УДК 681.32

С.С. УВАРОВ

Институт Проблем управления им. В.А. Трапезникова РАН, Россия

ОСНОВНЫЕ ПРИНЦИПЫ СОЗДАНИЯ ОТКАЗОУСТОЙЧИВЫХ СИСТЕМ НА ПЛИС

В работе предложена новая методика построения отказоустойчивых систем на ПЛИС. Одним из основных преимуществ этой методики является сохранение длин связей в комбинационных схемах.

ПЛИС, разбиение, реконфигурация, ячейка, связь, блок

Введение

К настоящему моменту вышло в свет большое количество статей, посвященных проблеме отказоустойчивости систем на ПЛИС [1 – 5]. Интерес к данной проблеме вполне понятен, поскольку ПЛИС находят все более широкое применение в аппаратуре специального применения, к которой предъявляются самые жесткие требования надежности и устойчивости к отказам.

Несмотря на большой объем материала по данной теме, можно утверждать, что на сегодняшний день проблема построения отказоустойчивых систем на ПЛИС не решена. В большинстве работ ПЛИС рассматривается в слишком абстрактном виде «матрицы одинаковых ячеек». При этом игнорируются проблемы изменения длин связей между ячейками при реконфигурации, и не рассматривается возможность отказа самих связей. Изменение длин связей влечет за собой изменение задержек распространения сигналов, что может привести к потере работоспособности схемы. Трассировочные ресурсы занимают по объему гораздо больше места в кристалле по сравнению с ячейками, и, следовательно, ими нельзя пренебрегать при моделировании отказов.

Работа поддержана грантом РФФИ № 05-08-01388

В данной работе предлагается принципиально новый подход к проблеме проектирования отказоустойчивых систем на ПЛИС. В основе метода лежит предположение о том, что при групповом перемещении ячеек с сохранением относительного расположения ячеек в группе связи между этими ячейками не изменяют свою длину. Это предположение вполне верно, например, для ПЛИС XILINX семейства VIRTEX и SPARTAN.

Основные принципы проектирования

Далее рассматриваются отказы ячеек (CLB в терминах XILINX) и связей. Под термином «реконфигурация» подразумевается изменение размещения элементов схемы по ячейкам, ведущее также и к изменению трассировки связей между ячейками. Целью реконфигурации является исключение отказавшей ячейки (связи) из множества ячеек (связей), задействованных для реализации схемы. Предполагается, что сохранение длин связей в комбинационных схемах является необходимым и достаточным условием сохранения работоспособности схемы. Для сохранения длин связей комбинационные схемы помещаются в *блоки*, входы и выходы блоков делаются синхронными (посредством D-триггеров). В случае отказа блоки перемещаются относительно матрицы ПЛИС, при этом относительное размещение и трассировка *внутри* каждого блока сохраня-

ются. Перемещение блоков приводит лишь к изменению *внешних* связей между блоками. Но изменение длин внешних связей в широком диапазоне $(\sim 0, \sim T)$ не влияет на работу схемы, поскольку источником и приемником сигнала на внешней связи являются синхронные элементы (D-триггеры). Здесь T – период тактового сигнала. Для того, чтобы при отказе любой ячейки (связи) была возможна реконфигурация (т.е. перестановка блоков), необходимо, чтобы эти блоки имели вполне определенные размеры. Далее предлагается алгоритм, позволяющий достаточно гибко задавать размеры блоков и при этом гарантированно получать отказоустойчивую систему (т.е. систему, которая способна парировать отказ произвольной ячейки (связи) путем перестановки блоков).

Прежде чем прейти к описанию алгоритма разбиения задающего размеров блоков, дадим упрощенный вариант разбиения, который, как далее станет очевидно, является частным случаем общего алгоритма разбиения. Упрощенное разбиение представляет собой разбиение матрицы ПЛИС на блоки, при котором размер каждого последующего блока равен половине имеющегося свободного места, и при этом на каждом шаге делается сечение только по одному измерению. Рис. 1 иллюстрирует упрощенный алгоритм разбиения. Градациями серого показаны сформированные блоки, а белым – резервные ячейки.

