УДК 681.326:519.713

В.И. Хаханов [1], д-р техн. наук,
Багхдади Аммар Авни Аббас [2], аспирант,
Е.И. Литвинова [1], д-р техн. наук, **А.С. Шкиль** [1], канд. техн. наук
[1] Харьковский национальный университет радиоэлектроники
(Украина, 61166, Харьков, пр-т Ленина, 14,
тел. (057) 7021326, e-mail: hahanov@kture.kharkov.ua),
[2] Ирак, Багдадский университет

Кубитные структуры данных вычислительных устройств *

Разработаны кубитные модели и методы повышения быстродействия программных и аппаратных средств анализа цифровых устройств при увеличении размерности структур данных и памяти. Введены основные понятия, термины и определения, необходимые для имплементации квантовых вычислений в практику анализа виртуальных компьютеров. Представлены результаты исследований в области проектирования и моделирования компьютерных систем в киберпространстве с использованием двухкомпонентного автомата <память, транзакции>.

Розроблено кубітні моделі та методи підвищення швидкодії програмних і апаратних засобів аналізу цифрових пристроїв при збільшенні розмірності структур даних і пам'яті. Введено основні поняття, терміни та визначення, необхідні для імплементації квантових обчислень у практику аналізу віртуальних комп'ютерів. Наведено результати досліджень з проектування і моделювання комп'ютерних систем у кіберпросторі з використанням двокомпонентного автомата <пам'ять, транзакції>.

Ключевые слова: кубитные структуры, квантовые вычисления, автомат «память-транзакция», моделирование цифровых систем, процессорные матрицы.

Рыночная привлекательность эмуляции квантовых методов вычислений при создании виртуальных (облачных) компьютеров (ВК) в киберпространстве основана на использовании кубитных моделей данных, ориентированных на параллельное решение задач дискретной оптимизации при существенном повышении затрат памяти. Не рассматривая физические основы квантовой механики, касающиеся недетерминированного взаимодействия атомных частиц [1—4], используем понятие кубита как двоичного или многозначного вектора для совместного и одновременного зада-

* Публикуется в порядке дискуссии.

ния булеана состояний дискретной области киберпространства на основе линейной суперпозиции унитарных кодов, ориентированных на параллельное использование методов анализа и синтеза компонентов киберпространства. В быстро развивающейся теории квантовых вычислений векторы состояний образуют квантовый регистр из n кубитов, формирующих унитарное или гильбертово пространство H, размерность которого имеет степенную зависимость от числа кубитов $\mathrm{Dim}\ H = 2^n$.

Мотивация нового подхода для проектирования ВК обусловлена появлением облачных сервисов, которые представляют собой специализированные и рассредоточенные в пространстве виртуальные компьютерные системы, реализуемые в аппаратуре или в программном продукте. Программисту уже не всегда интересно, как и куда имплементируются коды программных приложений, как и нет необходимости знать теорию проектирования цифровых автоматов на уровне вентилей, регистровых передач и использовать специфические компоненты вычислительной техники (триггеры, регистры, счетчики, мультиплексоры). Любой компонент функциональности представляется векторной формой таблицы истинности, реализуемой с помощью памяти. Логические функции в традиционном исполнении не рассматриваются. От этого частично уменьшается быстродействие, но, учитывая, что 94 % SoC-кристалла составляет память [5], оставшиеся 6 % также можно реализовать на памяти, что не будет критичным для большинства программных и аппаратных приложений. Поэтому практически для программирования эффективных ВК можно использовать теорию, основанную на двух компонентах более высокого уровня абстракции: память и транзакция.

Особенность организации данных в классическом компьютере состоит в том, что каждый бит, байт или другой компонент имеет свой адрес. Поэтому существует проблема эффективной обработки ассоциации (конечного алфавита символов) равных по значимости элементов, которые не имеют порядка по определению, например множество всех подмножеств. Решением может быть процессор, где элементарной ячейкой является образ или шаблон универсума из n унитарно кодированных примитивов, использующих суперпозицию для формирования булеана $|B(A)| = 2^n$ всех возможных состояний такой ячейки в виде множества всех подмножеств [6, 7].

Наблюдается определенная аналогия по векторным структурам данных булеана с кубитом квантового компьютера (КК). Понятию квантового кубита как элементарной структуры для хранения информации в КК, разрешающей суперпозицию состояний $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, в классическом компьютере [1] можно поставить во взаимно однозначное соответствие булеан состояний, формирующий, например, алфавит Кантора $A^k =$

$= \{0, 1, X, \varnothing\}$, $X = \{0,1\}$. В нем два первых примитива алфавита унитарно кодируются векторами (упорядоченными двоичными последовательностями): 0 = (10) и 1 = (01). Коды остальных двух символов можно получить как производные с помощью операции суперпозиции или объединения (логического сложения) (10) $\vee$ (01) = (11), а также операции пересечения (логического умножения) (10) $\wedge$ (01) = (00), что формирует четыре состояния булеана: {(10), (01), (11), (00)} [8] в виде двоичных векторов или кортежей.

Корректность использования определения «кубитная модель» для моделей цифровых устройств (ЦУ) можно обосновать в результате сравнения алгебры теории множеств и линейной, например алгебры Кантора $A^k = \{0,1, X, \varnothing\}$. Первые два символа есть примитивы или первичные элементы, которые не раскладываются на составляющие. Третий символ — производная в теории множеств, определяемая суперпозицией или объединением символов-примитивов: $X = 0 \bigcup 1$. Естественно, что в гильбертовом пространстве, где линейная алгебра оперирует примитивами $\alpha|0\rangle$ и $\beta|1\rangle$ кубита, третий символ также определяется суперпозицией двух составляющих вектора состояния: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Наблюдаемая полная аналогия, будучи продолжена на образование четвертого, недостающего в кубитной (линейной) алгебре символа, соответствующего пустому множеству, позволяет оперировать функцией, обратной суперпозиции. В теории множеств такой операцией является пересечение, определяющее пустое множество на символах-примитивах: $\varnothing = 0 \bigcap 1$. В гильбертовом пространстве такой операцией является скалярное (внутреннее) произведение или функция (обозначение) Дирака $\langle a | b \rangle$ [9], которая имеет геометрическую интерпретацию в виде $\langle a | b \rangle \approx |a| \times |b| \times \cos \angle(a, b)$. Если проекции a и b вектора квантового состояния ортогональны, а это действительно так, то получаем ожидаемый результат: $\langle a | b \rangle \approx |a| \times |b| \times \cos \angle(a, b) = |a| \times |b| \times \cos \angle 90° = 0$, т.е. скалярное произведение ортогональных векторов равно нулю. Этот результат является аналогом символа пустого множества при выполнении операции пересечения в алгебре множеств.

