МИКРОЭЛЕКТРОНИКА, 2021, том 50, № 6, с. 467–480

СХЕМОТЕХНИКА

УДК 621.3.049.771.14:004.021/.415

ПРОЕКТИРОВАНИЕ ПЛИС И РЕКОНФИГУРИРУЕМЫХ СнК С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ ПРОГРАММНОГО АНАЛИЗА И ПРОТОТИПИРОВАНИЯ

© 2021 г. В. И. Эннс[a], *, С. В. Гаврилов[b], **, В. М. Хватов[b], *, В. Г. Курбатов[a]**

[a] *Научно-исследовательский институт молекулярной электроники (АО "НИИМЭ"), ул. Академика Валиева, 6, стр. 1, Зеленоград, Москва, 124460 Россия*

[b] *Институт проблем проектирования в микроэлектронике Российской АН (ИППМ РАН), ул. Советская ул., 3, Зеленоград, Москва, 124365 Россия*

**E-mail: venns@niime.ru*

***E-mail: s.g@ippm.ru*

****E-mail: khvatov_v@ippm.ru*

Поступила в редакцию 14.05.2021 г.
После доработки 08.06.2021 г.
Принята к публикации 15.06.2021 г.

Описываются подходы и методы программного прототипирования программируемых логических интегральных схем (**ПЛИС**) и реконфигурируемых систем на кристалле (**РСнК**). Программное прототипирование, в отличие от классического прототипирования с применением готовых кристаллов ПЛИС, является частью нового этапа маршрута проектирования ПЛИС и реконфигурируемых СнК. Оно позволяет оценить эффективность реализаций пользовательских проектируемых схем и выбрать архитектуру базового кристалла до его фактического изготовления за счет оперативной настройки системы автоматизированного проектирования (**САПР**) на соответствующие изменения в конструкции, схемотехнике и топологии базового кристалла. Гибкость и динамичность настройки на требуемую архитектуру обеспечивается разработанной и описанной в данной статье формализацией представления схем в САПР, которая может быть применима как при анализе базового кристалла, так и при анализе пользовательских проектных схем.

DOI: 10.31857/S0544126921060077

I. ВВЕДЕНИЕ

Проектирование программируемых логических интегральных схем или реконфигурируемых систем на кристалле – сложный процесс, требующий большое количество времени на выбор параметров схемы, ее архитектуры, анализ трассировочных возможностей и моделирования ее компонентов [1–4]. Несмотря на то, что большинство этапов маршрута проектирования автоматизировано, оценивание трассируемости пользовательских схем и поиск слабых мест разработанной архитектуры выполняется человеком. Такой подход без должной верификации может привести к ошибке, обнаруженной только после этапа разработки топологии базовой схемы, что недопустимо в условиях ограниченных сроков.

Значительно упростить и ускорить процесс проектирования базового кристалла ПЛИС или РСнК позволяют методы программного анализа и оценки архитектуры. Существующие методы оценки, основанные на прохождении полного маршрута проектирования [5–9], используют упрощенное описание базовых схем, что не позволяет точно оценить их архитектуру, ее тонкости и слабые места. Также имеющиеся методы требуют описания схемы в специализированном формате, что ставит перед разработчиком архитектуры дополнительную задачу – изучение новых методов представления разработанного схемотехнического описания схемы.

Предложенный подход к оценке архитектуры ПЛИС и РСнК использует формат схемотехнического описания схем – CDL (Circuit Design Language) [10], который широко распространен среди разработчиков интегральных схем (**ИС**) и используется в системах автоматического проектирования компаний Cadence, Synopsys, Mentor Graphics. Описание схемы в данном формате автоматически генерируется из ее графического представления в схемотехническом редакторе указанных компаний. Поддержка языка CDL, наряду с разработанной формализацией представления схем в САПР, позволяет гибко провести оперативную настройку программного обеспечения на соот-

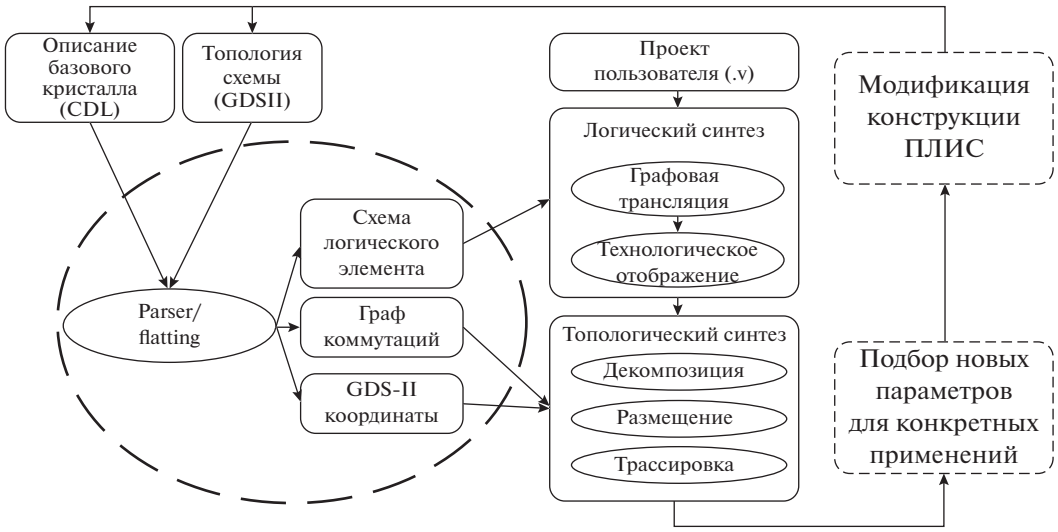

Рис. 1. Этапы программного прототипирования.

ветствующие изменения в конструкции, схемотехнике и топологии разрабатываемой гетерогенной СнК или ПЛИС и оценить как эффективность реализации различных пользовательских проектируемых схем, так и эффективность самой архитектуры базового кристалла.

Разработанный метод, включающий в себя оперативную настройку на изменения архитектуры схемы и оценку ее эффективности, является новым этапом маршрута проектирования реконфигурируемой или программируемой логики, называемый "*программным прототипированием*". В отличие от классического прототипирования, подразумевающего тестирование системы на кристалле или ее отдельных СФ-блоков в базисе готового кристалла ПЛИС [11, 12], программное прототипирование включает в себя тестирование и оценку самого базового кристалла, содержащего элементы ПЛИС, еще до его фактического изготовления.

II. МЕТОД ПРОГРАММНОГО ПРОТОТИПИРОВАНИЯ

Метод программного прототипирования, предложенный в данной работе, состоит из нескольких этапов (рис. 1).

1) Первым этапом является выбор и проектирование базовой архитектуры, на основе которой будут разрабатываться прототипы и выполняться дальнейшие модификации. Архитектура может быть как уникальной, так и выбранной из множества существующих решений, отличающихся структурой трассировочных ресурсов [13, 14] (островные и иерархические ПЛИС), структурой логического элемента (**ЛЭ**) и группы логических блоков (**ГЛБ**) (ЛЭ в ПЛИС фирмы Altera [15], конфигурационный логический блок в ПЛИС фирмы Xilinx [16], универсальные логические блоки (VersaTile) у фирмы Microsemi [17]). Выбрать архитектуру конечной схемы среди всего многообразия позволяют физические ограничения, заключающиеся в размере корпуса или в площади кристалла, необходимой для размещения конфигурационной памяти. Также ограничения накладываются исходя из требований, заданных конкретными пользовательскими проектами, которые выражаются в количестве программируемой логики, коммутационных возможностях схемы и определенном наборе СФ-блоков.

