УДК 004.94 DOI: 10.31114/2078-7707-2020-1-15-19

ПЛИС-ориентированная библиотека логических функций

А.А. Лялинский

Институт проблем проектирования в микроэлекетронике РАН

zelyal@inbox.ru

***Аннотация* — Библиотека логических функций является неотъемлемым атрибутом любой системы проектирования цифровых устройств. Чем разнообразнее ее состав (и, естественно, состав библиотеки цифровых ячеек, реализующих эти функции), тем оптимальнее будет структура разработанного на ее основе устройства. В данной работе рассматриваются вопросы создания библиотек логических функций в составе ПЛИС (массив программируемых логических ячеек). Показаны особенности создания такой системы и характеристики библиотеки, полученной для конкретной ПЛИС.**

***Ключевые слова* — библиотека логических функций, ПЛИС, FPGA, проектирование цифровых устройств.**

I. Введение

Очень часто при проектировании цифрового устройства используется библиотека цифровых ячеек. Это позволяет при проектировании всего устройства не разрабатывать заново схемотехнику нижнего уровня, а применять готовые, ранее апробированные решения. Чисто цифровую часть такой библиотеки можно разделить на разделы с комбинационными ячейками и с триггерами различных видов. Компоненты первого (комбинационного) раздела отличаются друг от друга по выполняемой логической функции, быстродействию, потребляемой мощности и нагрузочной способности. Функциональное многообразие комбинационного раздела библиотеки не связано с технологическими ограничениями, накладываемыми процессом изготовления кристалла, а определяется только временем и ресурсами, выделенными на создание библиотеки. Понятно, что чем больше набор логических функций библиотеки, тем больше возможностей для создания оптимального устройства в целом предоставляется его разработчику. Поэтому могут только приветствоваться любые работы как в целом по автоматизации создания библиотек цифровых ячеек, так и по проработке лежащих в их основе библиотек логических функций (БЛФ).

В разработке цифрового устройства можно выделить два подхода. Исторически первый – это «естественный» путь проектирования, когда под заданный функционал устройства разрабатывается (часто уникальный) набор логических ячеек, служащий для выполнения поставленной задачи. По мере удешевления процесса производства интегральных схем (ИС) и повышения их быстродействия стало возможным получать множество логических ячеек путем перепрограммирования работы одной базовой универсальной ячейки. Это значительно снижает время на разработку БЛФ в целом, так как время прохождения сигнала и потребляемая мощность для всех ячеек библиотеки почти не зависят от выполняемой логической функции. Такой подход получил название ПЛИС [1-3] (Программируемые Логические Интегральные Схемы или FPGA (Field-Programmable Gate Array) в англоязычной литературе).

В данной работе представлена методика получения БЛФ с учетом ограничений, накладываемых структурой базовой ячейки ПЛИС. Описаны алгоритмы создания БЛФ и полученные результаты.

II. Влияние особенностей базовой ячейки ПЛИС на генерацию БЛФ

Всё многообразие логических функций в ПЛИС мы получаем путем программирования базовой ячейки (рис. 1), представляющей собой блок мультиплексоров (БМ), имеющий фиксированное число информационных входов и входов прошивки. Все логические функции реализуются путем подачи определенных комбинаций на входы прошивки E0-E15.