Таким образом, таблица соответствия алгебры множеств и линейной алгебры подтверждает свойства изоморфизма между символами булеана и состояниями кубит-вектора, полученными с помощью изоморфных в алгебрах операций объединения — суперпозиции и пересечения — скалярного произведения:

Булеан A^k	0	1	$X = 0 \bigcap 1$	$\varnothing = 0 \bigcap 1$
Кубит $\|\psi\rangle$	$\|0\rangle$	$\|1\rangle$	$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|0\rangle \|\beta\|1\rangle$

.

Следовательно, структуру данных «булеан» можно рассматривать как детерминированный (невероятностный) образ или аналог квантового кубита в алгебре множеств (логики), примитивные элементы которой унитарно кодируются двоичными векторами (кортежами) и обладают свойствами суперпозиции, параллелизма и перепутывания. Это дает возможность использовать предлагаемые модели для повышения быстродействия анализа ЦУ на классических вычислителях, а также без существенной модификации — в КК, которые непременно появятся через 5—10 лет на рынке электронных технологий.

В работах [1—4] рассматривается возможность эмуляции классических вычислительных процессов на КК. Однако с учетом «реверсивности» или обратимости соответствия между упомянутыми алгебрами предлагается обратное преобразование — эмуляция некоторых преимуществ (суперпозиция, параллелизм, неопределенность) квантовых вычислений на классических процессорах с использованием булеана как основной модели структур данных для интерпретативного описания и моделирования цифровых систем (ЦС).

На рынке электронных технологий существует конкуренция между базами имплементации идеи [10]:

1) гибкая (мягкая) реализация проекта связана с синтезом интерпретативной модели программной формы функциональности или в аппаратном исполнении программируемых логических устройств на основе FPGA, CPLD; преимущества заключаются в технологичности модификации проекта, недостатки — в невысоком быстродействии функционирования ЦС;

2) жесткая реализация ориентирована на использование компилятивных моделей при разработке программных приложений или на имплементацию проекта в кристаллах VLSI [5, 10]. Преимущества и недостатки жесткой реализации инверсны по отношению к мягкому исполнению проектов: высокое быстродействие и невозможность модификации.

С учетом изложенных базовых вариантов реализации идеи предлагаются квантовые структуры данных, ориентированные на повышение быстродействия гибких моделей программного или аппаратного исполнения проекта.

Квантовые структуры описания ЦС. Кубит (n-кубит) есть векторная форма унитарного кодирования универсума из n примитивов для задания булеана состояний 2^{2^n} с помощью 2^n двоичных переменных. Например, если $n = 2$, то 2-кубит задает 16 состояний с помощью четырех переменных. Если $n = 1$, то кубит задает четыре состояния на универсуме из двух примитивов, (10) и (01), с помощью двух двоичных переменных: (00, 01, 10, 11) [1, 10]. При этом допускается суперпозиция (одновремен-

ное существование) в векторе 2^n состояний, обозначенных примитивами. Кубит (n-кубит) дает возможность использовать параллельные логические операции вместо поэлементных теоретико-множественных для существенного ускорения процессов анализа дискретных систем.

Далее кубит отождествляется с n-кубитом или двоичным вектором, если это не мешает пониманию излагаемого материала. Поскольку квантовые вычисления связаны с анализом кубитных структур данных, то далее будем применять определение «квантовый» для идентификации технологий, использующих три свойства квантовой механики: параллелизм обработки (двоичных векторов), суперпозицию состояний и их перепутывание. Синонимом кубита при задании двоичного вектора логической функции является Q-покрытие (Q-вектор) [8] как унифицированная векторная форма суперпозиционного задания выходных состояний, соответствующих адресным кодам входных переменных логического элемента.

Кубит в ЦС используется как форма задания структурного примитива, инвариантная к технологиям реализации функциональности (hardware, software). Более того, синтез ЦС на основе кубитных структур не привязан жестко к теореме Поста, определяющей пять условий (классов) существования функционально полного базиса [6]. На предлагаемом уровне абстракции n-кубит дает более широкие возможности для векторного задания любой n-входовой функции из булеана мощностью $|B(A)|=2^n$, непременно содержащего все функциональности, удовлетворяющие пяти классам теоремы Поста [6]. Формат структурного кубитного компонента цифровой схемы $Q^* = (X, Q, Y)$ включает интерфейс (входные и выходную переменные), а также кубит-вектор Q, задающий функцию $Y = Q(X)$, размерность которого определяется степенной функцией от числа входных линий $k = 2^n$.

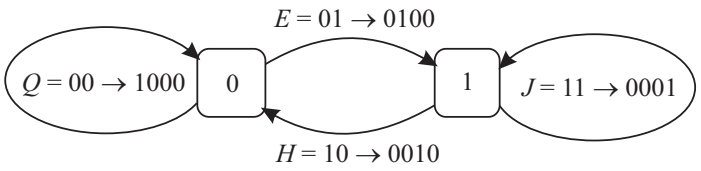
Практически ориентированная новизна кубитного моделирования заключается в замене таблиц истинности компонентов ЦУ векторами состояний выходов. Такие преобразования применительно к логическому элементу достаточно просты. Пусть функциональный примитив имеет двоичное покрытие (таблицу истинности)

$P =$

X_1	X_2	Y
0	0	1
0	1	1
1	0	1
1	1	0

,

которое можно трансформировать унитарным кодированием входных векторов на основе использования двухтактного алфавита [10, 11], первоначально предназначенного для компактного описания всех возможных



$Q = 00 \to 1000$	$E = 01 \to 0100$	$H = 10 \to 0010$	$J = 11 \to 0001$
$O = \{Q, H\} =$ $= \{00,10\} \to 1010$	$I = \{E, J\} =$ $= \{01,11\} \to 0101$	$A = \{Q, E\} =$ $= \{00,01\} \to 1100$	$B = \{H, J\} =$ $= \{01,11\} \to 0101$
$S = \{Q, J\} =$ $= \{00,11\} \to 1001$	$P = \{E, H\} =$ $= \{01,10\} \to 0110$	$C = \{E, H, J\} =$ $= \{01,10,11\} \to 0111$	$F = \{Q, H, J\} =$ $= \{00,10,11\} \to 1011$
$L = \{Q, E, J\} =$ $= \{00,01,11\} \to 1101$	$V = \{Q, E, H\} =$ $= \{00,01,10\} \to 1110$	$Y = \{Q, E, H, J\} =$ $= \{00,01,10,11\} \to 1111$	$\varnothing = 00 \to 0000$

Рис. 1. Двухтактный алфавит автоматных переменных и интерпретация символов

переходов автоматных переменных. На рис. 1 представлены символы, их двоичные и унитарные коды, предназначенные для описания двух соседних состояний автоматных переменных. Предложенный алфавит структурно задает булеан (множество всех подмножеств) состояний на универсуме из четырех примитивов $Y = \{Q, E, H, J\}$. Унитарный код соответствует формату вектора, содержащего два кубита, с помощью которых формируются 16 символов двухтактного алфавита. Используя последний, любое покрытие функционального двухвходового логического примитива можно представить двумя кубами или даже одним, учитывая, что они взаимно инверсны:

$$P = \left[\begin{array}{c|c} 00 & 1 \\ 01 & 1 \\ 10 & 1 \\ 11 & 0 \end{array}\right] = \left[\begin{array}{c|c} Q & 1 \\ E & 1 \\ H & 1 \\ J & 0 \end{array}\right] = \left[\begin{array}{c|c} V & 1 \\ J & 0 \end{array}\right] = \left[\begin{array}{c|c} 1110 & 1 \\ 0001 & 0 \end{array}\right] \to \boxed{1 \;\; 1 \;\; 1 \;\; 0}.$$