2) На втором этапе происходит передача информации о разработанной схеме в базы данных САПР. Первоначально, с помощью специализированного программного обеспечения, так называемого парсера, в САПР обрабатывается и анализируется схемотехническое описание схемы РСнК или ПЛИС в формате CDL и ее топология в формате GDSII [18]. С помощью обработки этих файлов структура программы автоматически подстраивается под архитектуру ПЛИС, формируя граф коммутаций, координаты ЛБ и карту памяти, на основании которой будет формироваться вектор прошивки. Возможность автоматической подстройки под любую архитектуру позволяет разработчикам РСнК и ПЛИС заранее оценивать их трассируемость и находить слабые места архитектуры, а разработчикам САПР заранее отлаживать ПО на будущей архитектуре под нужды заказчика. Это делает процесс разработки и конечный результат гораздо эффективнее.

3) Следующим этапом выполняется полный маршрут проектирования пользовательской схемы, включающий в себя логический и топологический синтез [19]. Логический синтез состоит из

графовой трансляции и технологического отображения в базисе целевого кристалла ПЛИС или РСнК [20, 21]. Топологический синтез, в свою очередь, состоит из декомпозиции списка соединений на отдельные группы или кластеры [22], размещения логических элементов на легальные позиции матрицы ПЛИС [23], трассировки соединений между ЛЭ с использованием коммутационных ресурсов, заложенных в архитектуре [24].

4) Заключающим этапом программного прототипирования является анализ полученных результатов, на основании которого подбираются новые параметры архитектуры и вносятся соответствующие модификации в схемотехническое описание базового кристалла. Программное прототипирование – процесс итерационный, поэтому этап внесения изменений в конструкцию схемы лишь завершает одну итерацию подбора подходящей архитектуры. Процесс прототипирования можно считать завершенным при соблюдении двух условий. Первое условие – соответствие результатов прототипирования всем заданным требованиям и ограничениям. Второе – успешное выполнение полного маршрута проектирования для ряда пользовательских проектных схем. В случае несоблюдения этих условий в архитектуру базового кристалла вносятся соответствующие изменения и процесс повторяется до получения положительного результата.

Далее в разделе III показаны особенности представления в САПР описаний базового кристалла и проектируемой схемы для выполнения программного прототипирования. В разделе III.a подробнее рассмотрен этап загрузки базового кристалла в САПР и показано разработанное формализованное представление схемотехнического описания конструкции ПЛИС и РСнК. В разделе III.b представлены особенности анализа и обработки в САПР топологии ПЛИС и РСнК. В разделе V содержатся практические результаты применения разработанного метода программного прототипирования базовой архитектуры ПЛИС, а также описываются изменяющиеся параметры архитектуры и характеристики, на основе которых выполнялось сравнение полученных прототипов.

III. ОСОБЕННОСТИ ПРЕДСТАВЛЕНИЯ БАЗОВОГО КРИСТАЛЛА И ПРОЕКТИРУЕМОЙ СХЕМЫ В САПР ДЛЯ ВЫПОЛНЕНИЯ ПРОГРАММНОГО ПРОТОТИПИРОВАНИЯ

а) Особенности представления списка соединений базового кристалла и проектируемой схемы

Оперативную настройку конструкции и схемотехники базового кристалла на новые потребности от конечного пользователя, а также оперативную настройку САПР на соответствующие изменения в конструкции, схемотехнике и топологии базового кристалла обеспечивает формализация соответствий между элементами базового проекта реконфигурируемой или программируемой гетерогенной системы на кристалле от производителя (базы) и пользовательскими проектируемыми схемами от конечного заказчика.

Для загрузки требуемой информации в САПР, схема базового проекта (реконфигурируемая или программируемая гетерогенная СнК или ПЛИС) представляется в виде описания в формате CDL, пользовательская проектируемая схема – на языке Verilog в формате плоского списка соединений.

В процессе обработки базовой схемы ее иерархическое описание определяется, как упорядоченная тройка:

$$\Pi = (S, L, s_m) - \text{иерархическое описание проекта}, \quad (1)$$

где $S = \{s_i, i = 1,\ldots,|S|\}$ – множество схем в иерархическом описании проекта;

$L \subset S$ – базис или подмножество базисных библиотечных подсхем для текущего уровня (или этапа) проектирования;

$s_m \in S$, $s_m \notin L$ – главная схема или схема верхнего уровня.

При этом каждое из схемных описаний в иерархии проекта определяется следующим образом:

$$\forall s \in S: s = (\mu(s), E(s), N(s), P(s), C(s)), \quad (2)$$

где: $\mu(s)$ – уникальное имя схемы (строка символов);

$E(s) = \{e_i, i = 1,\ldots,|E(s)|\}$ – множество элементов в схеме;

$N(s) = \{n_i, i = 1,\ldots,|N(s)|\}$ – множество цепей (узлов) в схеме;

$P(s) = \{p_i, i = 1,\ldots,|P(s)|\}$ – множество внешних выводов (контактов) схемы;

$C(s) = \{c_i, i = 1,\ldots,|C(s)|\}$ – множество соединений (коммутаций) схемы.

Множество элементов характеризуется следующими данными:

$$\forall e \in E(s): e = (\mu(e), m(e), P(e)), \quad (3)$$

где $\mu(e)$ – уникальное имя элемента (строка символов);

$m(e) \in S$ – модель элемента, представленная в иерархическом описании схемой следующего более низкого уровня иерархии;

$P(e) = \{p_i, i = 1,\ldots,|P(e)|\}$ – множество выводов элемента, совпадающее по составу с множеством внешних выводов модели (связанных с ним взаимно-однозначным соответствием):

$$P(e) \leftrightarrow P(m(e)); \quad |P(e)| = |P(m(e))|. \quad (4)$$

Множество внешних выводов схемы характеризуется следующими данными:

$$\forall p \in P(s)\colon\ p = (\mu(p), \tau(p)), \tag{5}$$

где $\mu(p)$ – уникальное имя вывода;

$\tau(p) \in \{\tau_{inp}, \tau_{out}, \tau_{bi}\}$ – тип вывода: вход, выход или двунаправленный.

Множество цепей (узлов) схемы характеризуется именем и соответствующим цепи набором соединений:

$$\forall n \in N(s)\colon n = \mu(n), \tag{6}$$

где $\mu(n)$ – имя цепи (узла), (строка символов);

$C(s)$ – множество соединений в схеме, определяемое как подмножество пар:

$$C(s) = \left\{(p,n)\colon\ p \in \left(\bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s)\right),\ n \in N(s)\right\} \tag{7}$$

таких, что, для любого контакта цепь единственная или не существует вовсе:

$$\begin{gathered}\forall p \in \left\{\bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s)\right\}\colon \\ (\exists! n \in N(s)\colon (p,n) \in C(s)) \vee \\ \vee\ (\forall n \in N(s)\colon (p,n) \notin C(s)).\end{gathered} \tag{8}$$

Другими словами, множество соединений в схеме определяется как однозначное отображение:

$$C^*(s) = \left\{\left(\bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s)\right) \to (N(s) \cup \emptyset)\right\}. \tag{9}$$

При этом обратное отображение определяет список соединений конкретной цепи и не может быть однозначным – количество соединений у каждой цепи должно быть не менее двух, в противном случае цепь будет считаться ошибочной или ложной:

$$\begin{gathered}C^{*-1}(s) = \left\{N(s) \to \left(\bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s)\right)\right\} \\ \forall n \in N(s)\colon \left|\{(p,n)\colon\ (p,n) \in C(s)\}\right| \geq 2.\end{gathered} \tag{10}$$

Как правило, внешний вывод может быть у цепи только один:

$$\begin{gathered}\forall n \in \\ \in N(s)\colon \left|\{(p,n)\colon\ (p,n) \in C(s) \wedge p \in P(s)\}\right| \leq 1.\end{gathered} \tag{11}$$

Отличие формализации базового проекта от формализации пользовательской схемы заключается в том, что имя внешнего контакта в описании схемы на языке CDL совпадает с именем соединенной с ним цепи:

$$\begin{gathered}\forall n \in N(s), \\ \forall p \in P(s)\colon (p,n) \in C(s) \Rightarrow \mu(n) = \mu(p).\end{gathered} \tag{12}$$

Для текущего этапа проектирования подсхемы базисного библиотечного уровня не содержат внутренних данных и представляют собой “черные ящики”:

$$\forall s \in L\colon E(s) = \emptyset,\ \ N(s) = \emptyset,\ \ C(s) = \emptyset. \tag{13}$$

В этом случае моделирование подсхем нижнего уровня выполняется на основе встроенных моделей, и само описание “черных ящиков” может быть скрыто от внешнего пользователя. Например, на схемотехническом уровне проектирования к базисному библиотечному уровню относятся транзисторы, емкости, сопротивления, индуктивности и др.

