УДК658.512.011:681.326:519.713

# КВАНТОВЫЕ МОДЕЛИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

*ХАХАНОВ В.И., МУРАД АЛИ А., ЛИТВИНОВА Е.И., ГУЗЬ О.А., ХАХАНОВА И.В.*

Предлагаются кубитные (квантовые) структуры данных и вычислительных процессов для существенного повышения быстродействия при решении задач дискретной оптимизации и отказоустойчивого проектирования. Описываются 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 диаграммы Хассе. Предлагается прототип квантового устройства, реализованного на основе программируемой логики.

## 1. Введение

Квантовые вычисления в последние 10 лет становятся интересными для рынка электронных технологий благодаря 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]. 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 Интернета [11]; средств синтеза отказоустойчивых цифровых примитивов и систем [12]; тестирования и моделирования цифровых систем на кристаллах [13-15]; технологий защиты информации и компьютерных систем [5-7].

## 2. 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 в том, что множество элементов в нем все равно 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, множество 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:

$$Q \cup E = 1000 \vee 0100 = 1100 = A;$$

$$S \cap V = 1001 \wedge 1110 = 1000 = Q;$$

$$\widetilde{B} = \overline{0011} = 1100 = A;$$

$$F \Delta P = 1011 \oplus 0110 = 1101 = Y;$$

$$H \Delta J = 0010 \oplus 0001 = 0011 = B;$$

$$F \Delta Y = 1011 \oplus 1111 = 0100 = Y;$$

$$F \Delta F = 1011 \oplus 1011 = 0000 = U(\varnothing);$$

Другая интерпретация булеана из четырех примитивов, представленная ниже:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 01 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

создает 16 различных функций от двух переменных. В то же время последнюю таблицу можно представить как коды символов многозначного алфавита, которыми легко оперировать для решения задач синтеза и анализа булевых функций:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| E | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| H | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| J | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | ∅ | J | H | B | E | I | P | C | Q | S | O | F | A | L | V | Y |

Такую таблицу легко построить для любого числа примитивов (n=2, 3,4, 5, 6, 7, 8 …), где теоретико-множественные операции над символами сводятся к логическим операциям над векторами. Два наиболее традиционных примитива (0,1) создают известный алфавит Кантора:

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| | ∅ | 1 | 0 | X |

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:

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

$$\begin{array}{|cc|}\hline 00&0\\01&0\\10&0\\11&1\\\hline\end{array}=\begin{array}{|cc|}\hline Q&0\\E&0\\H&0\\J&1\\\hline\end{array}=\begin{array}{|cc|}\hline V&0\\J&1\\\hline\end{array}=\begin{array}{|cc|}\hline 1110&0\\0001&1\\\hline\end{array};$$

$$\begin{array}{|cc|}\hline 00&0\\01&0\\10&1\\11&0\\\hline\end{array}=\begin{array}{|cc|}\hline Q&0\\E&0\\H&1\\J&0\\\hline\end{array}=\begin{array}{|cc|}\hline F&0\\H&1\\\hline\end{array}=\begin{array}{|cc|}\hline 1101&0\\0010&1\\\hline\end{array};$$

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

$$\begin{array}{|cc|}\hline 00&1\\01&1\\10&1\\11&1\\\hline\end{array}=\begin{array}{|cc|}\hline Q&1\\E&1\\H&1\\J&1\\\hline\end{array}=\begin{array}{|cc|}\hline Y&1\\\hline\end{array}=\begin{array}{|cc|}\hline 1111&1\\\hline\end{array};$$

$$\begin{array}{|cc|}\hline 00&0\\01&0\\10&1\\11&0\\\hline\end{array}=\begin{array}{|cc|}\hline Q&0\\E&0\\H&1\\J&0\\\hline\end{array}=\begin{array}{|cc|}\hline F&0\\H&1\\\hline\end{array}=\begin{array}{|cc|}\hline 1101&0\\0010&1\\\hline\end{array};$$

$$\begin{array}{|cc|}\hline 00&0\\01&1\\10&1\\11&1\\\hline\end{array}=\begin{array}{|cc|}\hline Q&0\\E&1\\H&1\\J&1\\\hline\end{array}=\begin{array}{|cc|}\hline Q&0\\C&1\\\hline\end{array}=\begin{array}{|cc|}\hline 1000&0\\0111&1\\\hline\end{array};$$

