УДК 658.512.011:681.326:519.713

Методы высокоуровневого и логического моделирования в САПР СБИС

В.И. Хаханов, Murad Ali Abbas, Е.И. Литвинова, Baghdad Ammar Avni Abbas, И.В. Хаханова,
hahanov@kture.kharkov.ua

Харьковский национальный университет радиоэлектроники

***Аннотация* — Предлагаются кубитные (квантовые) структуры данных и вычислительных процессов для существенного повышения быстродействия при решении задач поиска, распознавания, принятия решений, дискретной оптимизации и отказоустойчивого проектирования. Представлен суперпозиционный метод синтеза куба функциональности для ее имплементации в кристаллы программируемой логики. Описаны аппаратно ориентированные модели параллельного (за один цикл) вычисления булеана (множества всех подмножеств) на универсуме из n примитивов для решения задач покрытия, минимизации булевых функций, сжатия данных, синтеза и анализа цифровых систем за счет реализации процессорной структуры в форме диаграмме Хассе.**

***Ключевые слова* — мультипроцессор, векторно-логический анализ, критерий качества, верификация, диагностирование цифровых компонентов, процесс-модель поиска дефектов.**

I. Введение

Квантовые вычисления в последние годы становятся интересными для анализа кибернетического пространства, создания новых Интернет технологий, благодаря их некоторой альтернативности существующим моделям вычислительных процессов. Кроме того, рыночная привлекательность квантовых или кубитных моделей основывается на высоком параллелизме решения практически всех задач дискретной оптимизации, факторизации, минимизации булевых функций, эффективного сжатия, компактного представления и телепортации данных, отказоустойчивого проектирования [1-10] за счет существенного повышения аппаратных затрат. Но такая плата в настоящее время вполне допустима, поскольку существуют проблемы заполнения площадей силиконового кристалла, который содержит до 1 миллиарда вентилей при толщине пластины, равной 5 микрон. При этом современные технологии допускают создание пакета или «сэндвича», содержащего до 7 кристаллов, что соизмеримо с объемом нейронов головного мозга человека. Практически «беспроводное» соединение таких пластин основывается на технологической возможности сверления порядка 10 тысяч сквозных отверстий (vias) на 1 квадратном сантиметре. Наполнить полезной функциональностью такой объем допустимых на кристалле вентилей в настоящее время проблематично. Поэтому можно и нужно использовать «жадные» к аппаратуре модели и методы для создания быстродействующих средств параллельного решения практических задач. Имея в виду дискретность и многозначность алфавитов описания информационных процессов свойство параллелизма, заложенное в квантовых вычислениях, является особенно востребованным при создании эффективных и интеллектуальных «движков» для киберпространства или Интернет [4]; средств синтеза отказоустойчивых цифровых примитивов и систем [5]; тестирования и моделирования цифровых систем на кристаллах [6-8]; технологий защиты информации и компьютерных систем [4]. Здесь не рассматривается физическая основа квантовых вычислений, изначально заложенная в трудах ученых, ориентированных на возможность использования недетерминированных квантовых взаимодействий на уровне атома [9-10].

II. Кубитные квантовые модели данных и вычислительных процессов

Квантовый компьютер предназначен для отказоустойчивого проектирования и решения оптимизационных задач, связанных с полным перебором на основе использования теории множеств. Особенность классического компьютера в том, что множество элементов в компьютере упорядочено, поскольку каждый байт имеет свой адрес. Поэтому теоретико-множественные операции сводятся к перебору всех адресов примитивных элементов. Адресный порядок структур данных хорош для задач, где компоненты моделей можно строго ранжировать, что дает возможность выполнять их анализ за один проход или одну итерацию. Там, где нет порядка в структуре, например, в множестве всех подмножеств, классическая модель памяти и вычислительных процессов наносит вред времени анализа ассоциации равных по рангу примитивов, или, в лучшем случае, обработка ассоциативных групп является неэффективной. Что можно предложить для неупорядоченных данных вместо строгого порядка?