Сначала кодируются все пары символами двухтактного алфавита, затем выполняется объединение первых трех кубов по правилу (минимизации) оператора сограней [5]: векторы, отличающиеся в одной координате, объединяются в один. Далее полученное покрытие из двух кубов кодируется соответствующими данным символам кубитными векторами. Для моделирования исправного поведения достаточно иметь только один куб (нулевой или единичный), так как второй всегда является дополнением к первому. Следовательно, ориентируясь, например, на единичный куб, формирующий на выходе единицу, можно исключить бит состояния выхода примитива, что уменьшит размерность куба или модели прими-

тива до числа адресуемых состояний элемента, где адрес есть вектор, составленный из двоичных значений входных переменных, по которому определяется состояние выхода примитива. Ввиду тривиальности нет смысла показывать, что по аналогии любую таблицу истинности можно привести к кубитной функциональности в форме вектора выходных состояний логического элемента, имеющего n входов. Кроме того, двухтактный алфавит используется для компактного описания таблиц переходов конечных автоматов, что существенно уменьшает время анализа таких моделей. Например, полный ориентированный граф из 16 переходов на четырех вершинах {00, 01, 10, 11} минимизируется в один куб: {QQ = = (00–00), QE = (00–01), EE = (00–11), EQ = (00–10), QJ = (01–01), EJ = = (01–11), QH = (01–00), EH = (01–10), JJ = (11–11), HJ = (11–01), JH = = (11–10), HH = (11–00), JQ = (10–10), HQ = (10–00), JE = (10–11), HE = = (10–01)} = YY. Другой пример — описание DC-триггера в двухтактном алфавите можно легко свести всего к трем кубам покрытия, имеющим всего девять координат в минимальной таблице вместо 48 в исходном покрытии [11]:

C^{t-1}	C^t	D^{t-1}	D^t	Q^{t-1}	Q^t
1	0	1	1	X	1
0	0	X	X	1	1
0	1	X	X	1	1
1	1	X	X	1	1
1	0	0	0	X	0
0	0	X	X	0	0
0	1	X	X	0	0
1	1	X	X	0	0

=

C	D	Q
H	J	I
Q	Y	J
E	Y	J
J	Y	J
H	Q	O
Q	Y	Q
E	Y	Q
J	Y	Q

=

C	D	Q
H	J	I
L	Y	S
H	Q	O

.

Интерпретация трех кубов полученного покрытия:

1) задний фронт синхросигнала $C = H$ при неизменной единице на $D = J$ обеспечивает установку выхода триггера в единицу;

2) отсутствие заднего фронта на синхровходе $C = L$ сохраняет состояние выхода триггера;

3) задний фронт синхросигнала $C = H$ при неизменном нуле на $D = Q$ обеспечивает установку выхода триггера в нуль.

Компактность описания автоматных примитивов на примере DC-триггера является безусловным фактом эффективного использования кубитных (булеановых) структур данных для задания и анализа цифровых компонентов. Для счетных структур уменьшение объемов покрытий в двухтактном алфавите [10, 11] по сравнению с исчислением Рота (троичным алфавитом) достигает десятков раз [12, 13].

Таким образом, *n*-кубит отличается от классического байта или бита суперпозиционной структуризацией двоичного вектора, способного одновременно хранить *n* состояний (символов) функциональности на булеане мощностью $|B(A)|=2^n$ примитивов и параллельно выполнять логические операции над теоретико-множественными данными векторного формата. Например, операция симметрической разности над подмножествами $A = \{a, b, c, d, e, f\}$ и $B = \{a, c, f, g, h, k\}$ выполняется параллельно за один такт xor-операции, если каждый элемент представлен унитарным кодом, а подмножества — соответствующими векторами, которые являются в данном случае кубит-операндами для вычисления симметрической разности:

9 – кубит	a	b	c	d	e	f	g	h	k
A	1	1	1	1	1	1	0	0	0
B	1	0	1	0	0	1	1	1	1
$A \oplus B$	0	1	0	1	1	0	1	1	1

.

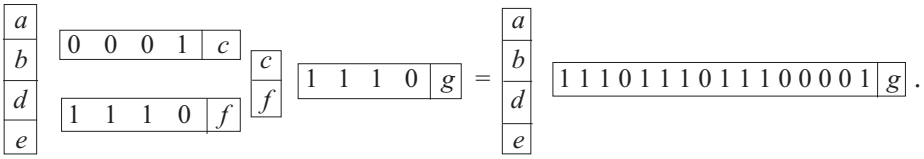
Изложенное выше дает основание для определения структурированного вектора как *n*-кубита, так как он обладает свойствами параллелизма вычислений, суперпозиционирования и неопределенности (перепутывания — здесь в каждом векторе шесть единиц) состояний, наблюдаемых в квантовых структурах данных.

Процедура моделирования на *Q*-векторе функциональности сводится к записи в выходную булеву переменную *Y* состояния бита *Q*-вектора, адрес которого сформирован на основе конкатенации значений входных переменных: $Y = Q(X) = Q(X_1 * X_2 ... * X_j ... * X_k)$. Для моделирования цифровых схем вводится *M*-вектор эквипотенциальных линий как аналог *Q*-вектора, задающий состояния всей вычислительной системы (ВС), который связывает *Q*-векторы логических примитивов (*Q*-примитивов) в структуру с помощью нумерации входных и выходных переменных каждого функционального элемента [10, 11]. Процедура обработки последнего определяется выражением $M(Y) = Q[M(X)] = Q[M(X_1 * X_2 ... * X_j ... * X_k)]$. С учетом сквозной нумерации *Q*-примитивов универсальная процедура моделирования текущего *i*-элемента будет иметь формат $M(Y_i) = Q_i[M(X_i)] = Q_i[M(X_{i1} * X_{i2} ... * X_{ij} ... * X_{ik_i})]$.

В данном случае существенно упрощается алгоритм анализа ЦС, который сводится к процедуре формирования адреса, что дает возможность в $2^n - 1$ раз повысить быстродействие интерпретативного моделирования, заменив таблицы истинности примитивов *Q*-векторами описания только выходных состояний. Кроме того, если логический элемент имеет *n* входов, то число строк таблицы истинности равно 2^n, и это означает, что ее размерность составляет $d = 2^n \times n$. Таким образом, если длина *Q*-вектора

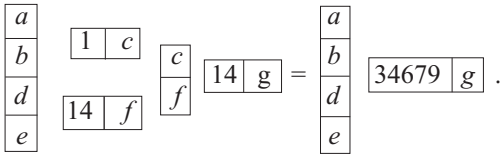
для любого логического примитива равна 2^n, то выигрыш в объеме памяти для его хранения и обработки составляет $r = \dfrac{2^n \times n}{2^n} = n$.