Иерархическое описание базового кристалла преобразовывается в САПР в соответствующее ему “плоское” представление путем рекурсивной flat-процедуры. Для заданного проекта $\Pi = (S, L, s_m)$ в “плоском” представлении сохраняются только те подсхемы, которые фактически применялись в s_m. Обозначим через $\varphi(s, s_t)$ логическую функцию, определенную на Декартовом произведении $S \times S$, принимающую значение 1 тогда и только тогда, когда s фактически используется в s_t:

$$\varphi\colon S \times S \to \mathfrak{B};\ \ \mathfrak{B} = \{0,1\};\ \ \varphi(s, s_t) = \left((s = s_t) \vee \left(\bigvee_{e \in E(s_t)} \varphi(s, m(e))\right)\right), \tag{14}$$

т.е., $\varphi(s,s_t)=1$, если $(s=s_t)$ или $\exists e\in E(s_t)$: $\varphi(s,m(e))=1$.

Для заданного проекта $\Pi=(S,L,s_m)$ "плоское" представление $\Pi_f(\Pi)=(S_f,L_f,s_f)$ строится по следующим правилам:

$$L_f=\{s:\ (s\in L)\wedge\varphi(s,s_m)\};$$
$$S_f=\{s_f\cup L_f\};$$
$$s_f=(\mu(s_f),E(s_f),N(s_f),P(s_f),C(s_f));\ \text{где:}$$
$$\mu(s_f)=\mu(s_m);$$
$$P(s_f)\leftrightarrow P(s_m).$$

Имена элементов $\mu(e), e\in E(s_f)$ и имена цепей $\mu(n), n\in N(s_f)$ в "плоском" представлении являются уникальными и содержат информацию об именах элементов более высоких уровней иерархии, в состав которых входят подсхемы, содержащие рассматриваемый элемент, до раскрытия иерархии.

"Плоское" представление, по своей сути, в итоге состоит из элементов, содержащихся в библиотеке базового проекта. В библиотеке выделяются следующие типы элементов: логические элементы L_{LE}, периферийные элементы (ПЭ) ввода-вывода L_{IO}, сложно-функциональные макроблоки L_M, трассировочные элементы L_{Ro} и иные вспомогательные элементы L_{BB} ("черные ящики"), не содержащие в своем составе ни один из перечисленных типов элементов $L_{LE}, L_{IO}, L_M, L_{Ro}$, выполняющие вспомогательные функции, (например, для программирования памяти), не связанные с непосредственным отображением элементов пользовательской проектируемой схемы:

$$L=L_{LE}\cup L_{IO}\cup L_M\cup L_{Ro}\cup L_{BB}. \qquad (15)$$

При раскрытии иерархии и преобразовании базового проекта в "плоское" представление учитывается факт наличия и фактическое количество элементов каждого из перечисленных типов в каждой из подсхем на более высоких уровнях иерархии. При обозначении фактического количества элементов перечисленных типов в некоторой заданной подсхеме s через $\sigma_{LE}(s)$, $\sigma_{IO}(s)$, $\sigma_M(s)$, $\sigma_{Ro}(s)$ принадлежность подсхемы к множеству "черных ящиков" можно определить по факту отсутствия в ней элементов из перечисленных типов, не зависимо от того, имеет ли такая схема детализацию в терминах подсхем и элементов более низкого уровня:

$$s\in L_{BB}\equiv$$
$$\equiv(\sigma_{LE}(s)+\sigma_{IO}(s)+\sigma_M(s)+\sigma_{Ro}(s)=0). \qquad (16)$$

При этом значение каждой из перечисленных функций подсчета элементов соответствующего типа $T\in\{LE, IO, M, Ro\}$ можно определить рекурсивно:

$$\sigma_T:\ S\to\mathcal{N}_0,\ \ \mathcal{N}_0=\mathcal{N}\cup 0;$$
$$\sigma_T(s)=\begin{cases}1 & \text{при}\quad s\in L_T\\ 0 & \text{при}\quad s\in L\backslash L_T\\ \sum\limits_{e\in E(s)}\sigma_T(m(e)) & \text{при}\quad s\notin L.\end{cases} \qquad (17)$$

По результатам таких подсчетов раскрытие иерархии при преобразовании базового проекта в "плоское" ограничивается (остановить) на уровне $L=L_{LE}\cup L_{IO}\cup L_M\cup L_{Ro}\cup L_{BB}$.

Элементы e: $m(e)\in L_{LE}\cup L_{IO}\cup L_M$ используются для отображения библиотечных элементов пользовательского проекта. Элементы e: $m(e)\in L_{Ro}$ применяются для отображения цепей и коммутаций пользовательского проекта, и на их основе в автоматическом режиме строится граф для решения задач трассировки.

Схемы библиотечного уровня $s\in L_{LE}\cup L_{IO}\cup L_M=L\backslash\{L_{BB}\cup L_{Ro}\}$ программируются на основе библиотеки под различные варианты функциональных решений с использованием программируемой памяти. Множество внешних выводов таких схем:

$$P(s)=\{p_i,\ i=1,\ldots,|P(s)|\},\quad s\in L_{LE}\cup L_{IO}\cup L_M \qquad (18)$$

разделяется на 3 независимых подмножества по функциональному назначению:

$$P(s)=P_r(s)\cup P_m(s)\cup P_s(s), \qquad (19)$$

где $P_r(s)$ – подмножество сигнальных или трассировочных выводов для подключения внешних сигнальных цепей с применением трассировочных ресурсов из L_{Ro},

$P_m(s)$ – подмножество программируемых выводов для управления различными вариантами функциональных решений;

$P_s(s)$ – подмножество специальных выводов для подключения специальных сигналов (например, для выбора цепей питания, земли, синхронизации, сброса и др.), подключение которых требует специальной обработки, отличной от подключения обычных цепей или сигналов.

Мощность множества $|P_r(s)|$ определяет максимально допустимое количество выводов элемента в пользовательской библиотеке L_u пользовательского проекта $\Pi_u=(S_u,L_u,s_{mu})$. Некоторые из выводов $P_r(s)$ могут не использоваться в конкретном библиотечном элементе из L_u или быть замкнутыми на землю/питание.

Мощность множества $|P_m(s)|$ определяет длину вектора программирования для реализации конкретных функций и режимов работы библиотечных элементов. За счет разных вариантов программирования одному экземпляру $s \in L_{LE} \cup L_{IO} \cup L_M$ может соответствовать множество различных реализаций в пользовательской библиотеке L_u с общим количеством элементов равным $2^{|P_m(s)|}$. В частности, количество вариантов классического LUT (LookUp Table) [25] с n входами равно:

$$2^{|P_m(s)|} = 2^{2^n}. \quad (20)$$

Таким образом, формирование элементов $s_u \in L_u$, $s_u = (\mu(s_u), \emptyset, \emptyset, P(s_u), \emptyset)$ пользовательской библиотеки L_u проекта $\Pi_u = (S_u, L_u, s_{mu})$ реализуется путем установки следующих соответствий для выводов библиотечных схем базового кристалла $P(s) = \{p_i, i = 1, \ldots, |P(s)|\}$, $s \in L_{LE} \cup L_{IO} \cup L_M$:

$$\begin{aligned} P_m(s) &\to \mathcal{B}^{|P_m(s)|}, \quad \mathcal{B} = \{0,1\}; \\ P_r(s) &\to P(s_u) \cup \{P_0, P_1, P_z\}, \end{aligned} \quad (21)$$

где P_0, P_1, P_z – условные обозначения вводов, предполагающие внешние соединения соответственно с узлом земли, питания или висячим узлом.