Процессор, где элементарной ячейкой служит образ или шаблон универсума из n примитивов, который генерирует $Q = 2^n$ всех возможных состояний такой ячейки в виде булеана или множества всех подмножеств. Прямое решение создания такой ячейки основано на унитарном позиционном кодировании состояний примитивов, которое с помощью суперпозиции последних образует множество всех подмножеств, формирующее в пределе универсум примитивов. Например, четыре примитива создают булеан, содержащий шестнадцать состояний (сочетаний), с помощью четырех двоичных разрядов:

A={Q=(1000), E=(0100), H=(0010), J=(0001), O={Q,H}=(1010), I={E,J}=(0101), A={Q,E}=(1100), B={H,J}=(0011), S={Q,J}=(1001, P={E,H}=(0110), C={E,H,J}=(1110), F={Q,H,J}=(1011), L={Q,E,J}=(1101), V={Q,E,H}=(1110), Y={Q,E,H,J} =(1111), U=(0000)}.

Операции над символами теоретико-множественного алфавита сводятся к логическим командам and, or, not, xor, которые формируют функционально полный базис согласно теореме Поста. Например, ниже представлены логические преобразования теоретико-множественных операций:

$$Q \cup E = 1000 \vee 0100 = 1100 = A;$$
$$S \cap V = 1001 \wedge 1110 = 1000 = Q;$$
$$F \Delta Y = 1011 \oplus 1111 = 0100 = Y;$$

Многозначность символов алфавита положительно влияет на минимизацию булевых функций. Например, компактное представление состояния входных переменных отдельных булевых функций от двух переменных при их кодировании символами 16-значного алфавита имеет не более двух кубов многозначного покрытия:

$$\begin{array}{|cc|}\hline 00&0\\01&0\\10&0\\11&0\\\hline\end{array} = \begin{array}{|cc|}\hline Q&0\\E&0\\H&0\\J&0\\\hline\end{array} = \boxed{Y\ 0} = \boxed{1111\ 0};$$

$$\begin{array}{|cc|}\hline 00&0\\01&0\\10&1\\11&1\\\hline\end{array} = \begin{array}{|cc|}\hline Q&0\\E&0\\H&1\\J&1\\\hline\end{array} = \begin{array}{|cc|}\hline A&0\\B&1\\\hline\end{array} = \begin{array}{|cc|}\hline 1100&0\\0011&1\\\hline\end{array};$$

Таким образом, переход от двоичных векторов входных сигналов к символам замкнутого многозначного алфавита дает принципиально новую возможность к минимизации кубических покрытий (таблиц истинности), которые всегда будут иметь не более двух кубов $f(X) = \{C_1, C_0\}$, $C_1 \cup C_0 = U$, $C_1 = \overline{C}_0$, формирующих единичное и нулевое значение выхода функции двумя взаимно дополняющими символами, в совокупности формирующими символ-универсум. При этом мощность универсума примитивов $\mathrm{card}U = n$ формирует общее число состояний $Q = 2^n$ или производных от них символов, которые кодируются 2^n двоичными разрядами. Последующее двоичное кодирование входного символа каждого из двух кубов дает возможность максимально приблизиться к реализации функционального примитива как элемента памяти программируемых логических устройств (PLD), где входное слово логического элемента есть адрес ячейки памяти (бита), в котором записано состояние выхода. Однако таблица истинности в форме памяти есть дизъюнктивная нормальная форма (ДНФ), которая необратима для решения задачи обратной импликации. Здесь выходом может служить явное задание функциональности в форме кубического покрытия, а точнее двух кубов покрытия, задающих все возможные решения по входам. При этом все логические элементы становятся одновходовыми, где входом является регистровая переменная или n-разрядный вектор, формирующий адрес памяти, хранящей $Q = 2^n$ бит как значений функции $Y = f(A) = f(x_1, x_2, \ldots, x_i, \ldots, x_n)$. Кубит есть двоичный вектор, содержащий n битов, для задания булеана (множества всех подмножеств) состояний $Q = 2^n$ на основе использования n примитивных символов (элементов). Кубит представляет собой совокупность равнозначных двоичных n битов, формирующих единичным значением n примитивов, для обозначения $Q = 2^n$ состояний, составляющих булеан – множество всех подмножеств от n примитивов. Здесь нет чисел! Все биты кубита равны при создании примитивов и отличаются только адресом. Любая теоретико-множественная операция выполняется за один такт, что невозможно при задании ассоциации примитивов на счетном (упорядоченном) пространстве памяти компьютера. Метрика (векторная и скалярная) анализа расстояний, предложенная в [4, 6], абсолютно пригодна для измерения взаимодействия многозначных (двоичных) кубитных объектов, процессов и явлений, путем использования xor-операции. В идеале использование кубитной структуры дает возможность представить любую функциональность в виде двух кубов, привязанных к нулю и единице. Такие кубы формируют конъюнктивную нормальную форму (КНФ) и ДНФ соответственно. Можно упрощать и далее путем исключения из рассмотрения нуля и единицы, неявно имея их в виду. При этом два куба, формирующие входные условия, будут всегда взаимно инверсными, поскольку они дополняют друг друга до универсума примитивов. Следовательно, необходимо оставлять лишь одну букву (символ), а значит один двоичный код, который есть таблица истинности (двухвходового) функционального примитива:

