_2);\\ x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{x_1 \mid x_2} = (x_1 \mid x_2) \mid (x_1 \mid x_2). \end{cases} \tag{3}$$

$$\begin{cases} \overline{x} = x \downarrow x;\\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{x_1 \downarrow x_2} = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2);\\ x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{\overline{x_1} \vee \overline{x_2}} = \overline{(x_1 \downarrow x_1) \vee (x_2 \downarrow x_2)} = (x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2). \end{cases} \tag{4}$$

2. *Функции, входящие только в один особенный класс ФАЛ.*

Это функции запрета и импликации. Ясно, что если к одной из них добавить любую ФАЛ, не входящую в тот класс, в который уже входит рассматриваемая функция, мы получим базис.

Рассмотрим импликацию в качестве первой функции базиса. Добавляя к импликации последовательно функции инверсии, запрета, сложения по модулю два и константы нуля, можно получить следующие базисы: $\{\to, \neg\}$, $\{\to, \Delta\}$, $\{\to, \oplus\}, \{\to, 0\}$.

В следующих четырех системах приводится соответствие между функциями основного базиса и функциями перечисленных выше базисов. Для базиса $\{\to, \neg\}$ имеем:

$$\begin{cases} \overline{x}; \\ x_1 \vee x_2 = \overline{x_1} \to x_2; \\ x_1x_2 = \overline{\overline{x_1x_2}} = \overline{\overline{x_1} \vee \overline{x_2}} = \overline{x_1 \to \overline{x_2}}. \end{cases} \tag{5}$$

Для базиса $\{\to, \Delta\}$ имеем:

$$\begin{cases} \overline{x} = x \to (x\Delta x); \\ x_1 \vee x_2 = \overline{x_1} \to x_2 = x_1 \to (x_1\Delta x_1) \to x_2; \\ x_1x_2 = \overline{\overline{x_1x_2}} = \overline{\overline{x_1} \vee \overline{x_2}} = \overline{x_1 \to \overline{x_2}} = \overline{x_1 \to (x_2 \to (x_2\Delta x_2))} = \\ = \left\langle a = x_1 \to (x_2 \to (x_2\Delta x_2)) \right\rangle = a \to (a\Delta a). \end{cases} \tag{6}$$

Для базиса $\{\to, \oplus\}$:

$$\begin{cases} \overline{x} = x \oplus (x \to x); \\ x_1 \vee x_2 = \overline{x_1} \to x_2 = x_1 \oplus (x_1 \to x_1) \to x_2; \\ x_1x_2 = \overline{\overline{x_1x_2}} = \overline{\overline{x_1} \vee \overline{x_2}} = \overline{x_1 \to \overline{x_2}} = \overline{x_1 \to (x_2 \oplus (x_2 \to x_2))} = \\ = \left\langle a = x_1 \to (x_2 \oplus (x_2 \to x_2)) \right\rangle = a \oplus (a \to a). \end{cases} \tag{7}$$

Для базиса $\{\to, 0\}$:

$$\begin{cases} \overline{x} = x \to 0; \\ x_1 \vee x_2 = \overline{x_1} \to x_2 = x_1 \to 0 \to x_2; \\ x_1x_2 = \overline{\overline{x_1x_2}} = \overline{\overline{x_1} \vee \overline{x_2}} = \overline{x_1 \to \overline{x_2}} = \overline{x_1 \to (x_2 \to 0)} = (x_1 \to (x_2 \to 0)) \to 0. \end{cases} \tag{8}$$

Положим, то в качестве первой функции базиса выступает функция запрета. Могут быть использованы следующие базисы: $\{\Delta, \neg\}$, $\{\Delta, \sim\}, \{\to, 1\}$. Базис $\{\Delta, \to\}$ рассмотрен выше.