Без ограничения общности можно предполагать, что пользовательская проектируемая схема от конечного заказчика $\Pi_u = (S_u, L_u, s_{mu})$ задана в "плоском" варианте, получена как результат автоматического синтеза с RTL-описания или результат раскрытия иерархического пользовательского описания в "плоское" представление, тогда $S_u = L_u \cup \{s_{mu}\}$.

Пусть $s_{mu} = (\mu(s_{mu}), E(s_{mu}), N(s_{mu}), P(s_{mu}), C(s_{mu}))$. Что касается внешних выводов пользовательской схемы $p_u \in P(s_{mu})$, то для них возможны два режима обработки:

– Один вариант – назначение периферийных элементов из L_{IO}. При этом в зависимости от типа вывода $\tau(p_u) \in \{\tau_{inp}, \tau_{out}, \tau_{bi}\}$ программируются различные варианты периферийных элементов – входных, выходных или двунаправленных.

– Второй вариант предполагает, что периферийные элементы уже назначены на этапе формирования пользовательской схемы, и выводы схемы $p_u \in P(s_{mu})$ представляются собой внешние интерфейсы для моделирования.

Сам этап назначения периферийных элементов предполагает не только выбор конкретного типа периферийного элемента $s_{IO} \in L_{IO}$ с программированием:

$$P_m(s_{IO}) \to \mathcal{B}^{|P_m(s_{IO})|}, \quad \mathcal{B} = \{0,1\}, \quad (22)$$

но и назначение конкретного периферийного элемента $e \in E(s_f)$, и следовательно, конкретного места размещения этого элемента в "плоском" представлении базового кристалла, т.е. установления соответствия (отображения):

$$\begin{aligned} P(s_{mu}) \to \{e:\ & e \in E(s_f), \\ & m(e) = s_{IO}, \quad s_{IO} \in L_{IO}\}. \end{aligned} \quad (23)$$

При этом сама процедура назначения конкретной периферийной площадки с размещением может выполнятся как в ручном или интерактивном режиме, так и в автоматическом режиме.

Аналогичная задача решается и для всех внутренних элементов пользовательской схемы, как для простых логических элементов, так и для сложно-функциональных макроблоков:

$$\begin{aligned} E(s_{mu}) \to \{e:\ & e \in E(s_f), \\ & m(e) \in \{L_{LE} \cup L_M \cup L_{IO}\}\}. \end{aligned} \quad (24)$$

Установление отображения, в котором каждому элементу пользовательской схемы устанавливается в соответствие элемент базового проекта, по сути своей есть размещение элементов пользовательской схемы.

b) Особенности представления топологии базового кристалла

Назначение периферийных элементов на конкретные площадки и размещение элементов пользовательской схемы на базовом кристалле выполняется, опираясь на результаты анализа его топологии. В случае, когда топология прототипа уже разработана, САПР анализирует файл в формате GDSII, содержащий всю необходимую информацию – точные координаты элементов e: $m(e) \in L_{LE} \cup L_{IO} \cup L_M$, передающие программе их расположение на плоском представлении базового кристалла $\Pi_f(\Pi) = (S_f, L_f, s_f)$, ориентация элемента и его геометрические размеры – ширина и высота.

Таким образом, положение каждого экземпляра $m(e)$ характеризуется координатами точки привязки его левого нижнего края, ориентацией и габаритными размерами:

$$X_{min}(m(e)), \quad Y_{min}(m(e)), \quad O_r(m(e)), \quad (25)$$

где $O_r(m(e)) \in \{O_0, O_R, O_{XY}, O_{XYR}, O_Y, O_{XR}, O_X, O_{XR}\}$ – ориентация, при этом, индексация ориентаций указывает на отсутствие (0) или наличие поворотов (R – поворот против часовой стрелки на 90°) и зеркальных отображений (X, Y) относительно соответствующей оси.

Для ускорения прототипирования используется упрощенное топологическое описание базово-

го кристалла, использующее относительные координаты его элементов, что позволяет пропустить этап топологического проектирования, но при этом передать программе расположение ЛЭ, ячеек ввода-вывода и макроблоков. Относительные координанты генерируются для всех необходимых элементов с помощью набора операций, разработанного на основе схемотехнического описания схемы, ее графического представления и специализированных лингвистических средств на языке Tcl. Генерация таких координат возможна с приведением ориентации всех типов элементов базового кристалла до нормальной $O_r(m_{ij}) = O_0$.

Если на верхнем уровне прототипа кристалла используются только однотипные логические элементы, то прототип можно рассмотреть в плоском представлении в виде полной матрицы ЛЭ:

$$M_f = \{m_{ij}: m_{ij} \in E_{LE}(s_f),\ m(m_{ij}) \in L_{LE},\ i = 1,\ldots,I_f,\ j = 1,\ldots,J_f\},$$
$$E_{LE}(s_f) \subset E(s_f),\ E_{LE}(s_f) = \{e: e \in E(s_f)\ \&\ m(e) \in L_{LE}\}. \quad (26)$$

Если же верхний уровень прототипа кристалла представлен в виде матрицы групп логических блоков (ГЛБ), где ГЛБ представлена матрицей из логических элементов, то при генерации координат такого прототипа для более детального отображения вводится упрощенное двухблочное представление $\Pi_b(\Pi) = (S_b, L_b, s_b)$. В таком представлении наряду с множеством элементов библиотечного уровня L, выделяется промежуточный уровень “блоков”, не входящих в L: $B = \{b_i\}$, $: B \cap L = \emptyset$. Вследствие того, что ГЛБ сгруппированы из одинаковых блоков, $|B| = |\{b\}| = 1$. Конечное упрощенное представление $\Pi_b(\Pi) = (S_b, L_b, s_b)$ состоит из следующих компонент:

$$L_b = \{s: (s \in L) \wedge \varphi(s, s_m)\};$$
$$S_b = \{s_b \cup L_b \cup B\},\ L_b \cap B = \emptyset;$$
$$s_b = (\mu(s_b), E(s_b), \emptyset, \emptyset, \emptyset);$$
$$\mu(s_b) = \mu(s_m).$$

В отличие от представления прототипа в САПР, при генерации его координат все множество ЛЭ разделяется на группы формально. В соответствии с этим, нет необходимости использовать множество цепей прототипа, множество его внешних выводов и соединений, а используется только множество элементов. Также, в отличие от отображения в САПР “плоского” представления, в данном случае выделяются только логические элементы L_{LE}, периферийные элементы ввода-вывода L_{IO}, сложно-функциональные макроблоки L_M, а также ГЛБ – B:

$$S_b = \{s_b \cup L_{LE} \cup L_{IO} \cup L_M \cup B\}. \quad (27)$$

Исходя из этого, подмножество блочных элементов есть $E_B(s_b) \subset E(s_b)$, $E_B(s_b) = \{e: e \in E(s_b)\ \&\ \&\ m(e) \in B\}$, а прототип можно рассмотреть в виде матрицы ГЛБ:

$$M_b = \{m_{ij}: m_{ij} \in E_B(s_b),\ m(m_{ij}) \in B,\ i = 1,\ldots,I_b,\ j = 1,\ldots,J_b\}. \quad (28)$$

Общее количество блоков в базовом кристалле определяется размером матрицы блоков:

$$|E_B(s_b)| = |M_b| = I_b J_b. \quad (29)$$

Аналогично сам блок в двухуровневом представлении состоит из элементов более низкого уровня иерархии:

$$b = (\mu(b), E(b), \emptyset, \emptyset, \emptyset), \quad (30)$$

где $E(b) = \{e: m(e), m(e) \in L_{LE} \cup L_{Ro} \cup L_{BB}\}$. Тогда подмножество логических элементов блока есть $E_{LE}(b) \subset E(b)$, $E_{LE}(b) = \{e: e \in E(b)\ \&\ \&\ m(e) \in L_{LE}\}$ и может быть представлено в виде матрицы ЛЭ в составе ГЛБ:

$$M_{LE} = \{m_{ij}: m_{ij} \in E_{LE}(b),\ m(m_{ij}) \in L_{LE},\ i = 1,\ldots,I_{LE},\ j = 1,\ldots,J_{LE}\}. \quad (31)$$