$$\begin{array}{|cc|}\hline 00&0\\01&1\\10&1\\11&0\\\hline\end{array} = \begin{array}{|cc|}\hline Q&0\\E&1\\H&1\\J&0\\\hline\end{array} = \begin{array}{|cc|}\hline S&0\\P&1\\\hline\end{array} = \begin{array}{|cc|}\hline 1001&0\\0110&1\\\hline\end{array} \rightarrow P = \boxed{0110}$$

$$Y = P = E \vee H = A_1 \vee A_2 = \overline{x}_1 x_2 \vee x_1 \overline{x}_2.$$

Полученный вектор интерпретируется здесь не только как совокупность адресуемых битов, но и как

куб, формирующий единичное значение выхода примитива, над которым можно выполнять параллельные векторные логические операции. Иначе, множество входных векторов можно трансформировать к одному вектору, где каждая позиция (адрес) соответствует входной комбинации, а значение позиции – состоянию выхода функции примитива. Это дает возможность уменьшить экспоненциальную (2^n) вычислительную сложность синтеза кубического покрытия функциональности на основе заданных таблиц истинности структурных компонентов цифрового устройства до линейной функции за счет увеличения числа битов при задании переменных с n до 2^n разрядов. Таким образом, структура куба функциональности – вектора выходных значений функционального примитива – изоморфна ДНФ, которая оперирует единичными термами булевых переменных. Аналогично можно записать КНФ, используя нулевой куб покрытия функционального примитива. Результат минимизации интересен и для сжатия информации путем получения минимальной ДНФ, с помощью которой всегда можно восстановить исходную таблицу истинности. Вычислительная сложность минимизации в кубитном исчислении на многозначном алфавите линейна относительно числа переменных. Возникает естественный вопрос. Зачем минимизировать функцию, если она компактно представляется состояниями выходов по всем адресам, составленным из кодов от любого числа переменных? Единственная практически ориентированная цель – выполнение процедур обратной импликации для реализации регулярных алгоритмов синтеза тестов, которые на порядок улучшат свое быстродействие за счет минимального числа кубов в покрытиях функциональных элементов.

Таким образом, на основании введенных кубитных структур данных и Хассе-модели [4] вычислительного процесса можно сделать некоторые выводы: квантовый компьютер создавали специалисты в области квантовой механики, которые привнесли идею создания нечислового компьютера на аналоговой основе представления информации. Введенное понятие кубита соответствует булеану примитивов, что является идеальной нечисленной формой описания компонентов объекта для их анализа, синтеза и оптимизации дискретных объектов.

Формы представления кубита: 1. Символы универсума примитивов, генерирующие множество всех подмножеств (булеан). 2. Двоичные векторы, где булеан представляется сочетанием единичных значений. 3. Диаграмма Хассе, формирующая булеан всех возможных решений на графе. 4. Полный граф переходов, определяющий множество всех подмножеств переходов в виде дуг графа. 5. Геометрическое представление на плоскости кубита в виде точек и отрезков, соответствующих булеану.

На практике более 90% всех задач IT-индустрии, связанных с поиском информации в киберпространстве, ее распознаванием и принятием решений относится к области дискретной математики, где трудно найти место числовой арифметике. Необходимо создавать ассоциативные логические мозгоподобные параллельные (квантовые) процессоры, эффективно оперирующие примитивами булеана (кубита) или элементами и множествами для решения задач дискретной математики. Теоретико-множественные операции необходимо заменять изоморфными логическими командами (and, or, not, xor) для последующего создания новой системы параллельного кубитного программирования для решения логических и оптимизационных задач. Другое решение организации вычислительных процессов связано с топологическим представлением кубита, где элементами выступают геометрические фигуры. Нечисленными задачами, ориентированными на предлагаемый процессор, являются: минимизация форм (булевых функций) описания сложных систем; поиск путей в графе; тестирование и диагностика цифровых систем; комбинаторные исследования процессов и явлений; интеллектуальный поиск данных, распознавание образов и принятие решений; дискретизация фаззи моделей и методов в задачах создания интеллекта.