Для базиса {Δ, ¬} имеем:

$$\begin{cases} \bar{x}; \\ x_1x_2 = \overline{x_1}\Delta x_2; \\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\bar{x}_1\bar{x}_2} = \overline{x_1\Delta\bar{x}_2}. \end{cases} \tag{9}$$

Для базиса {Δ, ~}:

$$\begin{cases} \bar{x} = x \sim (x\Delta x); \\ x_1x_2 = \overline{x_1}\Delta x_2 = x_1 \sim (x_1\Delta x_1)\Delta x_2; \\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\bar{x}_1\bar{x}_2} = \overline{x_1\Delta\bar{x}_2} = \overline{x_1\Delta(x_2 \sim (x_2\Delta x_2))} = \\ = \left\langle a = x_1\Delta(x_2 \sim (x_2\Delta x_2)) \right\rangle = a \sim (a\Delta a). \end{cases} \tag{10}$$

Для базиса {Δ, 1}:

$$\begin{cases} \bar{x} = 1\Delta x; \\ x_1x_2 = \overline{x_1}\Delta x_2 = 1\Delta x_1\Delta x_2; \\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\bar{x}_1\bar{x}_2} = \overline{x_1\Delta\bar{x}_2} = \overline{x_1\Delta(1\Delta x_2)} = 1\Delta(x_1\Delta(1\Delta x_2)). \end{cases} \tag{11}$$

3. **Функции, входящие в два особенных класса ФАЛ**.

Таких функций три: инверсия, сложение по модулю два и равнозначность.

Наборы функций {¬, ∨} и {¬, &} являются функционально полными, что следует из формул де Моргана:

$$\begin{cases} x_1x_2; \\ \bar{x} = x \sim xx; \\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\bar{x}_1\bar{x}_2} = \overline{(x_1 \sim x_1x_1)(x_2 \sim x_2x_2)} = \\ = \left\langle a = (x_1 \sim x_1x_1)(x_2 \sim x_2x_2) \right\rangle = a \sim aa. \end{cases} \tag{12}$$

Аналогично для базиса {⊕, &, 1} имеем:

$$\begin{cases} x_1x_2; \\ \bar{x} = x \oplus 1; \\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\bar{x}_1\bar{x}_2} = \overline{(x_1 \oplus 1)(x_2 \oplus 1)} = (x_1 \oplus 1)(x_2 \oplus 1) \oplus 1 = x_1 \oplus x_2 \oplus x_1x_2. \end{cases} \tag{13}$$

Для базисов {⊕, ∨, ~} и {⊕, ∨, 1} имеем:

$$\begin{cases} x_1 \vee x_2; \\ \overline{x} = x \oplus (x \vee x); \\ x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{\overline{x}_1 \vee \overline{x}_2} = \overline{(x_1 \oplus (x_1 \vee x_1))(x_2 \oplus (x_2 \vee x_2))} = \\ = \left\langle a = (x_1 \oplus (x_1 \vee x_1))\overline{(x_1 \oplus (x_1 \vee x_1))(x_2 \oplus (x_2 \vee x_2))} \right\rangle = a \oplus (a \vee a); \end{cases} \tag{14}$$

$$\begin{cases} x_1 \vee x_2; \\ \overline{x} = x \oplus 1; \\ x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{\overline{x}_1 \vee \overline{x}_2} = \overline{(x_1 \oplus 1) \vee (x_2 \oplus 1)} = ((x_1 \oplus 1) \vee (x_2 \oplus 1)) \oplus 1. \end{cases} \tag{15}$$

Для оставшихся базисов, {~, &, 0} и {~, ∨, 0}, получаем следующие соответствия с функциями основного базиса:

$$\begin{cases} x_1 x_2; \\ \overline{x} = x \sim 0; \\ x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{\overline{x}_1 \cdot \overline{x}_2} = \overline{(x_1 \sim 0)(x_2 \sim 0)} = ((x_1 \sim 0)(x_2 \sim 0)) \sim 0; \end{cases} \tag{16}$$

$$\begin{cases} x_1 \vee x_2; \\ \overline{x} = x \sim 0; \\ x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{\overline{x}_1 \vee \overline{x}_2} = \overline{(x_1 \sim 0) \vee (x_2 \sim 0)} = ((x_1 \sim 0) \vee (x_2 \sim 0)) \sim 0. \end{cases} \tag{17}$$

Таким образом, для всех неизбыточных базисов нами получены выражения, определяющие связь между функциями основного базиса, и функциями, входящими в выбранный неизбыточный базис.