Общее количество элементов в блоке определяется размером матрицы M_{LE}:

$$|E_{LE}(b)| = |M_{LE}| = I_{LE} J_{LE}. \quad (32)$$

Предполагается, что все логические элементы в двухуровневом блочном представлении локализованы в блоках:

$$\forall e \in E(s_b) \cup E(b): m(e) \in L_{LE} \rightarrow e \in E(b). \quad (33)$$

Другими словами, отсутствуют логические элементы на верхнем уровне иерархического двухуровневого блочного представления:

$$\nexists e\colon e \in E(s_b)\ \&\ m(e) \in L_{LE}.$$

Тогда общее количество логических элементов в составе базового кристалла определяется размерами матриц M_b, M_{LE}:

$$\begin{gathered} I_f = I_b I_{LE}, \\ J_f = J_b J_{LE}, \\ \left|E_{LE}(s_f)\right| = \left|M_f\right| = I_f J_f = \left|E_B(s_b)\right|\left|E_{LE}(b)\right| = I_b J_b I_{LE} J_{LE}. \end{gathered} \tag{34}$$

Для удобства использования введем следующие обозначения элементов множеств ГЛБ и ЛЭ:

$$\begin{gathered} m_{ijB} = m_{ij} \in E_B(s_b); \\ m_{ijLE} = m_{ij} \in E_{LE}(b). \end{gathered} \tag{35}$$

При генерации координат базового кристалла в двух блочном представлении принимается, что точка привязки – это левый нижний угол кристалла, а отчет ЛЭ идет с нижнего левого угла в верхний правый, т.е. снизу слева находится $m_{\min LE}$, а сверху справа $m_{\max LE}$. Также перед генерацией, помимо уже известных параметров, таких как количество ЛЭ в ГЛБ по горизонтали J_{LE} и вертикали I_{LE}, задаются следующие параметры:

• начальные координаты левой нижнего угла кристалла, соответствующие координатам левого нижнего угла самого нижнего ЛЭ:

$$X_{\min}(s_b),\ \ Y_{\min}(s_b) = X_0(m_{00LE}),\ \ Y_0(m_{00LE}); \tag{36}$$

• расстояние между ЛЭ внутри ГЛБ:

$$\Delta X(m_{ijLE}),\ \ \Delta Y(m_{ijLE}); \tag{37}$$

• габариты ЛЭ – ширина и высота:

$$W(m_{ijLE}),\ \ H(m_{ijLE}); \tag{38}$$

• расстояние между ГЛБ:

$$\Delta X(m_{ijB}),\ \ \Delta Y(m_{ijB}). \tag{39}$$

На основе известных параметров рассчитываются габариты ГЛБ:

$$\begin{aligned} W(m_{ijB}) &= J_{LE} W(m_{ijLE}) + (J_{LE} - 1)\Delta X(m_{ijLE}), \\ H(m_{ijB}) &= I_{LE} H(m_{ijLE}) + (I_{LE} - 1)\Delta Y(m_{ijLE}). \end{aligned} \tag{40}$$

Далее, используя описанные заданные и рассчитанные параметры выполняется расчет координат для каждого экземпляра m_{ijB}, состоящего из m_{ijLE}. После каждого ЛЭ учитывается расстояние, как по оси X, так и по оси Y, до следующего элемента. Через каждые $W(m_{ijB})$ по оси X и $H(m_{ijB})$ по оси Y координатам ЛЭ добавляется $\Delta X(m_{ijB})$ и $\Delta Y(m_{ijB})$ соответственно.

Следует отметить, что приведенные формулы для размеров и координат справедливы не только для матрицы блоков логических элементов $M_b = \{m_{ij}\colon m_{ij} \in E_B(s_b),\ m(m_{ij}) \in B,\ i = 1,\ldots,I_b,\ j = 1,\ldots,J_b\}$, но также для периферийных элементов:

$$E_{IO}(s_b) \subset E(s_b),\ \ E_{IO}(s_b) = \{e\colon e \in E(s_b)\ \&\ m(e) \in L_{IO}\} \tag{41}$$

и для макроблоков:

$$E_M(s_b) \subset E(s_b),\ \ E_M(s_b) = \{e\colon e \in E(s_b)\ \&\ m(e) \in L_M\}. \tag{42}$$

Количество макроблоков и их расположение на кристалле может быть совершенно разным, поэтому структурированного описания генерации их координат приведено не будет. Но периферийные элементы, как правило, располагаются по периметру матрицы ЛЭ или ГЛБ, следовательно, их начальные координаты можно описать относительно ЛЭ. В зависимости от стороны, по которой располагаются периферийные элементы – слева, справа, сверху, снизу, выделим соответствующие координаты:

$$
\begin{gathered}
X\left(m_{0left}\right),\quad Y\left(m_{0left}\right),\quad \text{где} \\
X\left(m_{0right}\right) = X\left(m_{00LE}\right) - \Delta X\left(m_{ijLE}, m_{ijIO}\right) - W\left(m_{ijIO}\right), \\
Y\left(m_{0left}\right) = Y_0\left(m_{00LE}\right) \\
X\left(m_{0right}\right),\quad Y\left(m_{0right}\right),\quad \text{где} \\
X\left(m_{0right}\right) = X\left(m_{00LE}\right) + J_b W\left(m_{ijB}\right) + (J_b - 1)\Delta X\left(m_{ijB}\right) + \Delta X\left(m_{ijLE}, m_{ijIO}\right) \\
Y\left(m_{0right}\right) = Y_0\left(m_{00LE}\right) \\
X\left(m_{0bottom}\right),\quad Y\left(m_{0bottom}\right),\quad \text{где} \\
X\left(m_{0bottom}\right) = X_0\left(m_{00LE}\right); \\
Y\left(m_{0bottom}\right) = Y\left(m_{00LE}\right) - \Delta Y\left(m_{ijLE}, m_{ijIO}\right) - H\left(m_{ijIO}\right), \\
X\left(m_{0top}\right),\quad Y\left(m_{0top}\right),\quad \text{где} \\
X\left(m_{0top}\right) = X\left(m_{00LE}\right) \\
Y\left(m_{0top}\right) = Y\left(m_{00LE}\right) + I_b H\left(m_{ijB}\right) + (I_b - 1)\Delta Y\left(m_{ijB}\right) + \Delta Y\left(m_{ijLE}, m_{ijIO}\right).
\end{gathered}
\tag{43}
$$

Остальные параметры для генерации координат элементов ввода-вывода идентичны параметрам ЛЭ и ГЛБ. Отличие заключается в том, что в роли ЛЭ выступает ПЭ, а в роли ГЛБ – группа периферийных элементов (ГПЭ). Индексация каждого экземпляра ПЭ происходит по одной из осей координат – по оси Y для ПЭ слева и справа, по оси X для ПЭ снизу и сверху.

Одновременно с генерацией координат на этом этапе формируется информация о соответствии элемента e: $m(e) \in L_{IO}$ внешнему выводу базового кристалла $P(s_m)$.

IV. ПРАКТИЧЕСКИЕ РЕЗУЛЬТАТЫ ПРОТОТИПИРОВАНИЯ

На основе формализованного представления базового кристалла и проектируемой схемы, описанного в разделах III.a и III.b, были разработаны программные средства САПР, позволяющие реализовать предложенный метод программного прототипирования. В качестве примера, показывающего эффективность разработанного метода, приведены результаты проектирования базового кристалла с итерационным изменением его архитектуры. Ближайший аналог исходного базового кристалла – ПЛИС MAX II компании Altera, имеющая с ним схожую структуру ЛЭ, ГЛБ и трассировочных ресурсов. Целью изменений архитектуры является уменьшение объема требуемой конфигурационной памяти и наращивание логической емкости базовой схемы без потери достигнутого уровня трассируемости пользовательских схем на ее основе.