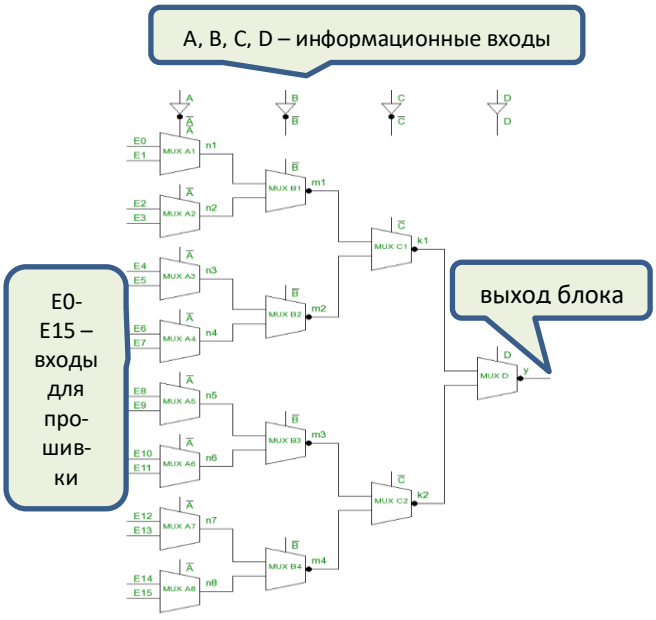

Рис. 1. Структура блока мультиплексоров базовой ячейки

Анализируя данный подход мы видим, что имеются три параметра, влияющие на построение таблицы истинности любой логической функции, реализованной на базовой ячейке. Это: *количество информационных входов, структура блока мультиплексоров* и *значения сигналов на входах прошивки*. Для сравнения: если бы мы реализовали какую-либо логическую функцию на некотором минимальном базисе элементарных логических функций (например, И-НЕ, NOT), то у нас был бы только один управляющий параметр – количество независимых переменных функции.

III. Функциональная полнота БЛФ

Исследование вопросов генерации множества логических функций (или более строго – вопроса функциональной полноты множества логических операций [4]) ведется уже достаточно давно, большое внимание при этом уделяется вопросу, можно ли с выбранным набором элементарных базисных функций обеспечить покрытие имеющейся БЛФ (практически то же самое – какой набор базисных функций требуется для реализации имеющейся БЛФ?).

В нашем случае роль базисных функций выполняет блок мультиплексора, сигналы на его входах прошивки создают варианты «базисных» функций, при этом для 16 входов взятого для исследования конкретного варианта ПЛИС мы имеем огромное количество вариантов поведения этого блока. В этих условиях представляет интерес исследование следующей задачи: при заданном количестве информационных входов, фиксированной структуре БМ и заданном количестве входов прошивки какое максимальное количество имеющих уникальную таблицу истинности функций [5] мы можем получить? Вопрос о независимости этих функций друг от друга (т.е. неважно, можно ли выразить какую-либо функцию через комбинацию других) мы опускаем, так как физически все функции реализуются однотипным путем – посредством кодирования блока мультиплексоров, что не позволяет провести какое-либо упрощение структуры ячейки.

IV. Описание алгоритма генерации БЛФ

Ключевым звеном в решении поставленной в предыдущей главе задачи генерации БЛФ является предложение не пытаться разложить все имеющиеся варианты таблиц истинности (порожденные всеми 2^{16} комбинациями на входах прошивки) на элементарные функции, а подготовить максимально возможное множество разнообразных логических функций и отобрать из него те, для которых таблица истинности совпадает с одним из экземпляров таблицы истинности блока мультиплексора. Первоначально это было опробовано в работе [6], в данной работе используется усовершенствованный вариант алгоритма. Укрупненно описываемый в данной работе алгоритм состоит из следующих этапов:

1) определяемся с числом информационных входов блока мультиплексоров (оно же – число независимых переменных логических выражений) и составом разрешенных элементарных логических операций;
2) вычисляем и сохраняем таблицы истинности для всех вариантов прошивки блока мультиплексоров;
3) для заданного числа независимых переменных и разрешенного состава операций получаем все возможные варианты написания логических выражений;
4) вычисляем и сохраняем таблицу истинности для каждого полученного логического выражения;
5) сравниваем полученные таблицы истинности блока мультиплексора и логических выражений и отбираем для дальнейшего использования совпавшие экземпляры.

Рассмотрим более детально второй и третий этапы (реализация остальных этапов достаточно очевидна).

A. *Генерация таблиц истинности блока мультиплексоров*

Данная задача решается достаточно просто написанием точной модели БМ на любом языке программирования. Далее выполняются два вложенных цикла:

- для всех комбинаций 0 и 1 на входах прошивки:
  - для всех комбинаций 0 и 1 на информационных входах:
    - вычислить и сохранить получаемые записи таблиц истинности.

Пример модели БМ на языке PHP:

```
function mux ($a, $i1, $i2)
   {if (!$a) $x = $i1; else $x = $i2; return $x;}

function muxi ($a, $i1, $i2) {return !mux($a, $i1, $i2);}

function device ($a, $b, $c, $d, &$e)
{
  $a = !$a; $b = !$b; $c = !$c;
  $n1 = mux ($a, $e[0], $e[1]);
  $n2 = mux ($a, $e[2], $e[3]);
  $n3 = mux ($a, $e[4], $e[5]);
  $n4 = mux ($a, $e[6], $e[7]);
  $n5 = mux ($a, $e[8], $e[9]);
  $n6 = mux ($a, $e[10], $e[11]);
  $n7 = mux ($a, $e[12], $e[13]);
  $n8 = mux ($a, $e[14], $e[15]);
  $m1 = muxi ($b, $n1, $n2);
  $m2 = muxi ($b, $n3, $n4);
  $m3 = muxi ($b, $n5, $n6);
  $m4 = muxi ($b, $n7, $n8);
  $k1 = muxi ($c, $m1, $m2);
  $k2 = muxi ($c, $m3, $m4);
  $y = muxi ($d, $k1, $k2);
  if (empty($y)) $y = (string)("0"); else $y = "1";
  return $y;
}
```