Синтез Q-покрытия на примере цифровой схемы $Y = \overline{ab \wedge \overline{de}} = \overline{ab} \vee de$ сводится к выполнению операции суперпозиции над Q-векторами примитивов, создающих структуру. Например, для трех примитивов (элементы and, and-not, and-not), составляющих схему, операция суперпозиции формирует Q-вектор всей функциональности, где его размерность будет больше, чем сумма Q-покрытий исходных примитивов:



При этом процедура моделирования Q-вектора структуры будет иметь более высокое быстродействие, так как она представлена только одним обращением к Q-покрытию для выемки содержимого из ячейки вместо трех, когда система представлена тремя примитивами. Выбор между структурной схемой и общей функциональностью (черный ящик) является выбором между размерностью модели и высоким быстродействием.

Трехэлементная схема, представленная выше Q-векторами, может быть задана схемотехнически (удобно для восприятия человеком), где вместо векторов будут фигурировать соответствующие десятичные номера:



При обработке такой формы функциональных покрытий необходимо преобразовать десятичный код в двоичный вектор и вычислить адрес ячейки, содержимое которой определяет состояние выходной переменной, в данном случае g. Естественно, что десятичный код существует на бумаге, а в компьютере — это всегда двоичный вектор. На самом деле гибкая схемотехника идентификации (нумерации) межсоединений имеет будущее, так как не связана с соединительными проводами, которые заменяются адресами или номерами линий. При этом создается структура ЦУ, обеспечивающая гибкость замены примитивов в случае обнаружения ошибок проектирования или дефектов.

Кубитное представление функциональных элементов дает также возможность ввести новые схемотехнические обозначения, связанные с деся-

тичным номером Q-вектора, задающего функциональность. Если система логических элементов имеет $n = 2$ входа, то число всех возможных функций $k = 2^{2^n}$, f — тип, или номер функционала (табл. 1). Более того, на основе множества кубитов первого уровня, задающих функции от двух переменных, можно ввести кубит второго уровня, унитарно кодирующий двухвходовые функции, что позволяет создавать структуру одновременного задания и анализа всех неупорядоченных состояний дискретной системы, где входными переменными являются функционалы первого уровня. Представленные в табл. 1 четыре вектора-примитива входных переменных (00, 01, 10, 11), образуют $k = 2^{2^2} = 2^4$ полное множество всех возможных функций, рассматриваемых в качестве примитивов второго уровня. Векторы-примитивы выходных переменных (16 столбцов от 0000 до 1111) формируют $k = 2^{2^4} = 2^{16}$ функциональных примитивов, входящих в состав более сложной дискретной системы, которые можно анализировать параллельно. Далее можно экстраполировать создание более сложной системы кубитов, где вектор Q = 0111110000001100, представленный в нижней строке табл. 1, будет рассматриваться как один из $k = 2^{2^{16}}$ примитивов третьего уровня иерархии. В каждом уровне иерархии кубитов число, или мощность, булеана состояний экспоненциально зависит от числа примитивов-векторов $k = 2^{2^n}$. Если вектор Q имеет все единичные значения, Q = = 1111111111111111, то он одновременно определяет пространство, содержащее 16 символов двухтактного алфавита, соответствующих булеану на универсуме из четырех примитивов [11].

Основная инновационная идея квантовых вычислений по сравнению с машиной фон Неймана заключается в переходе от вычислительных процедур над байт-операндом, определяющим в дискретном пространстве одно решение (точку), к квантовым параллельным процессам над кубит-операндом, одновременно формирующим булеан решений. В этом тезисе сформулировано будущее всех высокопроизводительных компьютеров для параллельного нецифрового анализа структур и сервисов дискретного киберпространства. Вычислительная сложность процедуры обработки множества из n элементов в «квантовом» процессоре и одного в машине

Таблица 1

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
f	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Q	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

фон Неймана одинакова вследствие соответствующего n-кратного повышения аппаратной сложности квантовой структуры данных.

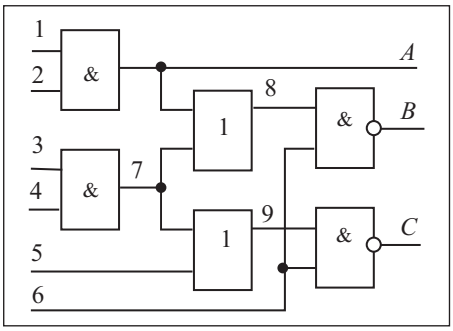


Рис. 2. Структура КС логических примитивов

Графовые структуры описания цифровых схем. Модельная схемотехника, не привязанная непосредственно к транзисторам, может быть представлена графовыми структурами, в которых каждая вершина (дуга) отождествляется с функциональным преобразованием, задаваемым Q-вектором. Тогда дуга (вершина) определяет взаимосвязи между функциональными Q-покрытиями, а также входные и выходные переменные. Реализация таких структур связана с ячейками памяти (LUT (Look Up Table) FPGA), которые способны хранить информацию в виде Q-вектора, где каждый бит или разряд имеет свой адрес, отождествляемый с входным словом. Тем не менее, программная реализация таких структур становится конкурентоспособной по быстродействию на рынке промышленных систем проектирования ЦС на кристаллах в результате адресной реализации процессов моделирования функциональных примитивов. Кроме того, аппаратная поддержка систем проектирования в виде HES (Hardware Embedded Simulator) [14] приобретает новую мотивацию на системном уровне проектирования ЦУ, когда программные и аппаратные решения имеют один и тот же кубитный формат.

Рассмотрим комбинационную схему (КС), содержащую шесть примитивов и три различных логических элемента (рис. 2). Данной схеме соответствуют три универсальные графовые формы цифровой функциональности, в которых использованы Q-векторы для задания поведения логических примитивов (рис. 3).

