УДК 681.3

С.Ф. ТЮРИН[1], А.В. ГРЕКОВ[2], Г.О. ОЛЬТ[3]

[1]Пермский государственный технический университет, Россия
[2]Пермский военный институт внутренних войск МВД РФ, Россия
[3]Пермский государственный университет, Россия

АЛГОРИТМ ПОИСКА РАБОТОСПОСОБНЫХ ЭЛЕМЕНТОВ В ОТКАЗОУСТОЙЧИВЫХ ЦИФРОВЫХ СХЕМАХ

В статье изложены результаты разработки алгоритма поиска работоспособных подмножеств множества функционально-полных толерантных элементов, используемых для синтеза отказоустойчивых программируемых логических интегральных схем. "Внутреннее" резервирование наиболее предпочтительно в программируемых логических устройствах с настройкой связей предложенных функционально-полных толерантных элементов, сохраняющих функциональную полноту при заданной модели отказов и обеспечивающих работоспособность на подмножестве базисов подмножества элементов. Алгоритм позволяет определять работоспособные подмножества элементов в случае отказов цифровых схем для последующего проведения реконфигурации с целью обеспечения гарантированного выполнения вычислительного процесса при допустимом снижении производительности.

Ключевые слова: *функционально-полные толерантные цифровые схемы, отказоустойчивость, базисные функции, векторы.*

Введение

Для синтеза отказоустойчивых ПЛИС на базе функционально-полных толерантных элементов [1 – 4] необходимо по заданной матрице совместимости базисных функций **M1** (рис. 1) и матрице элементов – **M2** (рис. 2), каждый элемент которой есть номер базисной функции из матрицы совместимости, найти подмножество в матрице **M2,** состоящее из заданного числа элементов таким образом, что бы все элементы этого подмножества соответствовали критериям [5]:

1) разность координат по столбцам и строкам не должна превышать заданной плотности **P**;

2) значение элемента i, j ≠ 0; i=1,n; j=1,m; (т.е. элемент должен быть рабочим);

3) єлементы подмножества должны быть совместимы по матрице M1.

Функция	№	1	2	3	4	5	6	7
not X_1 & not X_2 or not X_3 & not X_4	**1**	1	1	1	1	1	1	1
not X_1 or not X_3 & not X_4	**2**	1	1	0	0	0	1	1
not X_2 or not X_3 & not X_4	**3**	1	0	1	0	0	0	1
not X_1 & not X_2 or not X_3	**4**	1	0	0	1	0	1	0
not X_1 & not X_2 or not X_4	**5**	1	0	0	0	1	1	0
not X_1 & not X_2	**6**	1	1	0	1	1	1	0
not X_3 & not X_4	**7**	1	1	1	0	0	0	1

Рис. 1. Матрица совместимости базисных функций

№	1	2	3	4	…	m
1	0	2	0	0	…	0
2	5	7	6	0	…	2
3	0	0	5	4	…	3
4	1	3	5	1	…	6
…	…	…	…	…	…	…
n	2	1	1	3	…	4

**элементы, значения которых равны 0 - не рабочие*

Рис. 2. Матрица элементов

Для матрицы на рис. 2 таким подмножеством, с заданным числом элементов равным 2 и плотностью 0, может быть подмножество из элементов с координатами: (1; 2), (2; 3) или (4; 1), (4; 2) и т.д. Для решения этой задачи предлагается специальный алгоритм.

1. Экспериментальная часть

Алгоритм основан на анализе подмножеств из таблицы элементов. Алгоритм можно разбить на три шага.

Шаг 1. Выбираем случайным образом ненулевые элементы из таблицы элементов и записываем координаты (номер строки, номер столбца) каждого из них в отдельный вектор в позицию с номером ноль. Далее в каждый вектор в соответствующие позиции будем добавлять координаты элементов из

таблицы элементов, если эти элементы соответствуют всем трём критериям из постановки задачи. Элемент, претендующий на добавление в вектор, на соответствие первому критерию будем проверять следующим образом, если разность координат элемента, который претендует на добавление в вектор и координат нулевого элемента не больше заданного расстояния (плотности), то первый критерий выполняется. Полученные таким образом вектора будут подмножествами, которые на следующих шагах будут расширяться или объединяться с другими векторами пока не будут найдено искомое подмножество.