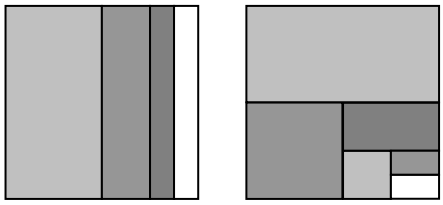

Рис. 1. Упрощенное разбиение

Очевидно, что такой способ разбиения позволяет путем перестановки блоков сделать любую отказавшую ячейку незадействованной. При этом упрощенный алгоритм разбиения обладает существенными недостатками, а именно: размеры блоков жестко фиксированы и подразумевается, что линейные размеры матрицы равны некоторой степени двойки. Далее предлагается обобщенный алгоритм разбиения, который избавлен от указанных недостатков.

Алгоритм разбиения

Для начала введем необходимые в дальнейшем обозначения:

X_1, X_2 – оси системы координат;

C – ячейка матрицы ПЛИС;

M – матрица ПЛИС;

B^k – k-й блок;

$\vec{B}^k = \begin{pmatrix} B_1^k \\ B_2^k \end{pmatrix}$ – вектор, определяющий размеры k-го блока;

$\vec{L}\left(C^j\right) = \begin{pmatrix} L_1\left(C^j\right) \\ L_2\left(C^j\right) \end{pmatrix}$ – вектор, определяющий расположение j-й ячейки

$\vec{L}\left(B^k\right) = \begin{pmatrix} L_1\left(B^k\right) \\ L_2\left(B^k\right) \end{pmatrix}$ – вектор, определяющий расположение k-го блока.

Под расположением k-го блока подразумевается расположение ячейки $C^l \in B^k$ такой, что ее координаты являются наименьшими среди всех ячеек $C \in B^k$, т.е., $\vec{L}\left(B^k\right) = \vec{L}\left(C^l\right)$, где $C^l : \forall C^j \in B^k$ верно, что $\vec{L}\left(C^l\right) < \vec{L}\left(C^j\right)$.

Введем также номера координат $i \in \{1,2\}$ и $j \in \{1,2\}$. Такое обозначение будет удобно в том случае, если требуется описать какую-либо из координат, не конкретизируя, какая именно это координата. Например, запись «$B_i^m = B_i^k$ и $B_j^m = \frac{1}{2} B_j^k$, где $i \in \{1,2\}$, $j \in \{1,2\}$ и $i \neq j$» следует читать так: «размеры m-го и k-го блоков равны по одной (**ка-**

кой-либо) координате, а по **другой** координате *m*-й блок вдвое меньше *k*-го».

Обозначим через N общее количество блоков, полученное при разбиении матрицы ПЛИС M.

Введем так же понятие *остаточного множества* ячеек S^k, которое есть множество всех ячеек, не принадлежащих ни одному из блоков с номерами от 1 до k т.е., $S^k = \left\{C^{Sj}\right\}$: для $\forall C^{sj} \in S^k$ верно, что $C^{Sj} \notin B^i \forall i : 1 \le i \le k$. Как в дальнейшем станет очевидно, ячейки, принадлежащие S^k, всегда будут образовывать прямоугольную область в матрице *M*, поэтому будет корректно ввести обозначения размеров S^k через вектор $\vec{S}^{\,k} = \begin{pmatrix} S_1^k \\ \vec{S}_2^k \end{pmatrix}$, и расположения через вектор $\vec{L}\left(S^k\right)$.

Можно утверждать, что множество $\mathrm{B} = \left\{\vec{B}^k, k = \overline{1,N}\right\}$ однозначно определяет *разбиение* матрицы M на блоки, а множество $\Lambda = \left\{\vec{L}^k, k = \overline{1,N}\right\}$ однозначно определяет *расположение* блоков в матрице M. Таким образом, алгоритм разбиения матрицы на блоки есть алгоритм нахождения множества B, а алгоритм реконфигурации есть алгоритм нахождения множества Λ, такого, что для любой отказавшей ячейки C^f верно, что $C^f \in S^N$ или, иначе говоря, что $C^f \notin B^k, \forall k = \overline{1,N}$ (при этом, естественно, полагается, что блоки не выходят за пределы матрицы и не накладываются друг на друга). Дадим описание алгоритма разбиения матрицы M на блоки B^k.