III. Синтез куба функциональности методом суперпозиции

Формально существуют два пути синтеза функциональности цифровой схемы, имеющей n входов. Первый формирует вектор длиной n путем заполнения нулевыми и единичными значениями функции на основе прямого моделирования всех $q = m \times 2^n$ входных воздействий на m примитивах. Второй – метод суперпозиции кубов примитивных элементов – также осуществляет доопределение всех координат вектора состояний выходов, но путем суперпозиции m кубических покрытий, входящих в состав схемной структуры. В этом случае вычислительная сложность получения функционального покрытия равна $q = 2 \times m$, при условии, что структурные компоненты схемы предварительно были ранжированы в соответствии с порядком распространения сигналов, а покрытие каждого примитива имеет 2 куба. Пример получения покрытия вторым путем для функциональности $f(X) = (X_1X_2) \vee (X_3X_4)$, имеющей три двухвходовых примитива (and, or, or), представлен ниже. Для этого используются два покрытия: C(and)=0001, C(or)=0111. Их взаимодействие как декартово произведение относительно бинарной операции or формирует следующий результат:

$$f(X) = (X_1X_2) \vee (X_3X_4) = (0001) \vee (0111) =$$

$$\begin{array}{c|c} \vee & 0\ 1\ 1\ 1 \\ \hline 0 & 0\ 1\ 1\ 1 \\ = 0 & 0\ 1\ 1\ 1 \\ 0 & 0\ 1\ 1\ 1 \\ 1 & 1\ 1\ 1\ 1 \end{array} = (0111\ 0111\ 0111\ 1111).$$

Интерес представляет формирование покрытия для функции, где существуют избыточные или несущественные переменные:

$$f(X) = (X_1X_2) \vee X_2 = \begin{bmatrix} X_1 & X_2 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \vee_{\times} \begin{matrix} X_2 \\ 0 \\ 1 \end{matrix} =$$

$$\begin{bmatrix} X_1 \wedge X_2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} X_2 \\ 0 \\ 1 \\ 0 \\ 1 \end{matrix} \vee = \begin{matrix} (X_1 \wedge X_2)X_2 \\ 0 \\ 1 \\ 0 \\ 1 \end{matrix} = \begin{matrix} f(X_1,X_2) \\ 0 \\ 1 \\ 0 \\ 1 \end{matrix} =$$

$$= \overline{X}_1X_2 \vee X_1X_2 = \begin{matrix} f(X_1,X_2) = X_2 \\ 0 \\ 1 \end{matrix}.$$

Здесь процедура получения куба выходных значений дополнена шагом минимизации, которая может существенно упростить логическую структуру за счет исключения несущественных переменных. Процесс-модель получения покрытия логической функциональности путем ее последовательного разложения по n переменным включает следующие пункты: 1. Выполнение декартова произведения $\{ \vee_{\times}, \wedge_{\times}, \oplus_{\times} \}$ логической функции от двух (n) переменных в целях формирования вектора выходных значений размерностью $p = 2^n$. Здесь каждый бит вектора одной переменной X_i взаимодействует по логической операции с каждым битом вектора другой переменной X_j: $\begin{bmatrix} X_1 & X_2 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} = \begin{matrix} X_1 \wedge X_2 \\ 0 \\ 0 \\ 0 \\ 1 \end{matrix}$. Размерность полученного вектора равна $p = p_i \times p_j$, где p_i, p_j – разрядности двоичных векторов при соответствующих переменных.

2. Последовательное выполнение декартовых (векторных) логических операций над всеми примитивами (логическими переменными): $P = \prod_{i=1}^{n} P_i$ для получения куба логической функциональности максимальной размерностью $p = 2^n$.

3. Минимизация длины куба функциональности путем исключения несущественных переменных, где устранение m переменных уменьшает в 2^m раз исходную размерность куба (вектора) функционирования схемы.