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

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, формирующих единичное и нулевое значение выхода функции. Последующее двоичное кодирование входного символа куба 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 ДНФ, которая не обратима 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 стано-

вятся одновходовыми, где входом является регистровая переменная или n-разрядный вектор, который формирует адрес памяти, хранящей $Q = 2^n$ бит, как значений функции $Y = f(A) = f(x_1, x_2, ..., x_i, ..., x_n)$.

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 – 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. 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 [1], абсолютно пригодная для измерения взаимодействия многозначных (двоичных) кубитных объектов, процессов и явлений путем использования xor-операции.

В идеале применение кубитной структуры дает возможность любую функциональность представить в виде двух кубов, привязанных к нулю и единице. Такие кубы формируют 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 инверсны, 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:

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

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

Полученная структура 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 их выполнения за счет минимального числа кубов в покрытиях функциональных элементов.

Таким образом, на основании введенных кубитных структур данных и Хассе-модели вычислительного процесса можно сделать некоторые выводы:

1) Quantum Computer создавали специалисты в области квантовой механики, которые привнесли идею создания нечислового компьютера на аналоговой основе представления информации.

2) Введенное понятие кубита соответствует булеану примитивов, что является идеальной нечисленной формой описания компонентов объекта для их анализа, синтеза и оптимизации дискретных объектов.

3) 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.

4) На практике более 90% всех задач IT-индустрии относится к области дискретной математики, где нет места числам.

5) Необходимо создавать ассоциативные логические мозгоподобные высокопараллельные (квантовые) процессоры, эффективно оперирующие примитивами булеана (кубита) или элементами и множествами для решения задач дискретной математики.

6) 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.

7) Другое решение организации вычислительных процессов связано с топологическим представлением кубита, где элементами выступают геометрические фигуры.

8) Нечисленные задачи, ориентированные на предлагаемый процессор: минимизация форм (булевых функций) описания сложных систем; поиск путей в графе; тестирование и диагностика цифровых систем; комбинаторные исследования процессов и явлений; интеллектульный поиск данных, распознавание образов и принятие решений; дискретизация фаззи-моделей и методов в задачах создания интеллекта.

### 3. Кубит-процессор (квантовый процессор)

Цель создания кубит-процесора – существенное уменьшение времени при решении задач оптимизации путем параллельного вычисления векторных логических операций над множеством всех подмножеств от примитивных компонентов за счет увеличения памяти для хранения промежуточных данных.

Задачи: 1) Определение структур данных для взятия булеана при решении задачи покрытия столбцов матрицы $M = \left|M_{ij}\right|, i = \overline{1,m}; j = \overline{1,n}$ единичными значениями строк. В частности, при m = n = 8 необходимо параллельно выполнить логическую операцию над 256 вариантами всех возможных сочетаний векторов (строк матрицы), составляющих булеан. 2) Система команд процессора должна включать операции (and, or, xor) над векторами (словами), размерности m. 3) Разработка архитектуры кубит-процессора для параллельного вычисления $2^n - 1$ вариантов сочетаний, направленных на оптимальное решение NP-полной задачи покрытия. 4) Реализация прототипа кубит-процессора на базе программируемой логики PLD и верификация (валидация) аппаратного решения на примерах минимизации булевых функций. 5) Приведение других практических задач дискретной оптимизации к форме задачи покрытия для последующего решения на кубит-процессоре.