B. *Генерация вариантов написания логических выражений*

Предложен и реализован вариант, использующий метод редукции (см. пример в табл. 1 для трех независимых переменных и двух типов операций). Случай с **n=1** (**n** - число независимых переменных) является, по существу, вырожденным, и множество функций для этого случая задается явно. Для случая с **n**=2 мы опять же, используя имеющийся у нас опыт в этой области, задаемся множеством базовых функций, попутно определяя пути его дальнейшего наращивания. Начиная с **n**=3 нам уже не требуется никаких априорных знаний, процесс генерации новых функций становится полностью детерминированным.

Таблица 1

Генерация логических выражений для двух типов операций (& и /) и 3 независимых переменных

Количество независимых переменных	Обозначение множества	Варианты функций	Количество вариантов	Всего
n_{max} = 1	M1	a, !a	2	2
n_{max} = 2	M2_1	a&b, a&!b, !a&b, !a&!b	4	16
	M2_2	то же, что и M2_1, но для оператора «|»	4	
	M2_3	!($M2\_1_i$), i = 1, … 4	4	
	M2_4	!($M2\_2_i$), i = 1, … 4	4	
n_{max} = 3	M3_1	$M2\_{j}$ & c, $M2\_{j}$ & !c, !($M2\_{j}$) & c, !($M2\_{j}$) & !c j = 1, … 4	4x16 → 64	256
	M3_2	то же, что и M3_1, но для оператора «|»	4x16 → 64	
	M3_3	!($M3\_1_i$), i = 1, … 64	64	
	M3_4	!($M3\_2_i$), i = 1, … 64	64	

Для варианта с тремя операциями – И (&), ИЛИ (|) и исключающее ИЛИ (^) - и четырьмя независимыми переменными мы получаем следующие значения (табл. 2).

Таблица 2

Генерация логических выражений для трех типов операций (&, / и ^) и 4 независимых переменных

	Количество независимых переменных			
	1	**2**	**3**	**4**
Количество вариантов	2	24	576	13824

V. Учет особенностей ПЛИС на генерацию БЛФ

Подытожим в данном разделе особенности ПЛИС, повлиявшие на генерацию БЛФ.

1. Количество независимых переменных логической функции фиксировано и равно 4. Реализация 1, 2 и 3-х входовых функций зависит от того, какой сигнал по умолчанию подается на оставшийся незадействованным вход (это задается при разработке схемотехники устройства). Если это 1, то организуется операция «И» с остальными входами, если 0 – то «ИЛИ»:

$$a \,\&\, b \,\&\, c \rightarrow \begin{cases} a \,\&\, b \,\&\, c \,\&\, 1 \\ \text{или} \\ a \,\&\, b \,\&\, c \mid 0 \end{cases}$$

2. Структура блока мультиплексоров не позволяет реализовать некоторые логические функции. Иными словами, существуют такие логические функции, которые нельзя реализовать ни при каких комбинациях 0 и 1 на программирующих входах.
3. Информационные входы, в общем случае, могут быть логически не эквивалентны, Так для рассматриваемой в данной работе ПЛИС входы A, B и C эквивалентны между собой и не эквивалентны входу D (рис. 2). Причина – отсутствие инверсии сигнала на этом входе. В результате реализация, например, функции «a&!d» требует иной прошивки, чем для функции «!a&d».

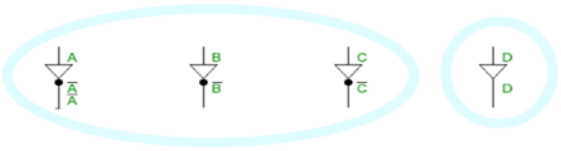

Рис. 2. Логическая неэквивалентность информационных входов блока мультиплексоров базовой ячейки

4. Все входы отличаются по скорости прохождения сигнала от данного входа к выходу. Чем ближе вход к выходу БМ, тем меньше задержка. Это приводит к тому, что функция f=d будет более быстродействующая, чем функция f=a. Или, аналогично, f=b&c&d более быстродействующая, чем f=a&b&c (рис. 3).