Для уменьшения вычислительных операций целесообразно иметь библиотеку функций выходов всех логических операций, встречающихся в функциональности. На самом деле таких типов для каждой схемы бывает не более 10. Например, для двухвыходовой схемы, представленной на рис. 1, существует только один тип – and-not.

Реализация процесс модели синтеза общей функциональности будет представлена следующими пунктами:

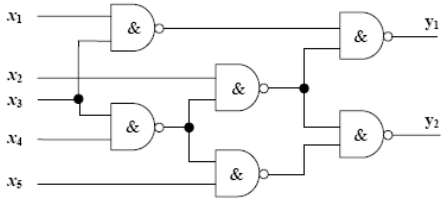

Рис. 1. Тест-структура из библиотеки ISCAS

1. Создание библиотеки примитивов в форме кубов их функциональностей: $C_i = (0001), i = \overline{1,6}$. Для данной схемы имеет место один тип двухвходового примитива: $C_{and-not} = \begin{bmatrix} \overline{X_1X_2} & Y \\ 00 & 1 \\ 01 & 1 \\ 10 & 1 \\ 11 & 0 \end{bmatrix} \begin{matrix} C \\ 1 \\ = 1 = (1110). \\ 1 \\ 0 \end{matrix}$

2. Процедура синтеза куба функциональности для схемной структуры путем суперпозиции кубов примитивных элементов имеет следующий вид:

$$C_5(Y_1) = C_1(\overline{X_2C_2}) = (\overline{\overline{X_1X_3}})[X_2(\overline{X_3X_4})];$$

$$C_6(Y_2) = \overline{X_2C_2}(\overline{C_2X_5}) = \overline{\overline{X_2\overline{X_3X_4}}\,\overline{\overline{X_3X_4}X_5}}$$

$$C_5 = \begin{matrix} \overline{X_1X_3} \\ 1 \\ 1 \\ 1 \\ 0 \end{matrix} \overline{\wedge}_{\times} \begin{bmatrix} & \overline{X_3X_4} \\ X_2 & 1 \\ 0 \;\overline{\wedge} & 1 \\ 1 & 1 \\ & 0 \end{bmatrix} =$$

$$= (0001\ 0010\ 0101\ 1111);$$

$$C_6 = \begin{bmatrix} & \overline{X_3X_4} \\ X_2 & 1 \\ 0 \;\overline{\wedge}_{\times} & 1 \\ 1 & 1 \\ & 0 \end{bmatrix} \overline{\wedge}_{\times} \begin{bmatrix} \overline{X_3X_4} & \\ 1 & X_5 \\ 1 \;\overline{\wedge}_{\times} & 0 \\ 1 & 1 \\ 0 & \end{bmatrix} =$$

$$= (0101\ 0100\ 0111\ 1110).$$

3. Минимизация полученного куба функциональности в целях получения более компактного решения путем суперпозиции таблиц или исключения несущественных переменных.

В рассмотренном примере синтез куба функциональности выполняется в соответствии с порядком возрастания адресов входных переменных при условии непротиворечивости значений одинаковых переменных в каждой строке получаемой таблицы истинности. Для получения компактной таблицы истинности функциональности, представленной схемной структурой, необходимо

выполнить суперпозицию двух таблиц, соответствующих выходам:

$$Y_1 = C_5 = (0001\ 0010\ 0101\ 1111);$$
$$Y_2 = C_6 = (0101\ 0100\ 0111\ 1110)$$

Можно получить решение в виде таблицы истинности, которая в данном случае является не полностью упорядоченной по адресам переменных для второй функции, но она представляет собой компактную таблицу из 16 строк.