АЛГОРИТМ РАЗБИЕНИЯ

Подготовка: $S^0 = M$, $k = 0$, $\vec{L}\left(S^o\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

Шаг k: произвольно выбираем $i \in \left\{\overline{1,2}\right\}$, задаем размеры ($k$+1)-го блока $\vec{B}^{k+1} = \begin{pmatrix} B_1^{k+1} \\ B_2^{k+1} \end{pmatrix}$ так, чтобы они удовлетворяли условиям:

$$B_i^{k+1} \le \frac{1}{2} S_i^k,$$

$$B_j^{k+1} = S_j^k, j \in \left\{\overline{1,2}\right\}, j \ne i.$$

Располагаем B^{k+1} так, что $\vec{L}\left(B^{k+1}\right) = \vec{L}\left(S^k\right)$.

Тогда остаточное множество ячеек S^{k+1} будет иметь размеры

$$S_i^{k+1} = S_i^k - B_i^k,$$

$$S_j^{k+1} = S_j^k, j \in \left\{\overline{1,2}\right\}, j \ne i,$$

и расположение $\vec{L}(S^{k+1})$:

$$L_i(S^{k+1}) = L_i\left(S^k\right) - B_i^{k+1},$$

$$L_j\left(S^{k+1}\right) = L_j(S^k).$$

Количество сформированных блоков:

$$N = k + 1.$$

Если все ячейки C^l, задействованные в схеме, помещаются в сформированные блоки, т.е. $\forall C^l \in \bigcup_{k=1}^{N} B^k$, *то:* **ВЫХОД,** *иначе:* $k = k + 1$ делаем **Шаг k**. •

Данный алгоритм разбиения описывает не только получение множества B, размеров блоков, но и одно из возможных их расположений. Это сделано для того, чтобы, во-первых, показать, что по построению все блоки *можно* разместить в матрице M (без наложений). И, во-вторых, алгоритм разбиения дает некоторую начальную конфигурацию Λ^0 (назовем ее *канонической конфигурацией*), относительно которой в дальнейшем будет задан алгоритм реконфигурации. Следует обратить внимание, на то, что алгоритм разбиения обеспечивает следующее свойство: $\left(S^{k+1} \cup B^{k+1}\right) \subset S^k : \forall k = \overline{1,..,(N-1)}$.

Теперь опишем алгоритм реконфигурации (перестановок блоков).

Для этого введём преобразование $\rho(k, \Lambda)$, которое преобразует множество Λ во множество

$\tilde{\Lambda} = \rho(k, \Lambda)$ определенным образом в зависимости от k. А именно, $\rho(k, \Lambda)$ есть такое преобразование, при котором

$$\tilde{\vec{L}}\left(B^k\right) \equiv \rho(\vec{L}(B^k), k) = \vec{L}(B^k) + \begin{pmatrix} \delta(B_1^k, S_1^k) \cdot S_1^k \\ \delta(B_2^k, S_2^k) \cdot S_2^k \end{pmatrix} \quad -$$

новое расположение *k*-го блока;

$\tilde{\vec{L}}\left(S^k\right) = \vec{L}\left(B^k\right)$ – новое положение остаточного множества ячеек S^k;

$\tilde{\vec{L}}\left(B^j\right) = \vec{L}\left(B^j\right)$ для $\forall j < k$ – блоки с номерами меньше k не перемещаются.

Здесь через δ обозначена δ-функция, обладающая следующим свойством: $\delta(a,b) = 1, \forall a, b : a = b$ и $\delta(a,b) = 0, \forall a, b : a \neq b$. Здесь и далее под перемещением блока, либо остаточного множества подразумевается перемещение всех входящих в него ячеек. Поскольку $B^l \subset S^k, \forall l > k$, то при перемещении S^k также перемещаются и все блоки с номерами большими *k*. Будем обозначать перемещенные блоки, ячейки и остаточные множества $\rho(k, \bullet)$. Преобразование $\rho(k, \Lambda)$ существует для любого $k = \overline{1, N}$, и любых разбиения B и размещения Λ, полученных для матрицы *M* с помощью вышеописанного алгоритма разбиения. Важно отметить, что перемещенный преобразованием ρ блок $\tilde{B}^k = \rho(k, B^k)$ не включает в себя ни одной ячейки из тех, которые принадлежали ему до перемещения, т.е. для $\forall k = \overline{1, N}, \forall C^l \in B^k, C^l \notin \tilde{B}^k = \rho(k, B^k)$. Это непосредственно следует из того, что $\rho\left(k, B^k\right) \subset S^k$. Предположим, что произошел отказ в ячейке $C^f \in B^N$, тогда преобразование $\rho(k, \Lambda)$ осуществляет перемещение одного лишь блока B^N таким образом, что $C^f \in S^N$, т.е., C^f не принадлежит ни одному из блоков. Теперь предположим, что отказала ячейка в произвольном блоке B^k. Тогда преобразование $\rho(k, \Lambda)$ осуществляет перестановку блоков таким образом, что $C^f \in \rho(S^k, k)$. Это в свою очередь означает, что C^f принадлежит либо одному из блоков с номером большим *k*, либо что $C^f \in \rho(S^N, k)$. Если $C^f \in \rho(S^N, k)$, то мы имеем искомую реконфигурацию, иначе $C^f \in \rho(B^l, k), l > k$. В последнем случае применяем преобразование $\rho(l, \Lambda)$, и т.д. до тех пор, пока не получим $C^f \in \rho(\rho(...\rho(S^N, l^m), l^{m-1}), l^1)$. Поскольку для любого блока преобразование ρ существует, то также существует и суперпозиция подобных преобразований, приводящая в итоге к искомой реконфигурации. При этом очевидно, максимальное количество преобразований не превосходит N.