Шаг 2. Все вектора будем сравнивать поэлементно следующим образом, если для двух элементов сравниваемых векторов выполняются критерии 1 и 3 (критерий 2 проверять нет смысла т.к. вектор содержит координаты только рабочих элементов), то объединим эти вектора. Если при сравнении выяснится, что какие либо два вектора содержат координаты одинаковых элементов, то оставим только один, а второй вектор удалим. Если при сравнении выяснится, что один из векторов содержит все элементы другого вектора, то удалим вектор, который содержится другом векторе. Если останется только один вектор, то шаг 2 выполняться не будет.

Шаг 3. Расширим вектора элементами из таблицы элементов подобно тому, как это было сделано на первом шаге, но теперь для проверки первого критерия будем использовать не нулевой, а первый элемент, при следующей итерации второй элемент и т.д. Если дойдём до последнего элемента в векторе, то выполним расширение, а при следующей итерации если размер вектора не увеличится, то для него расширение выполняться не будет, пока размер этого вектора не увеличится.

Шаги 2 и 3 выполняются в цикле. Критерием для завершения цикла может послужить изначально заданное число итераций, если какой-либо вектор будет содержать требуемое количество элементов или если останутся вектора, которые нельзя объединить и расширить.

2. Анализ полученных результатов

В табл. 1 показано, какие элементы будут добавлены в вектор, если на первом шаге в него был добавлен элемент с координатами (2, 2), а плотность равна нулю.

Количество векторов созданных на первом шаге влияет на производительность так .как каждый вектор – это подмножество, которое требует анализа. На рис. 3, 4 показаны результаты работы алгоритма для таблицы элементов размером 30x30 (900 элементов), 100 подмножеств и плотности 0.

Таблица 1

Добавление элементов в вектор

№	1	2	3	4	…	m
1	0	2	0	0	…	0
2	5	7	6	0	…	2
3	0	0	5	4	…	3
4	1	3	5	1	…	6
…	…	…	…	…	…	…
n	2	1	1	3	…	4

– *нулевой элемент вектора;*

– *элементы, которые будут включены*

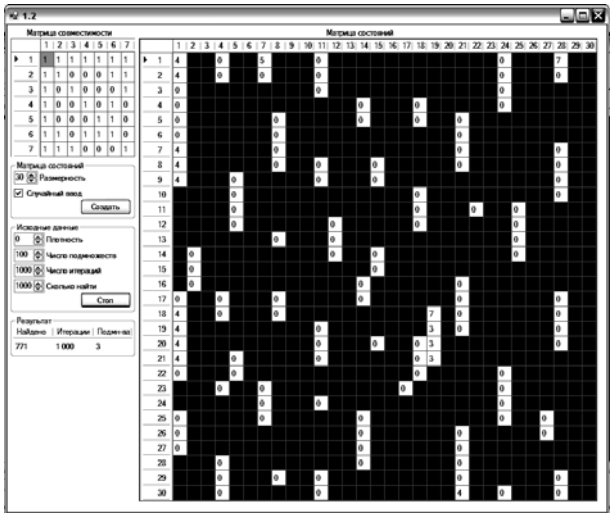

Рис. 3. Результат работы алгоритма для 100 подмножеств: темный цвет – элементы найденного подмножества