Структура, представленная на рис. 3, *а*, содержит 12 линий (дуг), нагруженных на квантовые функциональности ($1 = 0001, 7 = 0111, 14 = 1110$). Она подобна традиционной структурно-функциональной модели КС. Граф на рис. 3, *б*, подобный модели регистровых передач [15], является обратным по отношению к первой структуре. Здесь горизонтальные дуги отождествляются с функциональностями, а вершины — с группами входных линий, объединенных в регистровые переменные посредством вертикальных дуг, состояния которых образуют двоичный вектор, используемый в качестве адреса для вычисления состояния логического элемента или более сложного функционала. Переменные, участвующие в формировании адреса для Q-вектора функциональности, можно объединить в

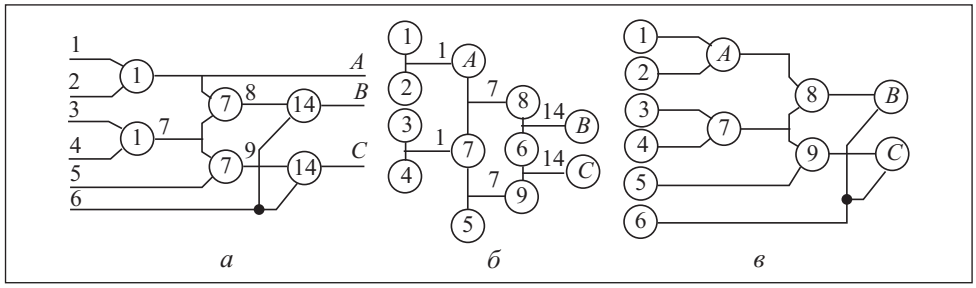


Рис. 3. Графовые формы квантовых функциональностей

одну вершину с указанием всех идентификаторов линий, создающих вектор-адрес. Регистровый граф КС ранжирован по уровням формирования входных сигналов, что обеспечивает параллелизм обработки элементов одного уровня и выполнимость итераций Зейделя [6, 11], повышающих быстродействие алгоритмов моделирования ЦС.

Структура, представленная на рис. 3, *б*, может быть использована для формализации как программных, так и аппаратных моделей вентильного, регистрового и системного уровней. Такое представление трудно воспринимается человеком, но оно технологично и легко понимается компьютером при автоматическом создании программных систем анализа вычислительных структур и сервисов киберпространства. Таким образом, регистровый граф цифровой схемы представляет собой гибкую систему адресно взаимосвязанных примитивов для формирования функциональной структуры любой сложности, прежде всего, в масштабах PLD, где все комбинационные элементы представлены постоянными запоминающими устройствами LUT [10], что обеспечивает высокое быстродействие и ремонт логических модулей в режиме онлайн.

Одномерный Q-вектор описания функциональности можно привязать к выходной (внутренней) линии устройства, состояние которой формируется в процессе моделирования рассматриваемого Q-покрытия. Тогда регистровая реализация комбинационного устройства (КУ) может быть представлена вектором моделирования M, невходные линии которого непосредственно связаны с выходами функциональных элементов. Упорядоченные значения входных переменных задают адрес бита Q-вектора, формирующего состояние рассматриваемой невходной линии (рис. 3, *в*). Если функциональности описываются одновыходовыми примитивами, то каждый из них можно отождествить с номером или координатой невходной линии, на которую нагружен данный элемент. Если функциональность многовыходовая, то Q-покрытие представляется матрицей с числом строк, равным числу выходов. Эффект от такого примитива заключается в параллелиз-

ме одновременного вычисления состояний нескольких выходов за одно обращение к матрице по текущему адресу.

Данное обстоятельство является существенным аргументом в пользу синтеза обобщенных кубитов для фрагментов ЦУ или всей схемы с целью их параллельной обработки в одном временном такте. Близкой к идеальной по компактности и времени обработки структурой данных, где Q-векторы функциональностей и номера входных переменных привязаны к невходным линиям ЦУ, является табл. 2, дающая представление о том, какие переменные цифровой схемы — внешние, сколько функциональных примитивов в структуре и какие входы нагружены на каждый Q-вектор. Достоинством табл. 2 является отсутствие вектора номеров выходов для каждого примитива при необходимости иметь номера входных переменных для формирования адресов, манипулирование которыми — достаточно времязатратный процесс. Модель анализа схемной структуры упрощается до вычисления двух адресов при формировании вектора $M_i = Q_i[M(X_i)]$ исключением сложного адреса выхода примитива в процессе записи состояний выходов в координаты M-вектора.

Кубитно-регистровый граф (см. рис. 3, *в*) представим в виде матрицы $\mu = |\mu_{ij}|$, $i = \overline{1, p}$; $j = \overline{1, q}$, параллельно-последовательной обработки логических примитивов:

μ_{ij}	1	2	3
1	1 / 2 — 1 | *A*	*A* / 7 — 7 | 8	8 / 6 — 14 | *B*
2	3 / 4 — 1 | 7	7 / 5 — 7 | 9	6 / 9 — 14 | *C*
3	*X* / *X* — 1 | *X*	*X* / *X* — 7 | *X*	*X* / *X* — 14 | *X*

Таблица 2

L	1	2	3	4	5	6	7	8	9	*A*	*B*	*C*
M	1	1	1	1	1	0	1	1	1	1	1	1
X	.	.	.	.	.	.	34	A7	75	12	86	96
Q	.	.	.	.	.	.	0	0	0	0	1	1
	.	.	.	.	.	.	0	1	1	0	1	1
	.	.	.	.	.	.	0	1	1	0	1	1
	.	.	.	.	.	.	1	1	1	1	0	0

Матричная модель отображает параллельно-последовательную структуру КС на основе взаимодействия *Q*-покрытий по уровням обработки логических элементов относительно формата (*X*—*Q*—*Y*), представляющих <входы — *Q*-вектор — выход> каждого примитива: [(1, 2—1—*A*), (3, 4—1—7)], [(*A*, 7—7—8), (7, 5—7—9)], [(8, 6—14—*B*), (6, 9—14—*C*)]. При этом адресный характер матричных примитивов дает возможность получить запасные элементы для восстановления работоспособности отказавших посредством переадресации (строка 3 матрицы) в режиме онлайн. Представленные в строке 3 матрицы $|\mu_{ij}|$ запасные примитивы не имеют номеров входных и выходных переменных, клетки которых отмечены символом *X*.

В процессе ремонта или перепрограммирования данные координаты доопределяются соответствующими номерами отказавшего элемента. Следует заметить, что не существует принципиальных ограничений для перепрограммирования логической функции запасного примитива в реальном времени при восстановлении работоспособности неисправного элемента. Поэтому все координаты запасных примитивов в строке 3 можно задекларировать пустыми для перепрограммирования структурности (номеров входов-выходов) и функциональности (*Q*-вектора) отказавшего элемента. Ограничением по инвариантности резервного примитива к замене любого неисправного может быть только равенство числа входов. В процессе моделирования, для корректной обработки функциональности примитива, необходимо к рассматриваемому моменту сформировать все его входные переменные. Поэтому кубитно-регистровый граф, а следовательно, и столбцы матрицы, разделены на уровни срабатывания, где все примитивы внутри одного уровня могут обрабатываться параллельно, а уровни — последовательно один за другим.

Регулярная структура кубитной матрицы ориентирована на решение следующих задач:

1. Ремонт логических примитивов в процессе функционирования посредством переадресации дефектных элементов на примитивы из запаса (строка 3) [16] подобно тому, как это делается в матричной памяти. Каждый тип логического примитива должен иметь, по крайней мере, один запасной компонент.

2. Индексная адресация каждого кванта матрицы $\mu_{ij} \in \mu$, $\mu_{ij} = (X_{ij}, Q_{ij}, Y_{ij})$ для оперативного ремонта отказавших примитивов в матрице $|\mu_{ij}|$ (можно заменить три дефектных примитива, по одному из каждого слоя).