2 Пример использования различных базисов для построения схем автоматики и вычислительной техники

На примере устройства, реализующего память в конечном автомате [33, 34], покажем эффективность выбора вместо базиса {И; ИЛИ; НЕ} базиса {ИЛИ-НЕ}.

Приведем логическую формулу работы асинхронного *RS*-триггера [10]:

$$y(t) = (S(t) \vee y(t-1))\overline{R(t)}, \tag{18}$$

где S – вход установки триггера в единицу; R – вход сброса; y – выход триггера; t – текущий момент времени; t – 1 – предшествующий момент времени.

Опуская обозначения времени, приведем более упрощенную форму записи логической формулы работы асинхронного RS-триггера:

$$y = (S \vee y)\overline{R}. \tag{19}$$

Покажем, как меняется сложность технической реализации при построении схемы RS-триггера на примере двух известных базисов – {И; ИЛИ; НЕ} и {ИЛИ-НЕ}. Используя элементы первого базиса, изобразим схему (рис. 1, *а*). Данная схема построена на трех логических элементах. При построении схемы в базисе {ИЛИ-НЕ} преобразуем формулу (19) к виду [14]:

$$y = S \downarrow y \downarrow R. \tag{20}$$

На рис. 1, *б* приведена логическая схема триггера на элементах ИЛИ-НЕ. Для реализации потребовались два логических элемента. Другими словами, использование базиса {ИЛИ-НЕ} взамен базиса {И; ИЛИ; НЕ} упрощает структуру логической схемы рассматриваемого дискретного устройства.

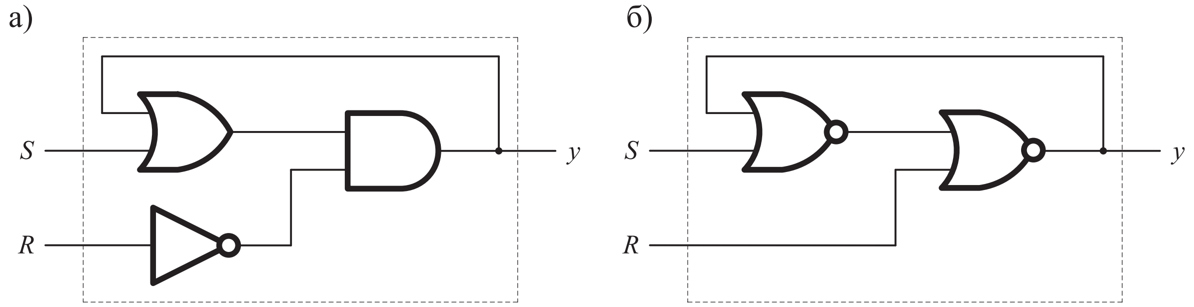

Рис. 1. Логическая схема асинхронного RS-триггера:
а – построенная в базисе {И; ИЛИ; НЕ}; *б* – построенная в базисе {ИЛИ-НЕ}

Пользуясь программным продуктом SIS [35–37], определим в библиотеке функциональных элементов stdcell2_2.genlib сложности структур, показанных на рис. 1. Сложность оценивается условным показателем площади, занимаемой устройством на кристалле микросхемы, в той или иной библиотеке. Результаты сведены в табл. 4.