Что дает практике получение покрытия в форме одного куба, который является вектором состояний выходов функциональности? 1. Прежде всего – это простота задания сколь угодно сложной схемы от n переменных одним двоичным вектором, размерностью 2^n. 2. Технологичность имплементации данной формы функциональности в кристаллы программируемой логики, где логические функции представлены таблицами истинности, реализованными в виде адресной памяти. 3. Технологичность синтеза функциональности на основе использования LUT (Look Up Table) путем разбиения вектора-куба на кратные степени двойки сегменты. Следует отметить, что логическую функцию произвольной структуры достаточно сложно реализовать на регулярных элементах памяти без внесения избыточности. Поэтому предлагаемый подход получения единого куба из логической схемы с последующим разбиением вектора на сегменты, соответствующие существующим в программируемой логике табличным LUT (4 входа), является эффективным технологическим решением 4. Компактность записи функциональности в виде одного вектора для решения задач анализа, моделирования неисправностей и исправного поведения, синтеза тестов и поиска дефектов. 5. Технологичность и эффективность адресной реализации булевых структур в программные модели, где адресное пространство не нужно разбивать на сегменты в соответствии с ограничениями на аппаратные примитивы. 6. Преобразование цифровой структуры в куб функциональности с последующим разбиением его на регулярные сегменты, соответствующие табличным LUT-примитивам, существенно облегчает решение задач тестирования прототипа, имплементированного в кристалл PLD. 7. Вектор (куб) выходов $C = (C_1, C_2, ..., C_i, ..., C_p), p = 2^n$, есть неявное и компактное (не аналитическое и не табличное) задание булевой функции, ориентированное на адресное выполнение логических операций:

$$Y = f(X_1, X_2, ..., X_i, ..., X_n) = A(X_1, X_2, ..., X_i, ..., X_n) =$$
$$= C(X_1, X_2, ..., X_i, ..., X_n) = C(A_1, A_2, ..., A_i, ..., A_n).$$

Если длина вектора C известна и равна p, то число переменных формирующих адрес для выемки необходимого бита из C-вектора определяется выражением: $n = \log_2 p$.

IV. Имплементация функциональности в технологическую структуру PLD

Решение задачи сводится к разбиению куба функциональности на сегменты, которым можно поставить в соответствие библиотечные примитивы конкретного кристалла. На кристалле PLD (FPGA) имеется в наличии определенное количество четырехвходовых примитивов – элементов памяти для хранения таблицы истинности LUT. Примитивы могут объединяться в логические секторы (Slice), состоящие из двух примитивов, объединенных мультиплексором, что дает возможность увеличить разрядность входных переменных до пяти. Последующее объединение двух секторов с помощью мультиплексора увеличивает разрядность входных переменных до шести. При этом на кристалле технологически предусмотрено, что каждые 2 пары секторов мультиплексируются в блоки CLB (Configurable Logic Block), что дает возможность увеличить число входов синтезируемой логической функции в пределах одного CLB до семи переменных. Структура упомянутых выше модулей (LUT, Slice, CLB) представлена на рис. 2.

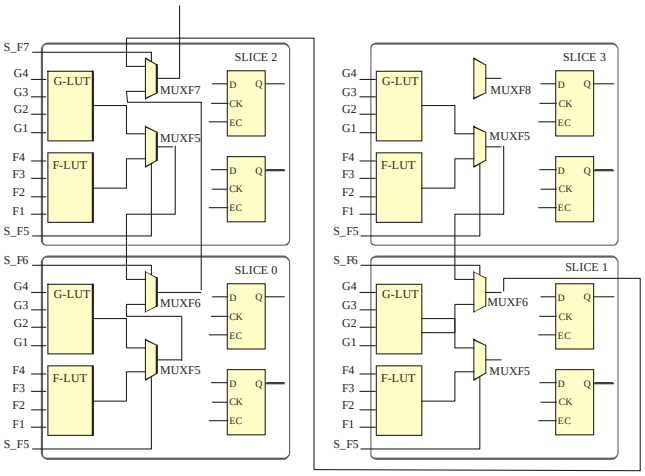

Рис. 2. Структура моделей PLD

Здесь значение функции от семи переменных формируется на выходе мультиплексора MUXF7. Последующее увеличение разрядности входных переменных связано с принудительной структурной организацией компонентов CLB в иерархические структуры более высокого уровня: 1 CLB способен формировать любую функцию от 7 переменных, 2 CLB – 8 переменных, 4 CLB – 9 переменных, 8 CLB – 10 переменных. В общем случае функциональная зависимость числа переменных n от количества CLB-блоков N имеет следующий вид: $n = \log_2 N + 7$.