В качестве примера предлагается решить задачи поиска оптимального единичного покрытия всех столбцов минимальным числом строк матрицы M, представленной ниже:

| M | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 1 | . | . | . | . | 1 | . | . |
| b | . | . | 1 | . | . | . | 1 | . |
| c | 1 | . | . | . | 1 | . | 1 | . |
| d | . | 1 | . | 1 | . | . | . | 1 |
| e | . | 1 | . | . | 1 | . | . | . |
| f | 1 | . | 1 | . | . | 1 | 1 | . |
| g | . | 1 | . | 1 | . | . | . | 1 |
| h | . | . | 1 | . | 1 | . | . | . |

Для этого необходимо сделать перебор всех 255 сочетаний: из восьми по одной строке, по двум, трем, четырем, пяти, шести, семи и восьми. Минимальное количество примитивов (строк), формирующее покрытие, есть оптимальное решение. Таких решений может быть несколько. Диаграмма Хассе есть компромиссное предложение относительно времени и памяти, или такая стратегия решения задачи покрытия, когда ранее полученный результат впоследствии используется для создания более сложной суперпозиции. Поэтому для каждой таблицы покрытия, содержащей n примитивов (строк), необходимо генерировать собственную мультипроцессорную структуру в форме диаграммы Хассе, которая далее должна быть использована для почти параллельного решения NP-полной задачи. Например, для четырех строк таблицы покрытия диаграмма Хассе – структура мультипроцессора – будет иметь следующий вид (рис.1).

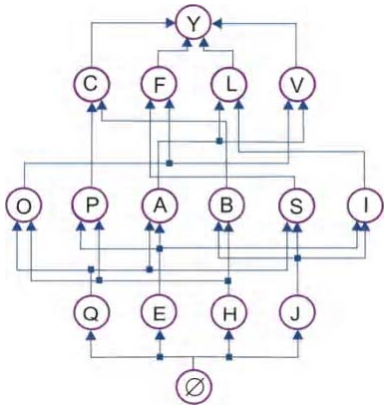


Рис. 1. Хассе-структура вычислительных процессов

Оптимальные решения задачи покрытия для матрицы M представлены сочетанием строк:

$$C = fgh \vee efg \vee cdf .$$

Достоинства кубит Хассе-процессора (Cubit Hasse Processor – CHP) заключаются в возможности использовать только двухвходовые схемы векторных логических операций (and, or, xor), а значит, в существенном уменьшении стоимости по Квайну реализации процессорных элементов (вершин) и памяти за счет применения последовательных вычислений и незначительного увеличения времени обработки всех вершин графа Хассе. Для каждой вершины используется критерий качества покрытия – наличие всех единиц в координатах вектора-результата. Если критерий качества выполняется, то все остальные вычисления можно не производить, поскольку диаграмма Хассе есть строго иерархическая структура по числу сочетаний в каждом ярусе. Это означает, что самое лучшее решение находится на более низком уровне иерархии. Варианты одного уровня равнозначны по реализации (стоимости), поэтому полученное первое качественное покрытие ( $Q = \sum_{i=1}^{n} q_i = n$ ) есть лучшее решение, предполагающее остановку всех последующих вычислений по стратегии диаграммы Хассе. Учитывая последовательно-параллельную стратегию, анализ вершин графа, время (цикл) обработки всех примитивов CH-процессора определяется числом уровней иерархии (количеством битов (примитивов, строк в таблице покрытия) в кубитной переменной), умноженным на время анализа одной вершины: $T = \log_2 2^n \times t = t \times n$ . При этом длина m строки таблицы покрытия не влияет на оценку быстродействия. Анализ вершины включает две команды: логическую (and, or, xor) и операцию вычисления критерия качества покрытия в форме скаляра путем применения функции and ко всем разрядам вектора-результата:

$$m_{ir,j} = M_{i,j} \vee M_{r,j}, (j = \overline{1,n}; \{i \neq r\} = \overline{1,m;});$$

$$m_{ir}^{s} = \wedge m_{ir,j} = \wedge (M_{i,j} \vee M_{r,j}).$$

Аппаратные затраты на реализацию СН-процессора зависят от суммарного числа вершин графа Хассе и от количества битов (разрядов) в строке таблицы покрытия:

$$H = 2^n \times k \times m,$$

где k – коэффициент аппаратной реализации (сложности) одного разряда бинарной векторной логической операции и последующей команды вычисления критерия качества покрытия.