В табл. 4 также приведены площади асинхронных RS-триггеров в базисах {И; ИЛИ; НЕ} и {ИЛИ-НЕ} – соответственно триггеры 1-го и 2-го типов. Эффективность использования базиса {ИЛИ-НЕ} в сравнении с использованием базиса {И; ИЛИ; НЕ} может быть оценена коэффициентом ε – отношением площадей триггеров 2-го и 1-го типов. Триггер 2-го типа занимает площадь более чем вдвое меньшую, чем триггер 1-го типа.

Таблица 4. Площади *RS*-триггеров в различных библиотеках

Элементы	Площадь в условных единицах
И (AND)	32
ИЛИ (OR)	64
НЕ (NOT)	16
ИЛИ-НЕ (NOR)	24
Элементы памяти	
RS-триггер 1-го типа	112
RS-триггер 2-го типа	48
ε	2,33

Заключение

Основной базис {И; ИЛИ; НЕ} часто используется для описания работы логических устройств автоматики и вычислительной техники. Однако в задачах синтеза аппаратных средств автоматики, реализующих различные функции алгебры логики, более простые структуры могут быть получены с использованием других функционально полных наборов.

В работе показано, что неизбыточных базисов двоичной логики существует не так уж и много, при этом они содержат от одной до трех функций. Функционально полные наборы получены полным перебором с использованием теоремы Поста – Яблонского: два базиса содержат по одной ФАЛ, девять – по две ФАЛ и шесть – по три ФАЛ. Для функций каждого базиса определены логические выражения, связывающие их с функциями основного базиса{И; ИЛИ; НЕ}, что может быть полезным разработчику устройств автоматики и вычислительной техники при реализации конкретного логического устройства с целью получения более простой его структуры, чем при реализации в основном базисе {И; ИЛИ; НЕ}.

Библиографический список

1. McCluskey E. J. Logic Design Principles : With Emphasis on Testable Semicustom Circuits / E. J. McCluskey. – N. Y. : Prentice Hall PTR, 1986. – 549 p.
2. Согомонян Е. С. Самопроверяемые устройства и отказоустойчивые системы / Е. С. Согомонян, Е. В. Слабаков. – М. : Радио и связь, 1989. – 208 с.
3. Сапожников Вал. В. Самопроверяемые дискретные устройства / Вал. В. Сапожников, Вл. В. Сапожников. – СПб. : Энергоатомиздат, 1992. – 224 с.
4. Goessel M. Error Detection Circuits / M. Goessel, S. Graf. – L. : McGraw-Hill, 1994. – 261 p.