В процессе прототипирования проводились изменения структуры ГЛБ и трассировочной архитектуры кристалла, в которую входят следующие типы межсоединений: локальная шина (Local), прямые связи (Direct Link – DL), шины R4C4 и R8C8 (R– row, C – column), длинные связи (LongBus), диагональные связи.

Также наряду с разрядностью перечисленных шин проводились изменения в структуре коммутаторов, аккумулирующих эти соединения. В данной архитектуре присутствует три типа коммутаторов:

- switch block (SB) – блок, объединяющий между собой шины R4C4/R8C8 и позволяющий соединить прямые связи с этими шинами;
- connection block (CB) – блок, собирающий шины R4C4/R8C8, прямые и длинные связи в локальную шину внутри ГЛБ;
- local connection block (LCB) – блок, распределяющий приходящие на локальную шины сигналы по всем необходимым ЛЭ в ГЛБ.

Далее подробнее остановимся на функционале всех присутствующих типов межсоединений. Локальная шина обеспечивает связь между ЛЭ внутри ГЛБ. Она подключена не только к каждому ЛЭ по отдельности, но и к глобальным межсоединениям строк и столбцов. Это обеспечивает прямую связь между ГЛБ и сводит использование глобальных шин к минимуму.

При этом ГЛБ могут быть соединены друг с другом в пределах одной строки двумя способами:

- прямая связь по локальной шине;
- шина R4, охватывающая четыре ГЛБ слева, четыре справа;
- шина R8, охватывающая восемь ГЛБ слева, восемь справа.

Прямая связь дает доступ к локальным шинам соседних ГЛБ, расположенным слева и справа, а также обеспечивает быструю передачу данных между ГЛБ и/или блоками ввода/вывода, не взаимодействуя с шинами R4 и R8. Каждая ГЛБ имеет соединения с шинами R4/R8 как в левую, так и в правую сторону.

Структура столбцовых межсоединений аналогична структуре строчных. Отличие лишь в том, что вместо шин R4/R8 для них используются шины C4/C8, которая позволяет соединить соседние ГЛБ в пределах одного столбца, охватывая четыре/восемь соседних ГЛБ вверх и столько же ГЛБ вниз.

Регулярная структура соединений в виде строк и столбцов фиксированной длины, позволяет предсказать точное время коротких задержек в распространении сигнала.

Имеющиеся в архитектуре длинные связи проходят через весь столбец или строку кристалла, позволяя соединить между собой дальние ЛЭ.

В качестве исходной схемы был взят прототип со следующими характеристиками трассировочной архитектуры: разрядность шин R4/C4 – 32 бита, разрядность шин R8/C8 – 64 бита, разрядность длинных связей – 10 бит, прямых связей – 10 бит, локальной шины– 22 бита. При этом пропускная коммутационная способность блока локальных связей не равна 100%. Его структура разрежена, что снижает пропускную способность до 75%. Данная схема имеет 16 столбцов и 20 строк ГЛБ, при этом каждая ГЛБ состоит из 10 ЛЭ. Общая площадь схемы равна 3200 ЛЭ.

Прототипирование выполнялось с использованием тестовой пользовательской проектной схемы s38417 из набора ISCAS'89 [26]. Результат ее логического синтеза составляет 3184 ЛЭ и 3215 межсоединений.

Результаты прототипирования показаны в табл. 1. Она состоит из столбцов с именами текущего прототипа и прототипа, на основе которого он разработан, описания текущего прототипа и результатов анализа имплементации схемы на его основе. Результаты представлены в виде количества неразведенный цепей и объема конфигурационной памяти, приходящегося на одну ГЛБ и в среднем на один элемент этой группы.

В табл. 1 жирным шрифтом выделены модификации текущего прототипа относительно предыдущего, указанного в соответствующем столбце. Из таблицы видно, что в прототипах 1.1–1.5 уменьшение памяти достигалось за счет редукции коммутатора connection block и уменьшения дальности и разрядности шин R/C. На прототипах 1.2–1.3 выявлено, что используемая в них длина и разрядность связей R/C недопустима, так как в этом случае трассируемость падает практически до 0 на любой из пользовательских проектных схем независимо от ее размера.

В прототипах 1.6–1.15 предпринята попытка уменьшить объем занимаемой памяти, не потеряв трассируемость, за счет сокращения прямых связей. Во-первых, их разрядность уменьшается до 5 бит, а во-вторых, теперь только одна половина ЛЭ имеет прямые связи вверх и вверх по диагоналям, а другая половина – только вниз и вниз по диагоналям. Подробнее доступные связи прототипа 1.6 показаны на рис. 2.

Вследствие того, что при дальнейшей редукции связей DL, трассируемость будет только падать, в прототипе 1.16 снова добавлен полный коммутатор прямых связей, позволяющий каждому ЛЭ в ГЛБ соединиться с соседний. При этом, в целях экономии конфигурационной памяти, разрядность длинных связей сокращена до 8.

Уровень трассируемости, который, в отличие от исходного кристалла, на прототипах 1.1–1.16 упал до ~113 не трассируемых цепей, повысило увеличение длин и разрядностей шин R/C, а также увеличение размера ГЛБ до 16 ЛЭ.

При достижении полной трассируемости, помимо сокращения объема памяти, к целям прототипирования добавляется уменьшение максимальной и средней длины цепей. Продолжение процесса программного прототипирования с учетом новых поставленных целей показано в табл. 2. Здесь за исходный базовый кристалл взят прототип 1.16, с увеличением размера ГЛБ до 16 ЛЭ, количества ГЛБ и пропорциональным добавлением разрядностей коммутационным шинам. Общая площадь такой схемы равна 4096 ЛЭ. В таблице отсутствует столбец с названием предыдущего прототипа, так как модификации вносятся последовательно в предыдущий прототип. Также вследствие того, что во всех прототипах полностью разведены все тестовые схемы, в таблице отсутствует столбец с количеством не трассируемых цепей.

Увеличение прототипа позволяет использовать для тестирования пользовательскую схему большего объема. В данном случае в качестве такой схемы была выбрана ac97 [27], результат логического синтеза которой составил 3732 ЛЭ и 3821 цепи.

На этом этапе прототипирования уменьшение памяти достигается за счет редукции коммутаторов SB и LCB, а также небольшим изменением разрядности локальных связей и разрядности шин R3C3 и R6C6 на пересечении соединений строки и столбца (см. табл. 2).

Результатом прототипирования стал базовый кристалл, разработанный на основе прототипа 2.6. Средний объем памяти на один ЛЭ в этом прототипе меньше исходного на 47.3 бита. При этом не утрачен исходный уровень трассируемости межсоединений, а общий размер кристалла увеличен на 896 ЛЭ.

Таким образом, программное прототипирование позволило оценить архитектуру базового кристалла до разработки его топологии и получить ПЛИС, удовлетворяющую всем заданным требованиям.