Таким образом, высокое быстродействие решения задачи покрытия достигается существенным повышением аппаратурных затрат (в $2^n \times k \times m / k \times m \times n = 2^n / n$ раз по сравнению с последовательной обработкой графовых вершин), которое обеспечивает компромиссный вариант между полностью параллельной структурой вычислительных процессов (здесь затраты аппаратуры определяются числом примитивов в каждой вершине $H = k \times m \times n \times 2^n$, а увеличение аппаратуры составляет $2^n$ раз) и последовательными вычислениями однопроцессорного компьютера (здесь быстродействие обработки графа Хассе равно $T = t \times 2^n$, а аппаратные затраты равны $H^* = k \times m \times n$). Уменьшение аппаратуры по сравнению с параллельным вариантом обработки графа составляет

$$Q^H = k \times m \times n \times 2^n / k \times m \times 2^n = n.$$

Как следствие существенной аппаратной избыточности, уменьшение времени анализа вершин графа по сравнению с последовательной обработкой структуры имеет следующую оценку:

$$Q^T = \frac{t \times 2^n}{t \times n} = \frac{2^n}{n}.$$

## 4. Практическая реализация

Модель квантового устройства разработана на языке Verilog. Элементарная ячейка процессора состоит из двух регистровых вентилей (рис.2). Регистровый элемент ИЛИ выполняет логическую операцию над двумя векторами, формируя вектор результата. Регистровый вентиль И выполняет свертку всех битов вектора по операции И, формируя однобитовый элемент, идентифицирующий оптимальное решение задачи покрытия.

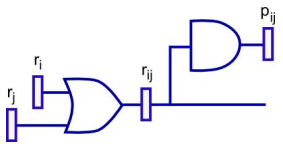


Рис 2. Элементарная ячейка квантового процессора

Фрагмент упрощенной схемы квантового процессора изображен на рис. 3. Здесь представлено формирование значений для вершин диаграммы Хассе шести уровней.

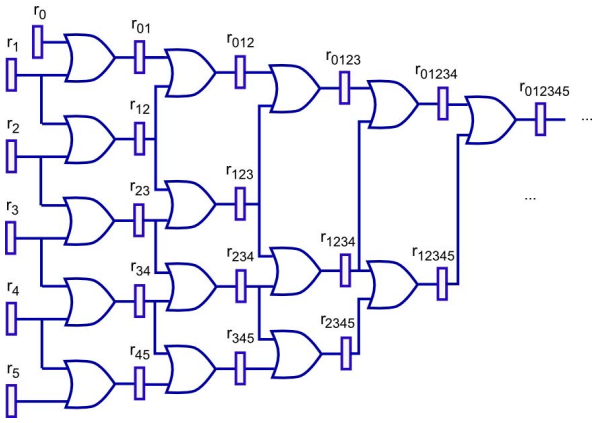


Рис.3. Фрагмент схемы RTL-уровня

Реализация устройства выполнена на кристалле FPGA фирмы Xilinx xc3s1600e-4-fg484.

Листинг Мар-отчет

Logic Utilization:

Number of Slice Flip Flops: 2,286 out of 29,504 7%

Number of 4 input LUTs: 2,715 out of 29,504 9%

Logic Distribution:

Number of occupied Slices: 1,514 out of 14,752 10%

Number of Slices containing only related logic: 1,514 out of 1,514 100%

Number of Slices containing unrelated logic: 0 out of 1,514 0%

*See NOTES below for an explanation of the effects of unrelated logic.