5. Abramovici M. Digital System Testing and Testable Design / M. Abramovici, M.A. Breuer, A.D. Friedman. – Comp. Sc. Press, 1998. – 652 p.
6. Fujiwara E. Code Design for Dependable Systems : Theory and Practical Applications / E. Fujiwara. – John Wiley & Sons, 2006. – 720 p.
7. Wang L-T. System-on-Chip Test Architectures : Nanometer Design for Testability / L.-T. Wang, C.E. Stroud, N.A. Touba. – Morgan Kaufmann Publishers, 2008. – 856 p.
8. Проектирование и верификация цифровых систем на кристаллах. Verilog & System Verilog / В.И. Хаханов, И.В. Хаханова, Е.И. Литвинова, О.А. Гузь. – Харьков : Новое слово, 2010. – 528 с.
9. Harris D.M. Digital Design and Computer Architecture / D.M. Harris, S.L. Harris. – N.Y. : Morgan Kaufman, 2013. – 569 p.
10. Сапожников Вал.В. Теория дискретных устройств железнодорожной автоматики, телемеханики и связи : учебник для вузов ж.-д. транспорта / Вал.В. Сапожников, Ю.А. Кравцов, Вл.В. Сапожников ; под ред. В.В. Сапожникова. – М. : УМК МПС, 2001. – 312 с.
11. Яблонский С.В. Введение в дискретную математику : учеб. пособие для вузов / С.В. Яблонский ; под ред. В.А. Садовничева. – 4-е изд., стер. – М. : Высшая школа, 2003. – 384 с.
12. Белоусов А.И. Дискретная математика : учебник для вузов / А.И. Белоусов, С.Б. Ткачев ; под ред. В.С. Зарубина, А.П. Крищенко. – 3-е изд., стер. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2004. – 744 с.
13. Поспелов Д.А. Логические методы анализа и синтеза схем / Д.А. Поспелов. – 3-е изд., перераб. и доп. – М. : Энергия, 1974. – 368 с.
14. Дмитриев В.В. Синтез триггеров на базе теории конечных автоматов / В.В. Дмитриев, К.С. Кононов, А.С. Перский // Автоматика на транспорте. – 2015. – № 1. – С. 73–83.
15. Гессель М. Построение комбинационных самопроверяемых устройств с монотонно независимыми выходами / М. Гессель, А.А. Морозов, Вал.В. Сапожников, Вл.В. Сапожников // Автоматика и телемеханика. – 1994. – № 7. – С. 148–160.
16. Busaba F.Y. Self-Checking Combinational Circuit Design for Single and Unidirectional Multibit Errors / F.Y. Busaba, P.K. Lala // Journal of Electronic Testing: Theory and Application. – 1994. – Issue 5. – Pp. 19–28.
17. Piestrak S.J. Design of Self-Testing Checkers for Unidirectional Error Detecting Codes / S.J. Piestrak. – Wrocław : Oficyna Wydawnicza Politechniki Wrocłavskiej, 1995. – 111 p.
18. Saposhnikov Val.V. Design of Self-Checking Unidirectional Combinational Circuits with Low Area Overhead / Val.V. Saposhnikov, Vl.V. Saposhnikov, A. Morosov, M. Göessel // Proc. IEEE Int. On-Line Testing Workshop (IOLTW). – Biarritz, France, 1996. – Pp. 56–67.
19. Saposhnikov Vl.V. Self-Dual Parity Checking – A New Method for On-Line Testing / Vl.V. Saposhnikov, A. Dmitriev, M. Goessel, Val.V. Saposhnikov // 14th IEEE VLSI Test Symposium, Princeton, New Jersey, USA, April 28 – May 1, 1996. – Pp. 162–168.