Реконфигурация представляет собой перестановку блоков канонической конфигурации. Далее приведем алгоритм, который определяет перемещения блоков, в случае отказа произвольной ячейки.

АЛГОРИТМ РЕКОНФИГУРАЦИИ

Подготовка: $k = 1$, $p = 1$.

Стадия 1: Определяем, номер блока k, в котором расположена отказавшая ячейка:

Шаг k: *Если* $\exists i \in \{1,2\} : L_i\left(C^f\right) > B_i^k \Rightarrow C^f \in S^k$ *то* {*если* $k = N$ *то* **ВЫХОД**, *иначе*} $L_i\left(C^f\right) = L_i\left(C^f\right) - B_i^k$, $L_j\left(C^f\right) = L_j\left(C^f\right)$, $k = k + 1$ повторяем **Шаг 1k**, *иначе* **Стадия 2**.

Стадия 2: Определяем суперпозицию преобразований $\rho\left(n^1, ..., n^m\right) \equiv \rho\left(n^1, \rho\left(..., \rho\left(n^m, \Lambda\right)\right)\right)$, необходимую для того, чтобы $C^f \in \rho\left(n^1, ..., n^m, S^N\right)$.

Шаг p: $n^p = k$. Запоминаем n^p. Применяем преобразование $\rho(n^p, \Lambda)$: $\Lambda = \rho\left(n^p, \Lambda\right)$. $k = k + 1$

Делаем **Шаг k** из **Стадии 1** •

Приведенный алгоритм реконфигурации определяет набор перестановок блоков в случае отказа

произвольной ячейки. Для того чтобы предложенный алгоритм был применим при автоматизированном размещении схемы в матрицу ПЛИС, необходимо описать изменение расположения элементов схемы в ячейках ПЛИС, т.е., фактически, определить изменение координат ячеек. Алгоритм реконфигурации дает набор $\left\{n^j\right\}$ параметров перестановок ρ. Необходимо описать перестановку ячеек через $\left\{n^j\right\}$. Все ячейки, принадлежащие блоку B^k, перемещаются в точности так же, как и B^k в целом. Введем вектор перемещения $\Delta\vec{L}(\bullet,k)=\rho\left(\vec{L}(\bullet),k\right)-\vec{L}(\bullet)$, тогда

$$\Delta\vec{L}\left(C^j,k\right)=\Delta\vec{L}\left(B^k,k\right),\forall C^j\in B^k.$$

поскольку

$$\rho(\vec{L}(B^k),k)=\vec{L}(B^k)+\begin{pmatrix}\delta(B_1^k,S_1^k)\cdot S_1^k\\ \delta(B_2^k,S_2^k)\cdot S_2^k\end{pmatrix}$$

и

$$\rho(\vec{L}(B^l),k)=\rho(\vec{L}(S^k),k)=\vec{L}(B^k),\forall l:k>l<N$$

то

$$\Delta\vec{L}\left(C^j,k\right)=\rho(\vec{L}(B^k),k)-\vec{L}(B^k)=$$
$$=\begin{pmatrix}\delta(B_1^k,S_1^k)\cdot S_1^k\\ \delta(B_2^k,S_2^k)\cdot S_2^k\end{pmatrix},\forall C^j\in B^k$$