Для оценки аппаратной сложности при синтезе или реализации вычислительной структуры необходимо иметь обратную зависимость числа логических блоков CLB от сложности логической функции, приведенной к количеству переменных n или мощности куба функциональности $N = 2^n$. Например, для функции от n=10 переменных необходимо иметь на кристалле 8

CLB. В общем случае число логических блоков CLB, в зависимости от числа переменных синтезируемой функции, равно $N = 2^{n-7}$, $n = 7,8,9, \ldots$.

Что касается количества примитивов LUT, то здесь оценка аппаратной сложности будет представлена следующим выражением:

$$N^* = 8L \times 2^{n-7} + 4M \times 2^{n-7} = (8L + 4M) \times 2^{n-7}.$$

Здесь 8L, 4M – есть аппаратная сложность реализации LUT и мультиплексора, входящих в состав CLB. При этом важные характеристики реализации функциональности – структурная глубина, как число уровней иерархии, выраженное в количестве LUT и/или мультиплексорах на самом длинном логическом пути и его задержка D – также зависят от числа переменных (задержки мультиплексора):

$$S = (n-3); \; D = (n-3) \times D_M.$$

V. Выводы

Реализация квантового процессора на основе структуры Хассе позволила уменьшить аппаратурные затраты по сравнению с чисто параллельной реализацией при одновременном уменьшении быстродействия процессора также в n раз. Вывод – необходимо создавать новые структуры данных для снижения аппаратной стоимости квантовых вычислений, и/или более интеллектуальные алгоритмы решения задачи покрытия на диаграммах Хассе. Синтез модели функциональности, представленной в форме одного куба, можно использовать при создании PLD-ориентированных прототипов вычислительных устройств, когда знание куба функциональности приводит к тривиальной технологии имплементации ее в кристалл.

Научная новизна – впервые предложена модель данных и структура аппаратной реализации квантового компьютера, которая характеризуется использованием структуры Хассе, что дает возможность существенно (x100) повысить быстродействие решения практических задач дискретной оптимизации. Предложен суперпозиционный метод синтеза куба функциональности, который характеризуется использованием структуры примитивных элементов и для определенного класса устройств позволяет существенно уменьшить время получения модели цифрового устройства, ориентированной на имплементацию в кристаллы PLD.

Практическая значимость – существенное повышение быстродействия при решении задач покрытия и других задач дискретной оптимизации за счет увеличения аппаратных затрат для параллельного выполнения векторных логических операций на Хассе-структуре квантового вычислительного устройства. Метод построения куба функциональности позволяет существенно упростить решение задачи синтеза цифровой структуры в кристалле на основе использования компонентов (LUT, Slice, CLB), что представляет определенный интерес для его промышленного использования.

Литература

[1] Beth T. Quantum computing: an introduction // Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS. Geneva, 2000. Vol. 1. P. 735–736.

[2] Jonker P., Jie Han. On quantum and classical computing with arrays of superconducting persistent current qubits // Proceedings of Fifth IEEE International Workshop on Computer Architectures for Machine Perception. 2000. P. 69–78.

[3] Keyes R.W. Challenges for quantum computing with solid-state devices // Computer.– Jan. 2005.– Vol. 38, Issue 1.– P. 65 – 69.

[4] Инфраструктура мозгоподобных вычислительных процессов / М.Ф. Бондаренко, О.А. Гузь, В.И. Хаханов, Ю.П. Шабанов-Кушнаренко. Харьков: Новое Слово. 2010. 160 с.

[5] Mark Gregory Whitney. Practical Fault Tolerance for Quantum Circuits // PhD Dissertation in Computer Science. Berkeley: University of California. 2009. 206 p.

[6] Хаханов В.И., Литвинова Е.И., Чумаченко С.В., Гузь О.А. Логический ассоциативный вычислитель // Электронное моделирование. 2011. № 1. С. 73-90.

[7] Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S. Information analysis infrastructure for diagnosis. Information // An international interdisciplinary journal. Japan, 2011. Vol. 14. № 7. P. 2419-2433.

[8] Хаханов В.И. Проектирование и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. Харьков: ХНУРЭ, 2009. 484 с.

[9] Stig Stenholm, Kalle-Antti Suominen. Quantum approach to informatics. Published by John Wiley & Sons, Inc., Hoboken, New Jersey. 2005. 238 p.

[10] Michael A. Nielsen & Isaac L. Chuang. Quantum Computation and Quantum Information. Published in the United States of America by Cambridge University Press, New York. 2010. 676 p.