20. Гессель М. Исследование комбинационных самопроверяемых устройств с независимыми и монотонно независимыми выходами / М. Гессель, А. А. Морозов, Вал. В. Сапожников, Вл. В. Сапожников // Автоматика и телемеханика. – 1997. – № 2. – С. 180–193.
21. Сапожников Вал. В. Метод построения комбинационных самопроверяемых устройств с обнаружением всех одиночных неисправностей / Вал. В. Сапожников, Вл. В. Сапожников, М. Гессель, А. А. Морозов // Электронное моделирование. – 1998. – Т. 20. – № 6. – С. 70–80.
22. Morosov A. Self-Checking Combinational Circuits with Unidirectionally Independent Outputs / A. Morosov, Val. V. Saposhnikov, Vl. V. Saposhnikov, M. Goessel // VLSI Design. – 1998. – Vol. 5. – Issue 4. – Pp. 333–345.
23. Saposhnikov Val. V. A New Design Method for Self-Checking Unidirectional Combinational Circuits / Val. V. Saposhnikov, A. Morosov, Vl. V. Saposhnikov, M. Göessel // Journal of Electronic Testing: Theory and Application. – 1998. – Vol. 12. – Issue 1–2 (February / April). – Pp. 41–53.
24. Matrosova A. Yu. Self-Checking Synchronous FSM Network Design with Low Overhead / A. Yu. Matrosova, I. Levin, S. A. Ostanin // VLSI Design. – 2000. – Vol. 11. – Issue 1. – Pp. 47–58.
25. Сапожников Вал. В. Самодвойственные дискретные устройства / Вал. В. Сапожников, Вл. В. Сапожников, М. Гёссель. – СПб. : Энергоатомиздат, Санкт-Петербургское отделение, 2001. – 331 с.
26. Matrosova A. Survivable Self-Checking Sequential Circuits / A. Matrosova, I. Levin, S. Ostanin // Proc. of 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2001), Oct. 24–26, San Francisco, CA, 2001. – Pp. 395–402.
27. Гессель М. Логическое дополнение – новый метод контроля комбинационных схем / М. Гессель, А. В. Морозов, Вал. В. Сапожников, Вл. В. Сапожников // Автоматика и телемеханика. – 2003. – № 1. – С. 167–176.
28. Сапожников Вал. В. Основы технической диагностики : учеб. пособие для вузов ж.-д. транспорта / Вал. В. Сапожников, Вл. В. Сапожников. – М. : Маршрут, 2004. – 318 с.
29. Сапожников Вал. В. Синтез самодвойственных дискретных систем / Вал. В. Сапожников, Вл. В. Сапожников, Р. Ш. Валиев. – СПб. : Элмор, 2006. – 224 с.
30. Göessel M. New Methods of Concurrent Checking: Edition 1 / M. Göessel, V. Ocheretny, E. Sogomonyan, D. Marienfeld. – Dordrecht: Springer Science+Business Media B. V., 2008. – 184 p.
31. Lala P. K. An Introduction to Logic Circuit Testing / P. K. Lala. – USA, Arkansas, Texarkana : Morgan & Claypool Publishers, 2009. – 99 p.
32. Закревский А. Д. Логические основы проектирования дискретных устройств / А. Д. Закревский, Ю. В. Поттосин, Л. Д. Черемисинова. – М. : Физматлит, 2007. – 592 с.
33. Matrosova A. Yu. Selection of the flip-flops for partial enhanced scan techniques / A. Yu. Matrosova, A. V. Melnikov, R. V. Mukhamedov, S. A. Ostanin, V. Singh // Вест-

ник Томского государственного университета. Управление, вычислительная техника и информатика. – 2012. – № 2. – С. 112–120.
34. Беннетс Р. Дж. Проектирование тестопригодных логических схем / Р. Дж. Беннетс. – М.: Радио и связь, 1990. – 176 с.
35. Sentovich E. M. SIS: A System for Sequential Circuit Synthesis / E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli // Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 4 May 1992. – 45 p.
36. Gopalakrishan P. Direct Transistor-Level Layout for Digital Blocks / P. Gopalakrishan, R. A. Rutenbar. – Boston : Kluwer Academic Pubishers, 2004. – 125 p.
37. Бибило П. Н. Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний / П. Н. Бибило, В. И. Романов. – Минск : Беларус. навука, 2011. – 279 с.

Efanov Dmitry V.,
Dmitriev Vyacheslav V.
«Automation and Remote Control on Railways» department
Petersburg state transport university

Binary logic bases

Digital devices synthesized with the use of different binary logic bases could have different complexity depending on the component base technology. The Post-Yablonsky theorem defines which elementary boolean functions form basis. But there is a task to determine the common number of not redundant bases, i. e. bases which lose their functionally completeness if any function was removed. This problem is interesting not from mathematical point of view only, but from the side of developer too – he could choose the most acceptable way of implementation. The task of enumeration of all not redundant bases of classical binary logic is considered. Authors adduce formulas of correspondence between primary basis {AND; OR; NOT} and functions of stated bases. Also authors adduce an example of implementation of not redundant bases for logical circuit design.

boolean function; functionally complete system; basis; special class of functions

Reference

1. McCluskey E. J. Logic Design Principles: With Emphasis on Testable Semicustom Circuits. N. Y., Prentice Hall PTR, 1986, 549 p.
2. Sogomonyan E. S., Slabakov E. V. Self-checking devices and failover systems. Moscow, Radio & Communication, 1989, 208 p.