Таблица 1. Результаты программного прототипирования базового кристалла размером 16 × 20 ГЛБ

Имя прототипа	Имя пред. прототипа	Описание прототипа	Количество неразведенных цепей, шт.	Объем памяти на ЛЭ/ГЛБ, бит
1.0	–	Исходный базовый кристалл R4C4 = 32, R8C8 = 64, LongBus = 10, DL = 10, Local = 22	0	**291.3**/2913
1.1	1.0	**Редукция коммутатора СВ** R4C4 = 32, **R8C8 = 32**, LongBus = 10, DL = 10, Local = 22	114	**278.7**/2787
1.2	1.1	**Редукция коммутатора СВ** **R4C4 = 16**, R8C8 = 32, LongBus = 10, DL = 10, Local = 22	–	**252.5**/2525
1.3	1.1	**Редукция коммутатора СВ** **R4C4 = 16**, **R6C6 = 24**, LongBus = 10, DL = 10, Local = 22		**249.3**/2493
1.4	1.3	**Редукция коммутатора СВ** **R3C3 = 24**, **R6C6 = 24**, LongBus = 10, DL = 10, Local = 22	111	**268.5**/2685
1.5	1.4	**Редукция коммутатора СВ** R3C3 = 24, R6C6 = 24, LongBus = 10, DL = 10, Local = 22	113	**249.3**/2493
1.6	1.5	R3C3 = 24, R6C6 = 24, LongBus = 10, **DL = 5**, Local = 22 **добавлены связи DL ↗ и ↙ (до SB) = 5** **DL (↖, ↑, ↗, ↖ до SB, ↗ до SB)** **доступны у 5 верхних ЛЭ из ГЛБ.** **DL (↙, ↓, ↘, ↙ до SB, ↘ до SB)** **доступны у 5 нижних ЛЭ из ГЛБ**	114	**244.9**/2449
1.7	1.6	R3C3 = 24, R6C6 = 24, LongBus = 10, DL = 5, Local = 22 **Удалены связи ↗ и ↙ до SB**	112	**233.7**/2337
1.8	1.6	R3C3 = 24, R6C6 = 24, LongBus = 10, **DL (←, →) = 10**, DL = 5, DL ↗ и ↙ (до SB) = 5, Local = 22	113	**246.1**/2461
1.9	1.8	R3C3 = 24, R6C6 = 24, LongBus = 10, DL (←, →) = 10, DL = 5, Local = 22 **удалены связи DL ↗ и ↙ (до SB) = 5**	116	**234.9**/2349
1.10	1.6	R3C3 = 24, R6C6 = 24, LongBus = 10, **DL (↖, ↗, ↙, ↘) = 5, DL = 10**, Local = 22	108	**247.1**/2471
1.11	1.10	R3C3 = 24, R6C6 = 24, LongBus = 10, DL (↖, ↗, ↙, ↘) = 5, DL = 10, Local = 22 **удалены связи DL ↗ и ↙ (до SB) = 5**	114	**235.9**/2359
1.12	1.0	R4C4 = 32, R8C8 = 64, LongBus = 10, **DL = 5**, Local = 22 **добавлены связи DL ↗ и ↙ (до SB) = 5**	112	**286.9**/2869
1.13	1.12	R4C4 = 32, R8C8 = 64, LongBus = 10, **DL = 5**, Local = 22 **удалены связи DL ↗ и ↙ (до SB)**	114	**274.1**/2741

Таблица 1. Окончание

Имя прототипа	Имя пред. прототипа	Описание прототипа	Количество неразведенных цепей, шт.	Объем памяти на ЛЭ/ГЛБ, бит
1.14	1.5	**Редукция коммутатора connection block** R3C3 = 24, R6C6 = 24, LongBus = 10, **DL = 4**, Local = 22 **добавлены связи DL ↗ и ↙ (до SB) = 4** **DL (для направлений ↖, ↑, ↗, ↖ до SB, ↗ до SB) доступны у 4 верхних ЛЭ из ГЛБ.** **DL (↙, ↓, ↘, ↙ до SB, ↘ до SB) доступны у 4 нижних ЛЭ из ГЛБ** **DL (←, →) доступны 4 центральным ЛЭ (3, 4, 5, 6 ЛЭ)**	115	**235.3**/2353
1.15	1.14	R3C3 = 24, R6C6 = 24, LongBus = 10, DL = 4, **DL до SB = 8**, Local = 22 **DL до SB внутри ГЛБ сокращена до 8**	118	**227.3**/2273
1.16	1.15	**Добавлен полный коммутатор на DL во все стороны** R3C3 = 24, R6C6 = 24, **LongBus = 8**, DL = 4, **DL до SB = 8**, Local = 22	112	**275.3**/2753
1.17	1.16	Количество ЛЭ в ГЛБ увеличено до 16. **R4C4 = 32, R8C8 = 64**, LongBus = 8, DL = 4, DL до SB = 8, Local = 22	0	**269**/4314
1.18	1.17	**R3C3 = 24, R6C6 = 48**, LongBus = 8, DL = 4, DL до SB = 8, Local = 22	0	**275**/4114

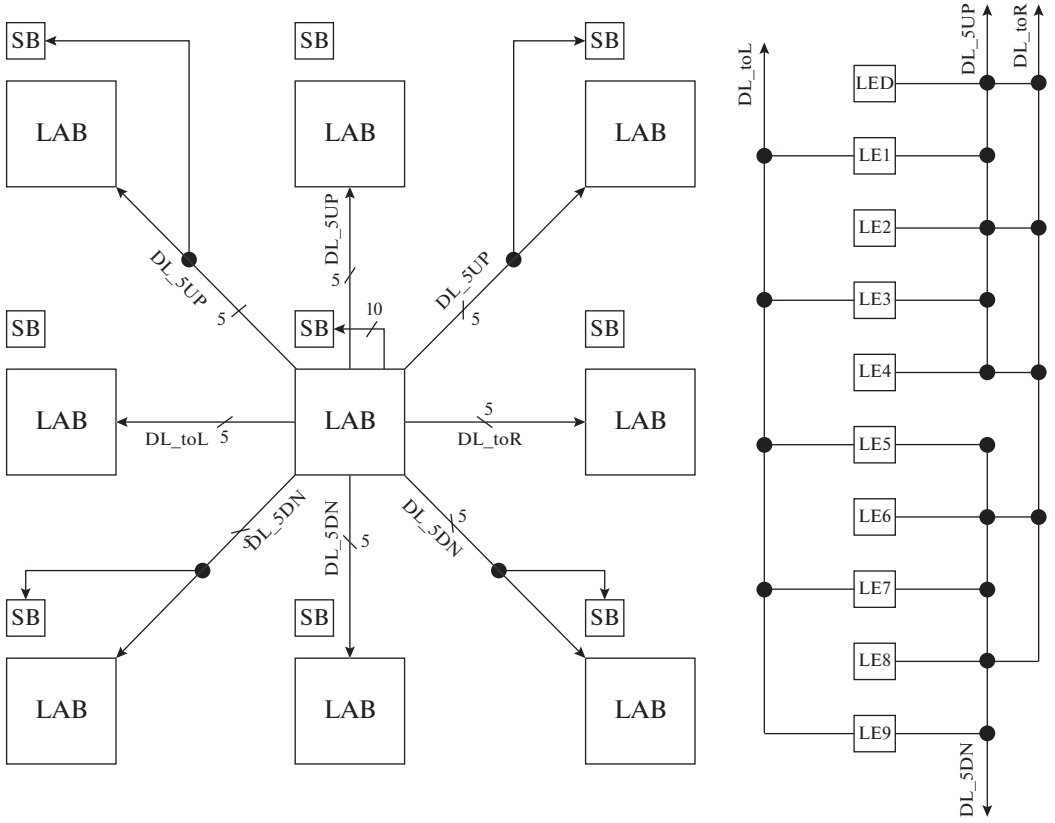

Рис. 2. Схематическое изображение структуры прямых связей прототипа 1.6.

Таблица 2. Результаты программного прототипирования базового кристалла размером 16 × 16 ГЛБ

Имя прототипа	Описание прототипа	Объем памяти на ЛЭ/ГЛБ, бит	5 максимальных длин цепи. Средняя длина цепи	
			S38417	Ac97
2.1	B LAB 16 LE с геометрией кристалла 16 × 16. R3C3 = 24, R6C6 = 48, LongBus = 8, DL = 4, Local = 34	**278**/4454	26/24/24/24/23 **8.56**	27/27/27/27/27 **9.97**
2.2	Добавлены прямые связи с LongBus на IO блоки	**278**/4454	27/27/27/23/22 **8.52**	37/37/37/37/37 **10.25**
2.3	Редукция коммутатора SB	**264**/4230	19/19/19/19/19 **7.54**	25/25/25/25/25 **8.42**
2.4	В коммутаторе SB сокращены повороты шин R3C3 и R6C6 с 8 разрядов до 4	**259**/4150	22/19/19/19/19 **7.55**	28/28/27/27/27 **8.66**
2.5	Сокращение локальных связей внутри ГЛБ с 10 до 8	**252**/4042	22/19/19/19/19 **8.31**	31/28/28/27/27 **9.60**
2.6	Редукция LCB < 75%	**244**/3914	22/22/19/19/19 **8.31**	28/28/28/28/27 **9.68**

V. ЗАКЛЮЧЕНИЕ

В данной статье представлен новый этап проектирования реконфигурируемых и гетерогенных СнК и ПЛИС – программное прототипирование, позволяющий оценить архитектуру базового кристалла до этапа разработки топологического вида схемы. Наряду с этим, описан разработанный метод выполнения программного прототипирования и формализованное представление схемотехнического описания конструкции РСнК, и ПЛИС в САПР, которое обеспечивает гибкую и оперативную настройку ПО на загружаемую схему.