Total Number of 4 input LUTs: 2,715 out of 29,504 9%

Number of bonded IOBs: 321 out of 376 85%

Number of BUFGMUXs: 1 out of 24 4%

Временные параметры проекта:

Tclk_to_clk = 4.672 ns

Tclk_to_pad_max = 11.552 ns

Period = max{ Tclk_to_clk , Tclk_to_pad_max };

Period = 11.552 ns

Fclk = 86,5 МГц

## 5. Заключение

Реализация квантового процессора на основе структуры Хассе позволила в n раз уменьшить аппаратурные затраты, что, в свою очередь, уменьшило быстродействие процессора также в n раз. Вывод – необходимо искать новые структуры данных для снижения аппаратной стоимости квантовых вычислений, или более интеллектуальные алгоритмы решения задачи покрытия на диаграммах Хассе.

*Научная новизна* – впервые предложена модель данных и аппаратной реализации квантового компьютера, которая характеризуется использованием структуры Хассе, что дает возможность существенно (x100)

повысить быстродействие решения практических задач дискретной оптимизации.

*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 компьютера.

**Литература: 1.** *Beth T.* Quantum computing: an introduction // Proc. of the IEEE International Symposium on Circuits and Systems, ISCAS. 2000. Geneva. 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.** *Glassner A.* Quantum computing. 3. // IEEE Computer Graphics and Applications. Nov/Dec 2001. Vol. 21, Issue 6. P. 72 – 82. **5.** *Marinescu D.C.* The Promise of Quantum Computing and Quantum Information Theory – Quantum Parallelism // Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05). 2005. P. 112-114. **6.** *Oskin M., Chong F.T., Chuang I.L.* A practical architecture for reliable quantum computers // Computer. Jan. 2002. Vol. 35, No.1. P.79-87. **7.** *Glassner A.* Quantum Computing, Part 1 // IEEE Computer Graphics and Applications. July/August 2001. P.84-92. **8.** *Aliferis P., Brito F., DiVincenzo D.P., Preskill J., Steffen M., Terhal B. M.* Fault-tolerant computing with biased-noise superconducting qubits // New Journal of Physics. January 30, 2009. 19 p. http://iopscience.iop.org/1367-2630/11/1/013061/pdf/1367-2630_11_1_013061.pdf **9.** *Huntting B., Mertz D.* Introduction to Quantum Computing // http://www.ibm.com/developerworks/linux/library/l-quant/index.html. **10.** *DiVincenzo D.P.* The Physical Implementation of Quantum Computation // IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA. July 9, 2004. Preprint available online http://arxiv.org/PS_cache/quant-ph/pdf/0002/0002077v3.pdf **11.** *Инфраструктура* мозгоподобных вычислительных процессов / М.Ф. Бондаренко, О.А. Гузь, В.И. Хаханов, Ю.П. Шабанов-Кушнаренко. Харьков: Новое Слово, 2010. 160 с.**12.** *Mark Gregory Whitney.* Practical Fault Tolerance for Quantum Circuits. PhD Dissertation in Computer Science. Berkeley: University of California. 2009. 206p. **13.** *Хаханов В.И., Литвинова Е. И., Чумаченко С. В., Гузь О.А.* Логический ассоциативный вычислитель //Электронное моделирование. 2011. № 1. С. 73-90. **14.** *Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S.* Information analysis infrastructure for diagnosis. Information. An international interdisciplinary journal. 2011. Vol. 14, No 7. P. 2419-2433. **15.** *Хаханов В.И.* Проектирование и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. Харьков: ХНУРЭ, 2009. 484с.

Поступила в редколлегию 26.09.2011

**Рецензент:** д-р техн. наук, проф. Филатов В.А.

**Хаханов Владимир Иванович**, декан факультета компьютерной инженерии и управления, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

**Мурад Али А.**, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: компьютерные системы и сети. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

**Литвинова Евгения Ивановна**, заместитель декана факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем. Увлечения: плавание, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-421. E-mail: kiu@kture.kharkov.ua.

**Гузь Олеся Алексеевна,** канд. техн. наук, доцент кафедры СКС Донецкой Академии автомобильного транспорта. Научные интересы: техническая диагностика цифровых систем.Адрес: Украина, 83086, Донецк, пр. Дзержинского, 7.

**Хаханова Ирина Витальевна**, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-421.