и

$$\Delta\vec{L}\left(C^i,k\right)=\vec{L}(B^k)-\left(\vec{L}(B^k)+\begin{pmatrix}\delta(B_1^k,S_1^k)\cdot S_1^k\\ \delta(B_2^k,S_2^k)\cdot S_2^k\end{pmatrix}\right)=$$
$$=-\begin{pmatrix}\delta(B_1^k,S_1^k)\cdot S_1^k\\ \delta(B_2^k,S_2^k)\cdot S_2^k\end{pmatrix},\forall C^i\notin B^k.$$

Поскольку преобразование $\rho\left(k,\Lambda\right)$ изменяет лишь только расположение блоков с номерами $\geq k$, то

$$\Delta\vec{L}\left(C^p,k\right)=\vec{0},\forall C^p\in B^h,h<k.$$

В итоге получаем, что для произвольной ячейки вектор перемещения определяется суммой векторов всех перемещений для блока, в котором расположена данная ячейка:

$$\Delta\vec{L}^w=\sum_{\left\{n^i\right\}}\Delta\vec{L}\left(B^k,n^i\right),$$

где суммирование ведется по набору параметров $\left\{n^i\right\}$, найденному по алгоритму реконфигурации.

Оценка эффективности

Для оценки эффективности алгоритма введем следующие величины:

T – общее число ячеек в матрице ПЛИС;

U – число занятых (задействованных в схеме) ячеек матрицы ПЛИС;

F – число свободных (незадействованных в системе) ячеек в матрице ПЛИС;

коэффициент заполнения, K=U/T – отношение числа заполненных ячеек к общему числу ячеек в матрице ПЛИС;

коэффициент резервирования S=F/T – отношение числа свободных ячеек к общему числу ячеек в матрице;

коэффициент избыточности R=F/U – отношение числа свободных ячеек к числу занятых ячеек.

Понятия коэффициента заполнения и коэффициента резервирования могут быть применены и к обычной (не отказоустойчивой) системе на ПЛИС, так как они характеризуют эффективность заполнения ПЛИС. Понятие коэффициента избыточности вводится специально для отказоустойчивой системы. Коэффициент избыточности показывает, во сколько раз требуется увеличить размеры системы (число ячеек) для того, чтобы она стала отказоустойчивой. При этом необходимо понимать, что коэффициент избыточности имеет практическое значение только лишь в том случае, если в систему действительно вносится избыточность.

Если же в качестве резервных элементов используются ранее не задействованные элементы (ячейки), то смысл коэффициента избыточности становится весьма абстрактным.

Рассмотрим отказоустойчивую систему, построенную по упрощенному алгоритму, который был предложен в начале статьи. Будем полагать, что все блоки заполнены на 100%. Для системы, состоящей из n блоков, коэффициент заполнения:

$$K = \frac{2^n - 1}{2^n}.$$

Таким образом, если изначально имеется система в ПЛИС, имеющая коэффициент заполнения K, и требуется сделать эту систему отказоустойчивой, то минимальное число блоков, необходимое для реализации системы, определяется как $n = \left\lceil \log_2\left(\frac{1}{1-K}\right)\right\rceil$, где $\lceil x \rceil$ – ближайшее целое число, не меньшее x.

Очевидно, что выгодно разбить систему на минимально возможное число блоков, так как в этом случае количество изменяемых связей и число всевозможных реконфигураций будут минимальными. В табл. 1 приведена взаимозависимость коэффициентов заполнения, резервирования и избыточности, количества блоков и числа реконфигураций. Из таблицы видно, что при крайне малом числе блоков можно реализовать систему с весьма высоким коэффициентом заполнения. Однако, полный набор реконфигураций, необходимый для парирования произвольного отказа, очень быстро растет с увеличением числа блоков. Таким образом, необходим определенный компромисс между коэффициентом заполнения и сложностью отказоустойчивой системы (в смысле величины полного набора реконфигураций).

Резервирование внешних связей

Теперь определим, каким образом будут изменяться связи при реконфигурациях системы.