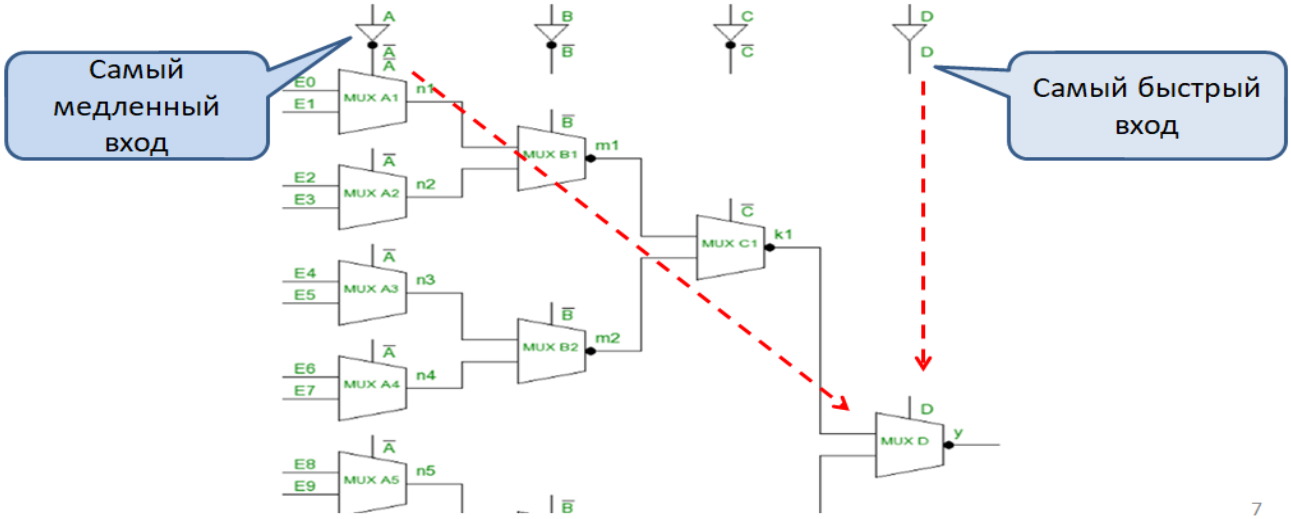

Рис. 3. Различие в быстродействии информационных входов блока мультиплексоров базовой ячейки

VI. Оптимизация состава полученной БЛФ

Достаточно часто разные по написанию функции имеют одну и ту же прошивку. В этом случае для выбора конкретной функции можно опираться на следующие положения:

1) для функций с количеством входов от **n**=1 до n_{max}-1 выбираем ту, входы которой расположены ближе к выходу БМ: вместо a & b & c будем использовать b & c & d . Для **n** = $\mathbf{n_{max}}$ это, естественно, не работает, так как задействованы все входы.
2) Если правило 1 не помогло сделать выбор, то просто выбираем функцию с более короткой длиной записи: вместо !(!c | !d) будем использовать c & d.

VII. Технические результаты

Для $\mathbf{n_{max}}$ = 4 и набора операций {&, |, ^} (И, ИЛИ, EXOR) получены следующие наборы функций:

Таблица 3

Полученные наборы функций

#	функция	информ. входы	условия для остальных входов
Функции с 1 переменной (всего 2)			
1	**d**	d	a=1, b=1, c=1
2	**!d**	d	a=1, b=1, c=1
Функции с 2 переменными (всего 10)			
3	**c&d**	c, d	a=1, b=1
4	**!c&d**	c, d	a=1, b=1
5	**c&!d**	c, d	a=1, b=1
6	**!c&!d**	c, d	a=1, b=1
7	**!c|!d**	c, d	a=1, b=1
8	**c|!d**	c, d	a=1, b=1
9	**!c|d**	c, d	a=1, b=1
10	**c|d**	c, d	a=1, b=1
11	**c^d**	c, d	a=1, b=1
12	**!c^d**	c, d	a=1, b=1
Функции с 3 переменными (всего 68)			
13	**b&c&d**	b, c, d	a=1
14	**!(b&c)&d**	b, c, d	a=1
15	**b&!d&c**	b, d, c	a=1
16	**!(b&c)&!d**	b, c, d	a=1
17	**!(b&c&d)**	b, c, d	a=1
… еще 63 функции			