3. Sapozhnikov Val. V., Sapozhnikov Vl. V. Self-checking digital devices. St. Petersburg, Energoatomizdat, 1992, 224 p.
4. Goessel M., Graf S. Error Detection Circuits. L., McGraw-Hill, 1994, 261 p.
5. Abramovici M., Breuer M. A., Friedman A. D. Digital System Testing and Testable Design. Comp. Sc. Press, 1998, 652 p.
6. Fujiwara E. Code Design for Dependable Systems : Theory and Practical Applications. John Wiley & Sons, 2006, 720 p.
7. Wang L.-T., Stroud C. E., Touba N. A. System-on-Chip Test Architectures : Nanometer Design for Testability. Morgan Kaufmann Publishers, 2008, 856 p.
8. Hahanov V. I., Hahanova I. V., Litvinova E. I., Guz O. A. Design and verification of digital SoC. Verilog & System Verilog. Kharkov, New word, 2010, 528 p.
9. Harris D. M., Harris S. L. Digital Design and Computer Architecture. N. Y., Morgan Kaufman, 2013, 569 p.
10. Sapozhnikov Val. V., Kravtsov Yu. A., Sapozhnikov Vl. V. Theory of digital devices of railway automation, remote control and communication. Textbook for railway high school. Moscow, UMK MPS, 2001, 312 p.
11. Yablonsky S. V. Introduction in discrete mathematics: Tutorial for high school. Eds. V. A. Sadovnichev. Moscow, High School, 2003, 384 p.
12. Belousov A. I, Tkachev S. B. Discrete mathematics. Textbook for high school. Eds. V. S. Zarubina, A. P. Krischenko. Moscow, Bauman MSTU, 2004, 744 p.
13. Pospelov D. A. Logical methods of analysis and synthesis of circuits. Moscow, Energy, 1974, 368 p.
14. Dmitriev V. V., Kononov K. S., Persky A. S. Synthesis of flip-flops based on finite state machines theory. Automation on transport, 2015, vol. 1, pp. 73–83.
15. Goessel M., Morozov A. A., Sapozhnikov Val. V., Sapozhnikov Vl. V. Formation of combinational self-checking devices with monotonically independent outputs. Automation & Remote Control, 1994, № 7, pp.148–160.
16. Busaba F. Y., Lala P. K. Self-Checking Combinational Circuit Design for Single and Unidirectional Multibit Errors. Journal of Electronic Testing: Theory and Application, 1994, Issue 5, pp. 19–28.
17. Piestrak S. J. Design of Self-Testing Checkers for Unidirectional Error Detecting Codes. Wrocław, Oficyna Wydawnicza Politechniki Wrocłavskiej, 1995, 111 p.
18. Saposhnikov Val. V., Saposhnikov Vl. V., Morosov A., Göessel M. Design of Self-Checking Unidirectional Combinational Circuits with Low Area Overhead. Proc. IEEE Int. On-Line Testing Workshop (IOLTW), Biarritz, France, 1996, pp. 56–67.
19. Saposhnikov Vl. V., Dmitriev A., Goessel M., Saposhnikov Val. V. Self-Dual Parity Checking – A New Method for On-Line Testing. 14th IEEE VLSI Test Symposium, Princeton, N. J., USA, April 28 – May 1, 1996, pp. 162–168.
20. Goessel M., Morozov A. A., Sapozhnikov Val. V., Sapozhnikov Vl. V. Research of combinational self-checking devices with independent and monotonically independent outputs. Automation & Remote Control, 1997, № 2, pp. 180–193.
21. Sapozhnikov Val. V., Sapozhnikov Vl. V., Goessel M., Morozov A. A. Method of formation of combinational self-checking devices with detection of all single faults. Electronic Modeling, 1998, vol. 20, № 6, pp. 70–80.