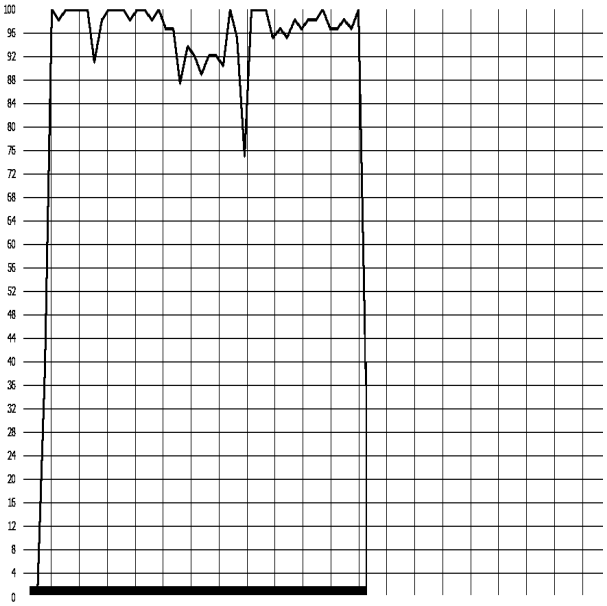

Рис. 4. Загрузка процессора при работе алгоритма для 100 подмножеств: верхняя линия – процент загрузки ЦП; нижняя – время работы

На рис. 5, 6 изображён результат работы алгоритма для той же таблицы, одного подмножества и плотности 0.

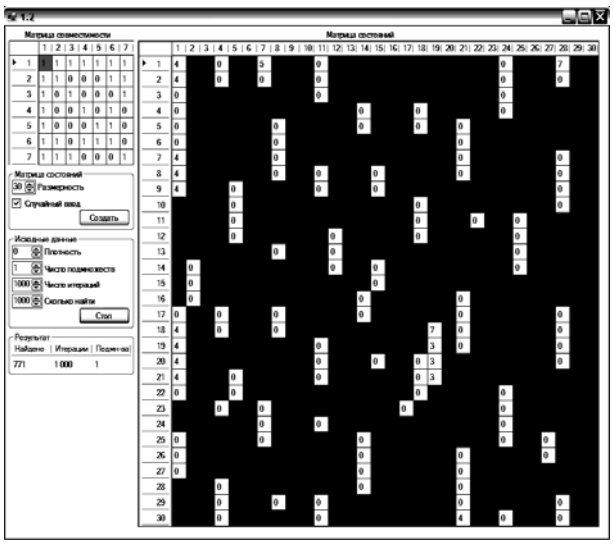

Рис. 5. Результат работы алгоритма для одного подмножества: темный цвет – элементы найденного подмножества

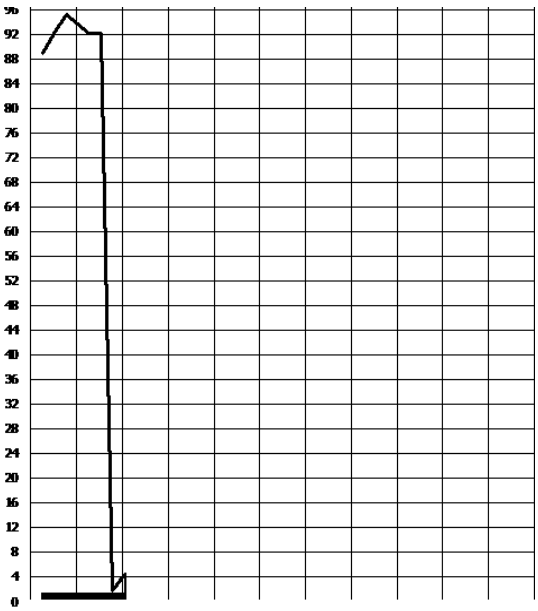

Рис. 6. Загрузка процессора при работе алгоритма для одного подмножества: верхняя линия – процент загрузки ЦП; нижняя – время работы

Сравнивая рис. 3 и 5, видим, что алгоритм нашёл то же самое подмножество элементов.

Как видно из графиков на рис. 4 и 6, время работы алгоритма для одного подмножества меньше т.к. шаг 2 сразу пропускается.

Но чем меньше подмножеств будет сгенерировано на первом шаге работы алгоритма, тем меньше вероятность найти максимальное подмножество совместимых элементов.

Заключение

При рассмотрении работы алгоритма для одного подмножества искомое подмножество могло состоять из одного элемента, например для таблицы элементов, изображённой на рис. 3, таким подмножеством может быть элемент с координатами (1, 28).

С другой стороны, если взять слишком большое число подмножеств, алгоритму потребуется большое время для выполнения второго шага. Однозначно определить количество подмножеств нельзя, всё зависит от входных данных.