Для простоты рассмотрим упрощенное разбиение, показанное на рис. 1 справа. Как уже было сказано выше, все связи внутри блоков остаются неизменными при их перемещениях. Меняются лишь связи с буферами ввода/вывода и связи между блоками. На рис. 2 показан полный набор реконфигураций для системы, состоящей из 3 блоков. Этот рисунок показывает изменения связей при всевозможных перестановках блоков.

Для того чтобы любая конфигурация из полного набора реконфигураций была возможна, необходимо, чтобы существовали все связи между блоками. Для трассировки связей как внутри блоков, так и между блоками используются одни и те же трассировочные ресурсы. Поэтому не исключена ситуация, когда после реконфигурации (перестановки блоков) некоторые связи окажутся нереализуемыми.

Таблица 1

Оценка эффективности

K	K(%)	n (число блоков)	Количество реконфигураций	S	S(%)	R	R(%)
1/2	50%	1	1	1/2	50%	1	100%
3/4	75%	2	4	1/4	25%	1/3	33,3%
7/8	87,5%	3	8	1/8	12,5%	1/7	14,3%
15/16	93,8%	4	16	1/16	6,2%	1/15	6,6%
31/32	96,9%	5	32	1/32	3,1%	1/31	3,2%
63/64	98,4%	6	64	1/64	1,6%	1/63	1,6%
127/128	99,2%	7	128	1/128	0,8%	1/127	0,8%

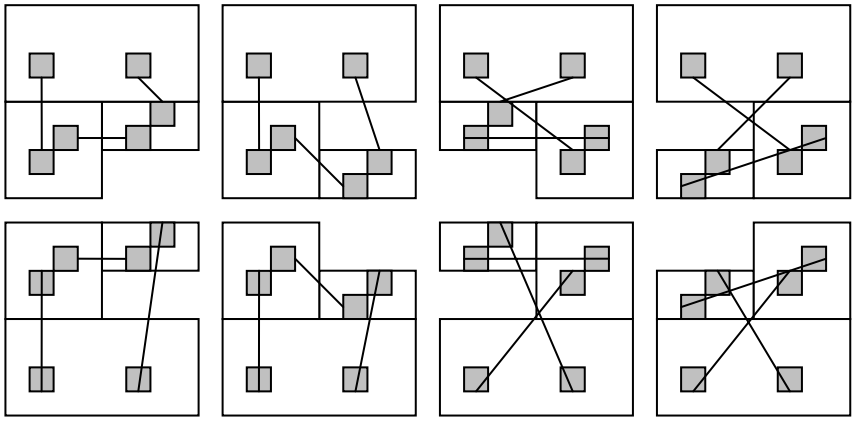

Рис. 2. Полный набор реконфигураций для трех блоков

Это возможно из-за того, что все трассировочные ресурсы, позволяющие соединить выход одного блока со входом другого, могут оказаться заняты внутренними связями блока. Для избежания подобных проблем предлагается зарезервировать все возможные внешние связи для каждой из конфигураций на начальном этапе проектирования, и только лишь после этого приступить к размещению и трассировке внутренней «начинки» блоков. При этом полагается, что общее количество трассировочных ресурсов ПЛИС в результате резервирования уменьшится незначительно. Необходимо определить, во сколько раз увеличится число связей между блоками в результате резервирования. Для этого присвоим блокам номера, начиная с «1», в порядке уменьшения их размера. Для общности будем полагать, что все буферы ввода/вывода *виртуально* относятся к блоку с номером «1».

Будем говорить, что связь имеет *длину L*, если она соединяет два блока с номерами N_1 и N_2, и $|N_1 - N_2| = L$. Таким образом, блоки 1 и 2 связаны связями длиной 1, блоки 2 и 4 – связями длиной 2 и т.д., блок 1 соединяется с буферами ввода/выводя связями длины 0, блок 2 с буферами соединяется связями дины 1 и т.д. Из рис. 1 видно, что блок 1 имеет два возможных расположения в матрице ПЛИС, блок 2 – четыре, и т.д. Из рис. 1 также понятно, что блоки с номерами, отличающимися на 1, могут иметь 4 различных *взаимных* расположения (один относительно другого). Каждое взаимное расположение порождает соответствующую трассировку логических связей. Таким образом, для всех возможных реконфигураций логическая связь длины 1 требует резервирования 4 различных маршрутов. Рассуждая аналогичным образом, получаем, что для связей длиной 2 требуется 8 резервных маршрутов, для связей длиной 3 – 16 резервных маршрутов, и вообще, для связи длины L требуется резервирование $2^{(L+1)}$ маршрутов. Те же рассуждения применимы и к связям блоков с буферами ввода/вывода. Под связью длины «0» подразумевается **только лишь** связь между блоком 1 и буферами ввода/вывода (**к внутренним связям блоков введенное выше понятие длины не применяется**).