22. Morosov A., Saposhnikov Val. V., Saposhnikov Vl. V., Goessel M. Self-Checking Combinational Circuits with Unidirectionally Independent Outputs. VLSI Design, 1998, vol. 5, Issue 4, pp. 333–345.
23. Saposhnikov Val. V., Morosov A., Saposhnikov Vl. V., Göessel M. A New Design Method for Self-Checking Unidirectional Combinational Circuits. Journal of Electronic Testing: Theory and Application, 1998, vol. 12, Issue 1–2 (February / April), pp. 41–53.
24. Matrosova A.Yu., Levin I., Ostanin S.A. Self-Checking Synchronous FSM Network Design with Low Overhead. VLSI Design, 2000, vol. 11, Issue 1, pp. 47–58.
25. Sapozhnikov Val. V., Sapozhnikov Vl. V., Goessel M. Self-dual discrete devices. St. Petersburg, Energoatomizdat, St. Petersburg department, 2001, 331 p.
26. Matrosova A., Levin I., Ostanin S. Survivable Self-Checking Sequential Circuits. Proc. of 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2001), Oct. 24–26, San Francisco, CA, 2001, pp. 395–402.
27. Goessel M., Morozov A. V., Sapozhnikov Val. V., Sapozhnikov Vl. V. Logical addition – new method of combinational circuit test. Automation & Remote Control, 2003, № 1, pp. 167–176.
28. Sapozhnikov Val. V., Sapozhnikov Vl. V. Base of technical diagnostics. Tutorial for railway high school. Moscow, Marshrut, 2004, 318 p.
29. Sapozhnikov Val. V., Sapozhnikov Vl. V., Valiev R. Sh. Synthesis of self-dual digital systems. St. Petersburg, Elmor, 2006, 224 p.
30. Göessel M., Ocheretny V., Sogomonyan E., Marienfeld D. New Methods of Concurrent Checking: Edition 1. Dordrecht, Springer Science+Business Media B. V., 2008, 184 p.
31. Lala P. K. An Introduction to Logic Circuit Testing. USA, Arkansas, Texarkana, Morgan & Claypool Publishers, 2009, 99 p.
32. Zakrevsky A. D., Pottosin Yu. V., Cheremisinova L. D. Logical base of digital device design. Moscow, Fizmathlit, 2007, 592 p.
33. Matrosova A.Yu., Melnikov A. V., Mukhamedov R. V., Ostanin S.A., Singh V. Selection of the flip-flops for partial enhanced scan techniques. Tomsk State University Journal of Control and Computer Science, 2012, № 2, pp. 112–120.
34. Bennets R. Dzh. Design of testable logical circuits. Moscow, Radio & Communication, 1990, 176 p.
35. Sentovich E. M., Singh K. J., Lavagno L., Moon C., Murgai R., Saldanha A., Savoj H., Stephan P. R., Brayton R. K., Sangiovanni-Vincentelli A. SIS: A System for Sequential Circuit Synthesis. Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 4 May 1992, 45 p.
36. Gopalakrishan P., Rutenbar R.A. Direct Transistor-Level Layout for Digital Blocks. Boston, Kluwer Academic Pubishers, 2004, 125 p.
37. Bibilo P. N., Romanov V. I. Logical design of digital devices with the use of production-frame model of knowledge representation. Minsk, Belorusskaja Nauka, 2011, 279 p.

Статья представлена к публикации членом редколлегии
Вал. В. Сапожниковым
Поступила в редакцию 06.05.2015, принята к публикации 02.06.2015

ЕФАНОВ Дмитрий Викторович – кандидат технических наук, доцент кафедры «Автоматика и телемеханика на железных дорогах» Петербургского государственного университета путей сообщения Императора Александра I.
e-mail: TrES-4b@yandex.ru

ДМИТРИЕВ Вячеслав Владимирович – ассистент кафедры «Автоматика и телемеханика на железных дорогах» Петербургского государственного университета путей сообщения Императора Александра I.
e-mail: webus@pisem.net