Функции с 4 переменными (всего 546)			
a&b&c&d	!(a&b&c&d)	a&b&c&!d	!a|!b|!c|d
a&b&c|d	!(a&b&c|d)	a&b&c|!d	!(a&b&c|!d)
a&b&!c&d	a&b&c^d	!a|!b|c|d	a&b&c^!d
a&b&!c|d	!a|!b|c|!d	!(a&b&!c|!d)	a&b&!c^d
a&b&!c^!d	a&b&!c&!d	!(a&b|c|!d)	!(a&b|c&d)
a&b|c&!d	!(a&b&!c|d)	!(a&b|!c^d)	!(a&b|c|d)
a&b|c|!d	a&b&!c|!d	a&b|c^d	!(a&b|c^d)
a&b|!c^d	a&b|c&d	a&b|c|d	!(a&b|c&!d)
… еще 514 функций			

Алгоритм генерации БЛФ реализован на основе php-скриптов, включен в состав сайта fpga.ippm.ru, размещенного на внутреннем веб-сервере ИППМ РАН и предназначенного для проведения характеризации цифровых ячеек в составе ПЛИС в режиме веб-доступа.

Время работы алгоритма – в пределах 10 сек.

VIII. Выводы

Разработана ориентированная на ПЛИС система генерации библиотек логических функций. Высокий уровень автоматизации процессов подготовки данных и обработки полученных результатов, а также удобный веб-интерфейс позволяют снизить временные затраты и повысить качество создаваемых библиотек.

Литература

[1] https://ru.wikipedia.org/wiki/%D0%9F%D0%9B%D0%98%D0%A1 (дата проверки: 03.02.2020).

[2] Соловьев В., Климович А. «Введение в проектирование комбинационных схем на ПЛИС» http://www.chipinfo.ru/literature/chipnews/200305/3.html (дата проверки: 03.02.2020).

[3] Стешенко В.Б. «Школа разработки аппаратуры цифровой обработки сигналов на ПЛИС» https://web.archive.org/web/20071206141712/http://www.msclub.ce.cctpu.edu.ru/pld/steshenko/stat_20.htm (дата проверки: 03.02.2020).

[4] Раздел на Википедии по функциональной полноте: https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0 (дата проверки: 17.01.2020).

[5] Яблонский С. В., Гаврилов Г. П., Кудрявцев В. Б. Функции алгебры логики и классы Поста. — М.: Наука, 1966. — (Математическая логика и основания математики).

[6] Лялинский А.А. Генерация больших наборов логических функций для систем автоматизации проектирования цифровых интегральных схем // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. № 1. С. 9-15.

Characterization of FPGA-based Digital Libraries

A.A. Lyalinsky

Institute for Design Problems in Microelectronics of Russian Academy of Sciences, Moscow

zelyal@inbox.ru

***Abstract* — Library of logical functions is an essential attribute of any digital device design system. The more diverse its composition (and, of course, the composition of the library of digital cells that implement these functions), the more optimal the structure of the device developed on its basis will be. This paper discusses the creation of libraries of logical functions in the FPGA (array of programmable logical cells). The features of creating such a system and the characteristics of the library obtained for a specific FPGA are presented.**

***Keywords* — digital circuit library, characterization of digital cells, web technology, website, cloud technology, circuit simulation.**

References

[1] https://en.wikipedia.org/wiki/Programmable_logic_device checking date: 03.02.2020).

[2] Solov'ev V., Klimovich A. «Vvedenie v proektirovanie kombinacionnyh skhem na PLIS» http://www.chipinfo.ru/literature/chipnews/200305/3.html (checking date: 03.02.2020).

[3] Steshenko V.B. «SHkola razrabotki apparatury cifrovoj obrabotki signalov na PLIS» https://web.archive.org/web/20071206141712/http://www.msclub.ce.cctpu.edu.ru/pld/steshenko/stat_20.htm (checking date: 03.02.2020).

[4] https://en.wikipedia.org/wiki/Functional_completeness (checking date: 17.01.2020).

[5] YAblonskij S. V., Gavrilov G. P., Kudryavcev V. B. Funkcii algebry logiki i klassy Posta. — M.: Nauka, 1966. — (Matematicheskaya logika i osnovaniya matematiki).

[6] Lyalinsky A.A. Web-based automatic generation of input patterns at characterization of digital cells // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2012. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2012. P. 95-100.