Подробно рассмотрен этап загрузки базового кристалла и представлены особенности анализа и обработки топологии РСнК и ПЛИС в САПР. Продемонстрированы практические результаты применения разработанного метода программного прототипирования архитектуры ПЛИС. Описаны возможные параметры архитектуры и характеристики, на основе которых выполнялось сравнение полученных прототипов.

СПИСОК ЛИТЕРАТУРЫ

1. Разработка и изготовление на отечественном предприятии по технологии с минимальными топологическими нормами не более 0,18 мкм библиотеки аналоговых IP блоков для использования в составе сверхбольших интегральных схем “Система на кристалле”: отчет о НИР / ОАО “НИИМЭ” / исполн.: Красников Г.Я., Эннс В.И. и др. М.: Зеленоград, 2017. 387 с. N ГР 13411.1400099.11.056.
2. *Эннс В.И.* СнК, БМК или ПЛИС: выбор варианта исполнения цифровой интегральной схемы // Компоненты и технологии. 2018. № 4. С. 100–102.
3. *Красников Г.Я.* Возможности микроэлектронных технологий с топологическими размерами менее 5 нм // Наноиндустрия. 2020. Т. 13 № S5-1(102). С. 13–19.
4. *Красников Г.Я., Панасенко П.В., Волосов В.А., Щербаков Н.А.* Тенденции развития технологии сложно функциональной гетероинтегрированной ЭКБ // Международный форум “Микроэлектроника-2018”, 4-я Международная научная конференция “Электронная компонентная база и микро электронные модули”: Сборник тезисов, Алушта. Алушта: Рекламно-издательский центр “ТЕХНОСФЕРА”, 2018. С. 341–344.
5. *Li X., Yang H., Zhong H.* Use of VPR in Design of FPGA Architecture // 2006 8th International Conference on Solid-State and Integrated Circuit Technology Proceedings. Shanghai, China: IEEE. 2006. P. 1880–1882.
6. *Luu J. et al.* VPR 5.0: FPGA CAD and Architecture Exploration Tools with Single-Driver Routing, Heterogeneity and Process Scaling // ACM Transactions on Reconfigurable Technology and Systems (TRETS), Monterey, California, USA: ACM, 2008. P. 133–142.
7. *Parvez H. et al.* A new coarse-grained FPGA architecture exploration environment // 2008 International Conference on Field-Programmable Technology. Taipei, Taiwan: IEEE.2008. P. 285–288.
8. *Kannan P., Balachandran S., Bhatia D.* On metrics for comparing routability estimation methods for FPGAs // Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324). New Orleans, LA, USA: IEEE, 2002. P. 70–75.
9. *Gao Hai-xia et al.* A novel Monte-Carlo method for FPGA architecture research // Proceedings. 7th International Conference on Solid-State and Integrated Circuits Technology, 2004. Beijing, China: IEEE. 2004. V. 3. P. 1944–1947.

10. *Doman D.* Engineering the CMOS Library: Enhancing Digital Design Kits for Competitive Silicon // John Wiley & Sons Ltd. 2012. 256 p.
11. *Amos D., Lesea A., Richter R.* FPGA-based Prototyping Methodology Manual: Best Practices in Design-for-prototyping // Synopsys Press. 2011. 470 p
12. *Ohba N., Takano K.* An SoC design methodology using FPGAs and embedded microprocessors // In Proceedings of the 41st annual Design Automation Conference (DAC '04). Association for Computing Machinery, N.Y., NY, USA. P. 747–752.
13. *Чочаев Р.Ж., Железников Д.А., Иванова Г.А., Гаврилов С.В., Эннс В.И.* Модели и методы анализа структуры коммутационных ресурсов ПЛИС // Известия высших учебных заведений. Электроника. 2020. Т. 25. № 5. С. 410–422.
14. *Gandhare S., Karthikeyan B.* Survey on FPGA Architecture and Recent Applications // 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 2019. P. 1–4.
15. MAX II Device Handbook // Altera Corp. [Электронный ресурс]. Системные требования: AdobeAcrobat Reader. Режим доступа: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max2/max2_mii5v1.pdf (дата обращения: 01.04.2021).
16. UltraScale Architecture Configurable Logic Block User Guide // Xilinx, Inc. [Электронный ресурс]. Системные требования: Adobe Acrobat Reader. Режим доступа: https://www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf (дата обращения: 01.04.2021).
17. ProASIC3 nano FPGA Fabric User's Guide // Microsemi Corp. [Электронный ресурс]. Системные требования: Adobe Acrobat Reader. Режим доступа: http://www.ibselectronics.com/ibsstore/datasheet/Microsemi/PA3_nano_UG.pdf. (дата обращения: 01.04.2021).
18. GDSII™ Stream Format Manual, Release 6.0 // Calma Company. [Электронный ресурс]. Системные требования: Adobe Acrobat Reader. Режим доступа: http://bitsavers.informatik.uni-stuttgart.de/pdf/calma/GDS_II_Stream_Format_Manual_6.0_Feb87.pdf. (дата обращения: 01.04.2021).
19. *Гаврилов С.В., Железников Д.А., Заплетина М.А., Хватов В.М., Чочаев Р.Ж., Эннс В.И.* Маршрут топологического синтеза для реконфигурируемых систем на кристалле специального назначения // Микроэлектроника. 2019. Т. 48. № 3. С. 211–223.
20. *Васильев Н.О., Тиунов И.В., Рыжова Д.И.* Метод логического ресинтеза схем в маршруте проектирования на ПЛИС // МЭС 2020 Проблемы разработки перспективных микро- и наноэлектронных систем. 2020 (МЭС-2020). Выпуск 4. С. 39–44.
21. *Иванова Г.А., Рыжова Д.И., Гаврилов С.В., Васильев Н.О., Стемпковский А.Л.* Методы и алгоритмы для логико-топологического проектирования микроэлектронных схем на вентильном и межвентильном уровне для перспективных технологий с вертикальным затвором транзистора // Микроэлектроника. 2019. Т. 48. № 3. С. 201–210.
22. *Гаврилов С.В., Железников Д.А., Чочаев Р.Ж., Хватов В.М.* Алгоритм декомпозиции на основе метода имитации отжига для реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем. 2018. Выпуск I. С. 199–204.
23. *Гаврилов С.В., Железников Д.А., Чочаев Р.Ж., Эннс В.И.* Адаптация метода моделирования отжига для размещения элементов в базисе реконфигурируемых систем на кристалле // Электронная техника. Серия 3. Микроэлектроника. 2018. Вып. 4(172). С. 55–61.
24. *Заплетина М.А., Железников Д.А., Гаврилов С.В.* Иерархический подход к трассировке реконфигурируемой системы на кристалле островного типа // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2020. № 3. С. 16–21.
25. *Robert J.* Francis. 1992. A tutorial on logic synthesis for lookup-table based FPGAs // In Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design (ICCAD'92). IEEE Computer Society Press, Washington, DC, USA. P. 40–47.
26. *Brglez F., Bryan D., Kozminski K.* Combinational profiles of sequential benchmark circuits // IEEE International Symposium on Circuits and Systems, Portland, OR, USA. 1989. V. 3. P. 1929–1934.
27. *Usselmann R.* AC 97 Controller IP Core // Rudolf Usselmann. [Электронный ресурс]. Режим доступа: https://opencores.org/projects/ac97 (дата обращения: 03.04.2021).