Таким образом, очевидно, что система должна быть декомпозирована на блоки таким образом, чтобы по возможности минимизировать общее количество связей между блоками. Причем связей с большой длиной должно быть как можно меньше, поскольку они порождают большее количество резервных маршрутов.

Обобщение на дополнительные функциональные узлы ПЛИС

Выше была описана методика построения системы, устойчивой к отказу произвольной ячейки, или связи, соединяющей ячейки. Однако современные ПЛИС помимо ячеек (CLB) могут содержать боль-

шое количество блоков памяти, умножителей, АЛУ и в будущем, возможно, еще каких либо дополнительных ресурсов. Так, например, ПЛИС VIRTEX-4 SX55 помимо CLB содержит 512 блоков АЛУ (DSP-SLISE) и столько же блоков двухпортовой памяти. Принципиально, что эти дополнительные функциональные блоки расположены в виде матрицы, и весь кристалл ПЛИС можно разбить на одинаковые секции, содержащие один дополнительный блок и некоторое количество ячеек. К полученным секциям можно применить алгоритмы разбиения и реконфигурации. Таким образом, можно теоретически задействовать все дополнительные блоки ПЛИС, за исключением всего лишь одного, резервного.

Заключение

В работе была предложена методика проектирования отказоустойчивых систем на базе ПЛИС. Под отказоустойчивостью в данном случае подразумевалась возможность реконфигурации системы, в случае отказа произвольной ячейки (или связи) позволяющая восстановить работоспособность системы путем перезагрузки конфигурационного файла ПЛИС. При этом особое внимание было уделено точному сохранению длин связей (задержек) в системе. Именно это, по мнению автора, и является **необходимым** условием, обеспечивающим работоспособность системы после реконфигурации. Сохранение задержек в комбинационных схемах **является принципиальным отличием** предложенной методики от всех ранее известных.

В работе не были рассмотрены методики диагностики, позволяющие найти отказавшую ячейку (связь) в ПЛИС. Подразумевается, что эти методики известны.

На современном этапе развития САПР проектирования ПЛИС предполагается, что заранее подготавливаются все возможные варианты конфигураций ПЛИС (в виде конфигурационных файлов), и в случае отказа загружается необходимый конфигурационный файл. Здесь отказоустойчивость можно также понимать и как ремонтопригодность. В случае выхода из строя отдельной ячейки в ПЛИС нет необходимости проводить технологически сложные работы, связанные с ее заменой. Вообще, не требуются никакие «механические» действия (кроме, разве что, подключения к разъёму JTAG). Ремонт системы вообще может быть осуществлен дистанционно, что, например, актуально для спутниковых систем. Также предложенная методика может быть использована и при реализации обычных (не отказоустойчивых систем) на кристаллах с заводскими дефектами.

Литература

1. John Lach, William H. Mangione-Smith, Miodrag Potkonjak. Low Overhead Fault-Tolerant FPGA Systems // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – June 1998. – Vol. 6, No. 2.

2. John Lach, William H. Mangione-Smith, Miodrag Potkonjak. Efficiently Supported Fault-Tolerance in FPGA // Presented at FPGA'98 Monterey CA, 1998. – February 22-24.

3. Abderrahim Doumar, Hideo Ito. Detecting, Diagnosing and Tolerating Faults in SRAM-Based Field Programmable Gate Arrays: A Survey // IEEE Transactions on Very Large Scale Integration (VLSI) Systems, June 2003. – Vol. 11, No. 3.

4. Jing Haung, Mehdi Baradaran Tahoori, Fabrizio Lombardi. Routability and Fault Tolerance of FPGA Interconnect Architectures. – Dept. of Electrical and Computer Engineering Northeastern University Boston MA 02115.

5. Vijay Lakamraju and Russell Tessier. Tolerating Operational Faults in Cluster-based FPGAs. – Dept. of Electrical and Computer Engineering University of Massachusetts Amherst, MA 01003.

Поступила в редакцию 2.02.2007

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