Литература

1. Тюрин С.Ф. Синтез адаптируемой к отказам цифровой аппаратуры с резервированием базисных функций / С.Ф. Тюрин // Приборостроение. – 1999. – № 1. – С. 36-39.

2. Тюрин С.Ф. Адаптация к отказам одновыходных схем на генераторах функций с функционально-полными толерантными элементами / С.Ф. Тюрин // Приборостроение. – 1999. – № 7. – С. 32-34.

3. Тюрин С.Ф. Проблема сохранения функциональной полноты булевых функций при «отказах» аргументов / С.Ф. Тюрин // Автоматика и телемеханика. – 1999. – № 9. – С. 176-186.

4. Тюрин С.Ф., Несмелов В.А., Харитонов В.А. и другие. Программируемое логическое устройство. - Патент РФ № 2146840. - Опубл. БИ № 8. - 2000 г.

5. Тюрин С.Ф. Функционально-полные толерантные цифровые схемы на базе ПЛИС фирмы «ALTERA» / С.Ф. Тюрин, С.В. Богатырев, А.В. Голубев, А.В. Греков, А.А. Прохоров, Д.А. Прохоров // Радіоелектронні і комп'ютерні системи. – 2007. – № 8 (27). – С. 66-70.

Поступила в редакцию 16.02.2009

Рецензент: д-р техн. наук, проф. А.М. Романкевич, Национальный технический университет Украины «Киевский политехнический институт», Киев, Украина.

АЛГОРИТМ ПОШУКУ ПРАЦЕЗДАТНИХ ЕЛЕМЕНТІВ В ОТКАЗОУСТОЙЧИВЫХ ЦИФРОВИХ СХЕМАХ

С.Ф. Тюрин, А.В. Греков, Г.О. Ольт

У статті викладені результати розробки алгоритму пошуку працездатних підмножин множини функціонально-повних толерантних елементів, використовуваних для синтезу відмовостійких програмувальних логічних інтегральних схем. "Внутрішнє" резервування найбільш краще в програмувальних логічних устроях з настроюванням зв'язків запропонованих функціонально-повних толерантних елементів, що зберігають функціональну повноту при заданій моделі відмов, що й забезпечують працездатність на підмножині базисів підмножини елементів. Алгоритм дозволяє визначати працездатні підмножини елементів у випадку відмов цифрових схем для наступного проведення реконфигурации з метою забезпечення гарантованого виконання обчислювального процесу при припустимім зниженні продуктивності.

Ключові слова: функціонально-повні толерантні цифрові схеми, відмовостійкість, базисні функції, вектори.

SEARCH ALGORITHM FOR OPERATIONAL ELEMENTS IN THE FAILURE-RESISTANT DIGITAL CIRCUITS

S.F. Tyurin, A.V. Grekov, G.O. Olt

In the article are presented the results of developing the search algorithm for the operational subsets of the set of the full-function tolerant elements, utilized for the synthesis of the failure-resistant programmable logic devices. “Internal” redundancy most preferably in the programmable logic devices with tuning of the connections of the full-function tolerant elements, which preserve functional completeness with the assigned model of failures and of the ensuring fitness for work on the subset bases of the subset of elements. Algorithm makes it possible to determine the operational subsets of elements in the case of the failures of digital circuits for the subsequent conducting of reconfiguration for the purpose of the guarantee of the guaranteed fulfillment of computational process with the permissible reduction in the productivity.

Key words: full-function tolerant digital circuits, failure resistance, basic functions, vectors.

Тюрин Сергей Феофентович – д-р техн. наук, проф., проф. кафедры автоматики и телемеханики Пермского государственного технического университета, Пермь, Россия, e-mail: tyurinsergfeo@rambler.ru.

Греков Артём Владимирович – преподаватель кафедры программного обеспечения вычислительной техники и автоматизированных систем Пермского военного института внутренних войск МВД РФ, Пермь, Россия, e-mail: grekartemvl@mail.ru.

Ольт Георгий Олегович – студент 3 курса механико-математического факультета Пермского государственного университета, Пермь, Россия.