3. Обеспечение высокого быстродействия прототипа КУ, реализованного на основе кубитных примитивов, имплементируемых на кристалле PLD в LUT-элементы [10], предоставлением возможности параллельной обработки элементов одного слоя.

4. Создание матричного кубитного мультипроцессора, ориентированного на анализ аппаратных прототипов КУ большой размерности, позволяющих существенно ускорить процессы тестирования и верификации ЦС на кристаллах [14].

5. Разработка методов анализа КС, ориентированных на матричное исполнение кубитных структур логических элементов, посредством их имплементации в элементы памяти кристаллов PLD.

6. Создание генератора кода для масштабируемого синтеза квантовых матриц КС на основе использования структур схемотехнических примитивов кристаллов PLD.

7. Проектирование управляющего автомата для функциональной обработки и сервисного обслуживания (восстановления работоспособности) кубитной матрицы КУ, имплементированного в PLD структуру.

Алгоритм управляющего автомата для моделирования кубитной матричной структуры КС состоит из трех шагов:

1. Инициирование очередного входного воздействия для КУ.

2. Выбор очередного слоя (столбца матрицы) с номером i для параллельной обработки кубитных примитивов Q с целью формирования состояний выходов по адресу входного слова, представленного вектором $M(X_{ij})$: $M(Y_{ij}) = Q_{ij}[M(X_{ij})]$, $j = \overline{1, q}$, где x_{ij} — последовательность номеров входных переменных для примитива Q_{ij}; M — вектор моделирования всех линий КУ; q — число столбцов.

3. Приращение индекса столбца $i = i + 1$ и переход к шагу 2 обработки очередного слоя кубитных примитивов. По окончании анализа всех столбцов матрицы $i = p$ выполняется инкремент индекса очередного входного воздействия $t = t + 1$ с последующим переходом к шагу 1. При достижении конечного числа входных наборов $t = n_{\max}$ цикл обработки теста для кубитной матрицы заканчивается.

Автоматная структура *MQT*-процессора. Основой процесс-модели функционирования вычислителя (КС), представленного в форме кубитной матрицы, является операция транзакции (считывания-записи) двоичной информации на LUT-элементах памяти структуры PLD $M(Y) = Q[M(X)]$, которая формирует сколь угодно сложные функциональности и сервисы. Все вычислительные процессы в компьютерных системах, сетях и киберпространстве можно свести к одной операции транзакции на любой структуре, способной хранить информацию. Все технологические и схемотехнические узлы, на которых реализуются аппаратные и программные продукты, можно не принимать во внимание при синтезировании виртуальных информационных сервисов, в которых следует использовать только операцию запись-считывание, как базовую процедуру над Q-по-

крытиями. Универсальность *Q*-покрытий позволяет описывать все конструктивы вычислительной техники для синтеза и анализа объектов, процессов и явлений в киберпространстве.

Предлагается *MQT*-модель кубитного ВК, основанная на транзакциях между компонентами памяти, реализующими логические функциональности с помощью *Q*-векторов, объединенных в систему с помощью вектора состояний переменных *M* (Memory). Моделирование логической функции примитива основано на считывании бита из *Q*-вектора $Q=(q_1, q_2, ..., q_i, ..., q_k)$, $k=2^n$, $q_i \in \{0,1\}$, и последующей записи данного бита в вектор *M*. Транзакция может существовать только при наличии заданных отношений на компонентах памяти, число которых должно быть не менее одного. Цикличность транзакции

$$M \xleftarrow{Y_i} Q_i \xleftarrow{X_i} M, \tag{1}$$

представленной на рис. 4, *а*, *б*, Read-Write операциями

$$M(Y_i)=Q_i\,[M(X_i)] \tag{2}$$

в *MQT*-структуре, определяется использованием вектора *M* взаимных связей примитивов, образующего гибкую, адресно ориентированную и гальванически разорванную автоматную модель, в которой главным компонентом ВС является кубит-память:

$$A=<M, Q, f, g, X, Y>, \tag{3}$$

$$M(t+1)=f\,[X(t), Q(t), M(t)], \tag{4}$$

$$Y(t)=g\,[X(t), Q(t), M(t)], \tag{5}$$

$$M(Y_i)=Q_i\,[M(X_i)]. \tag{6}$$

Выражения (3)—(6) соответствуют классическому определению автомата Мура, но равенство (6) задает основную и единственную процедуру анализа адресуемых *Q*-компонентов ЦС для формирования координат вектора состояния *M*. Последний представляет собой адресно вычисляемые реакции *Q*-примитивов устройства на входные воздействия, полученные конкатенацией координат вектора *M*, адреса которых соответствуют номерам входных переменных рассматриваемого примитива. Поскольку вектор выходных сигналов *Y* является подмножеством упорядоченных переменных *M*, он может быть исключен из рассмотрения. Адресные регистровые переменные Y_i, X_i каждого примитива дают возможность формировать состояние вектора *M* моделированием структуры квантовых примитивов *Q*.

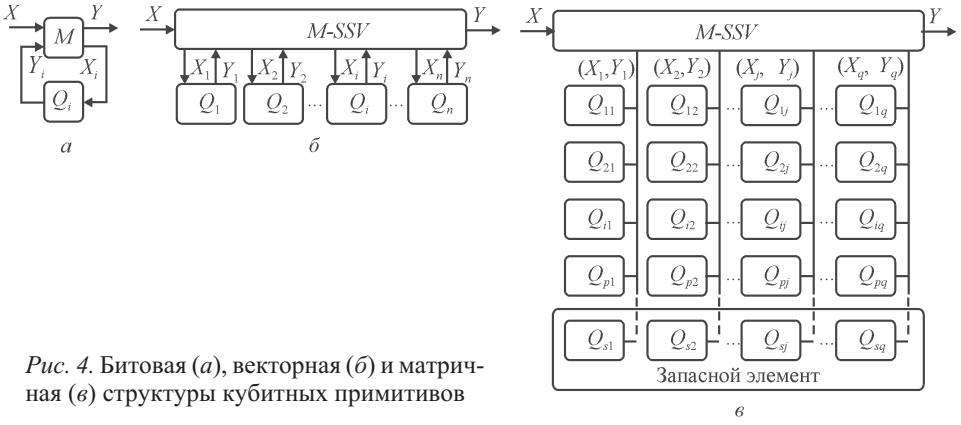


Рис. 4. Битовая (*а*), векторная (*б*) и матричная (*в*) структуры кубитных примитивов

Представленная на рис. 4 модель ЦУ в виде структурного *MQT*-автомата содержит вектор моделирования или состояния системы, который объединяет и гибко структурирует все кубитные примитивы для формирования функциональности, заданной спецификацией. При этом цикл обработки каждого *Q*-элемента заключается в транзакционном взаимодействии между собой пары компонентов *M*—*Q* (см. рис. 4, *а*), которые в совокупности реализуют универсальную функциональность равенства (2) с помощью двух транзакций (1).

Обработка всех кубитных элементов на входном воздействии *X* позволяет сформировать вектор *M* состояния ЦС SSV (System State Vector), который имеет также выходные переменные *Y* (см. рис. 4, *б*) для управления другими компонентами вычислительной структуры. Быстродействие анализа (запись-считывание) векторной (линейной) структуры из $n = p \times q$ примитивов имеет оценку: $\gamma = (R + W) \times n$, где *W* и *R* — циклы записи и считывания при обработке очередного элемента. Строго последовательный характер обработки упорядоченных по возрастанию номеров выходов квантовых примитивов можно усовершенствовать, сокращая время анализа схемы. Для этого необходимо построить двумерную структуру — матрицу *Q*-элементов, ранжированных по уровням параллельной обработки групп примитивов, оформленных в столбцы (см. рис. 4, *в*).

Быстродействие такой структуры по сравнению с линейной повышается в *q* раз: $\lambda^m = 1/q\,(R+W) \times \eta = (R+W) \times p$, $\eta = p \times q$, что соответствует времени обработки КС на основе жестко заданных гальванических связей. Но главное преимущество матрицы гибких связей компонентов — наличие нижней строки запасных примитивов для ремонта в режиме онлайн, что делает любой проект, содержащий комбинационную логику, тестопригодным, вследствие оперативной переадресации отказавшего *Q*-при-

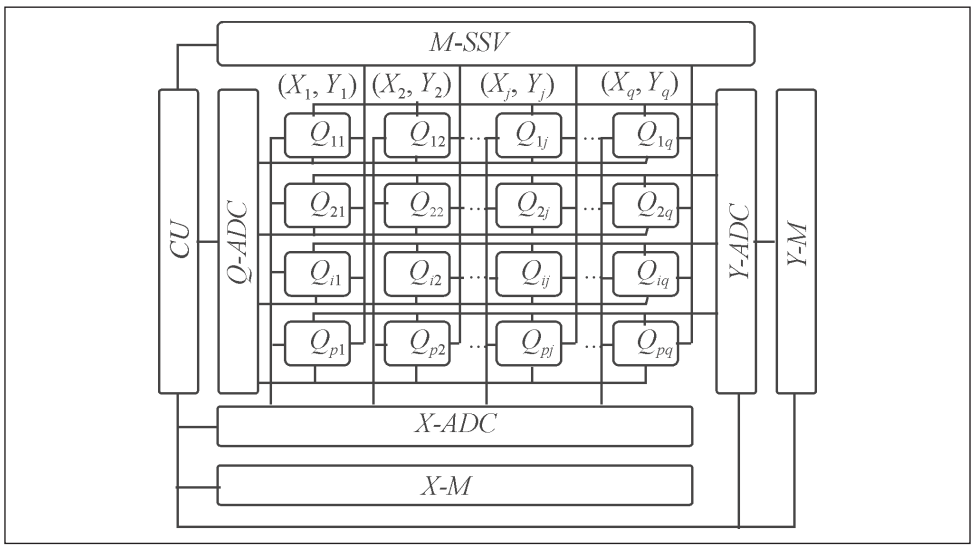


Рис. 5. Матричная структура операционного устройства: *CU* — устройство управления; *Q-ADC* — дешифратор адресов примитивов; *M-SSV* — вектор состояния ЦС; *X-ADC* и *Y-ADC* — дешифраторы адресов входов и выходов; *X-M* — память входов; *Y-M* — память выходов

митива на элемент из ремонтного запаса. Для этого при синтезе такой двумерной структуры желательно располагать в одном столбце однотипные примитивы, что уменьшит издержки на запасные элементы.

Инновационная идея матрицы кубитных примитивов реализует комбинационные *Q*-покрытия (векторы) на адресуемых: координатах кубита и элементах памяти, мягко соединенных в цифровую схему с помощью вектора состояний линий. Это дает возможность ремонтировать отказавшие логические примитивы в реальном времени с помощью их переадресации на запасные компоненты при достаточно высоком быстродействии функционирования вычислительного устройства (ВУ). Платой за такое преимущество является существенная аппаратная избыточность по сравнению с жесткой КС (рис. 5) .

Упрощение автомата Мура до пары транзакционно взаимодействующих компонентов на основе *M*-вектора моделирования, адресно структурирующего элементы, привело к тривиальной модели *MQT*-автомата: $A =< M, Q, X, Y >,\ M(Y_i) = Q_i\ [M(X_i)]$. Привязка идентификаторов функциональных *Q*-примитивов к номеру (адресу) выходной линии ЦС позволяет упростить *MQT*-автомат управления еще на одну транзакционную операцию, связанную с формированием адреса состояния выхода примитива *Y*: $A =< M, Q, X >, M_i = Q_i\ [M(X_i)]$. В этом случае отсутствует систе-

ма выходных переменных, которые заменены координатами *M*-вектора моделирования линий ЦУ.

Представленный *MQT*-автомат имеет следующие особенности:

1. Отсутствие жестких гальванических связей между элементами, формирующими комбинационную логику.

2. Наличие гибких связей *Q*-примитивов, формируемых с помощью *M*-вектора состояния ЦС.

3. Частичное уменьшение быстродействия вычислительной структуры в результате устранения гальванических связей.

4. Обеспечение новой возможности КС, которая заключается в восстановлении работоспособности системы в случае возникновения отказов в примитивных элементах.

5. Инвариантность реализации *MQT*-компьютеров по отношению к среде использования (виртуальное пространство, критические объекты) и субстанции имплементации (твердые кристаллы и материалы, жидкие, газообразные и плазменные формы существования материи).

6. Структура адресного взаимодействия *Q*-компонентов памяти между собой посредством *M*-вектора моделирования (сосредоточение входов и выходов ЦС) подобна технологии использования всех устройств компании Apple, когда входом и выходом любого гаджета является киберпространство (интернет или облака данной компании). Ни на макро- ни на микроуровне среды применения автоматов нет необходимости использовать внешние входы и выходы ЦС, которая ориентирована на жесткое взаимодействие с киберпространством и не представляет особого интереса как автономное ВУ. Вместо входов и выходов существует единое адресное пространство, в данном случае вектор *M*, откуда в ВС поступают входные данные, а после преобразований туда же записываются результаты или выходные данные: $A =< M, Q >$, $M_i = Q_i(M_j)$. Поэтому *MQT*-модель системного описания структур данных подобна отношениям между ВУ (*Q*-компоненты) и интернетом (*M*-память). По-видимому, это — настоящее и будущее технологической культуры, когда компьютер (гаджет) связывается с киберпространством, которое является входом и выходом для любого ВУ.

7. Для обработки кубитных вычислительных *MQT*-структур более низкого уровня действует простая и эффективная транзакционная процедура — кубит функциональности берет данные из адресного пространства *M*, а затем туда же записывает результат преобразования: $M_i = Q_i(M_j)$.

8. Предлагаемая *MQT*-модель вычислительной ячейки киберпространства включает структурный *MQ*-автомат адресной гибкой организации взаимодействия функциональных примитивов $MQ = \{M, Q, [(M \times Q) \to M] \vee \vee [M_Y = Q(M_X)]\}$ на основе использования *M*-вектора. *MT*-автомат осу-

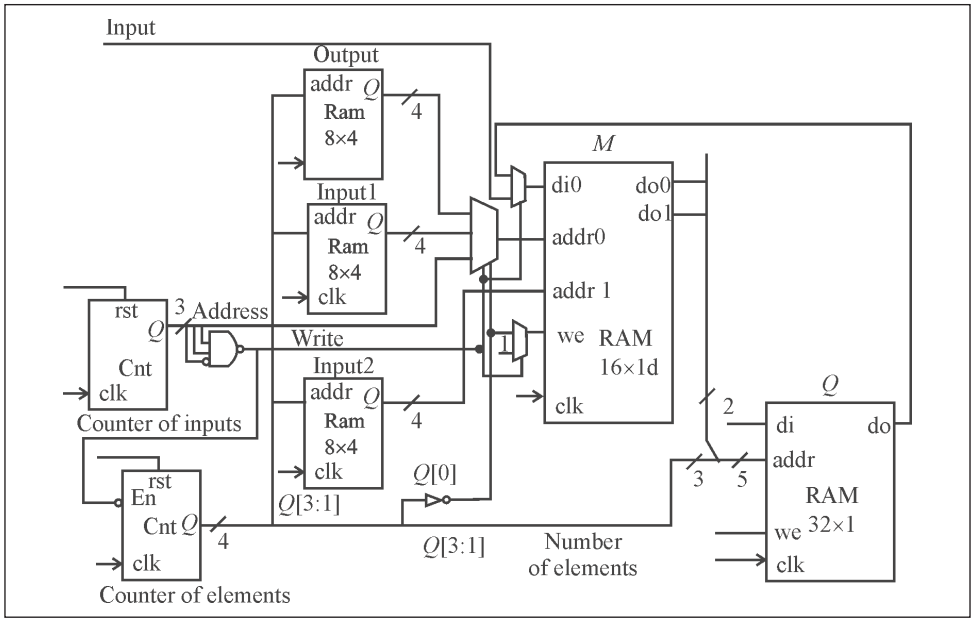


Рис. 6. Аппаратная реализация кубитной структуры КС: input — вход для последовательного занесения входных значений вектора *M*; rst — общий сброс системы, в данном случае счетчиков; clk — вход синхронизации; counter of inputs — счетчик заполнения входных координат вектора *M*; counter of element — счетчик номера обрабатываемого примитива, предоставляющий два такта для считывания входного набора из двух координат вектора *M*; *Q* [3:1] — шина номера обрабатываемого примитива; *Q* [0] — переменная режима считывания входных значений из вектора *M* или записи результата в *M*; Ram 8×4 output и Ram 8×4 input 1, Ram 8×4 input 2 — блоки памяти, сохраняющие номера выходных и входных линий примитивов; RAM 16×1d — двухпортовая память хранения вектора *M*, где addr0 — адрес входа 1 при значении 00 на входах управления мультиплексора, адрес записи результата при значении 01 на входах управления мультиплексора, адрес для инициализации входных данных при значении 1*X* на входах управления мультиплексора; addr1 — адрес входа 2 обрабатываемого примитива; di0 — вход данных памяти при обработке примитива или внешний вход input при инициализации входных данных; we — разрешение записи в вектор *M*; do0 и do1 — выходы, соответствующие входу addr0 и входу addr1; RAM 32×1 предназначен для хранения *Q*-векторов задания функциональностей КС: di — вход данных для инициализации структуры кубитов; addr — [4:0]: addr[4:2] — номер элемента; addr [1:0] — входной набор для примитива

ществляет управление обработкой кубитных примитивов, используя операцию транзакции, регламентируемую характеристическим выражением $M_i = Q_i(M_j)$, определяющим взаимосвязи адресуемых компонентов и процедуру их транзакционной обработки.

Таким образом, предложенный *MQT*-автомат, используя только элементы памяти и единственную операцию транзакции, дает возможность системно проектировать высоконадежные вычислительные и информа-

ционные сервисы как в реальном, так и в виртуальном пространстве. В качестве примера проанализируем цифровую схему на одном тестовом наборе. Вектор моделирования и кубитная структура схемы имеют следующий вид:

$$M = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & A & B & C \\ \hline 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

Q_{ij}	1	2	3
1	1 2 | 0001 | A	A 7 | 0111 | 8	8 6 | 1110 | B
2	3 4 | 0001 | 7	7 5 | 0111 | 9	6 9 | 1110 | C

Суть анализа цифровой структуры заключается в заполнении вектора моделирования по всем координатам:

1. Запись в элементы вектора M по шести координатам входных переменных двоичного набора 000111.

2. Обработка примитива с номером Q_{11}, имеющим входные переменные 1 и 2, а выход, обозначенный символом A. Для этого формируется адрес 00 путем конкатенации содержимого ячеек 1 и 2 вектора M. Применяя данный адрес к Q-вектору 0111, определяем содержимое нулевой ячейки, равное нулю, которое записывается в вектор M по адресу A выходной переменной обрабатываемого кубита.

3. Повторение процедуры, описанной в п. 2, ко всем Q-примитивам позволяет представить двоичными сигналами все координаты вектора M: 000111001010.

Аппаратная имплементация приведенного и масштабируемого примера кубитной структуры ЦУ с использованием элементов памяти представлена на рис. 6.

Аппаратная реализация примера КС составляет 150 вентилей, включающих 20 LUT системы элементов компании Xilinx Spartan 3E. Быстродействие функционирования или формирования вектора двоичного моделирования равно 180 ns.

Выводы

1. Предложенные кубитные модели описания ЦС и компонентов характеризуются компактностью описания таблиц истинности в форме Q-покрытий вследствие унитарного кодирования входных состояний, что позволяет повысить быстродействие программных и аппаратных средств

интерпретативного моделирования ВУ в результате адресной реализации анализа логических примитивов.

2. Представленная матричная модель кубитных примитивов для реализации КС обеспечивает адресное объединение *Q*-покрытий на элементах памяти, гибко соединенных в цифровую схему с помощью вектора состояний линий, что дает возможность ремонтировать отказавшие логические примитивы в реальном времени с помощью их переадресации на запасные компоненты при достаточно высоком быстродействии функционирования ВУ.

3. Введенная гибкая автоматная *MQT*-модель компьютера с использованием только адресуемых структур памяти и операции транзакции для программной и аппаратной реализации комбинационных и последовательностных функциональностей позволяет создавать быстродействующие и надежные ВС для проектирования сервисов киберпространства на основе параллельных логических операций и ремонта неисправных адресуемых функциональных примитивов.

4. Инновационная идея квантовых вычислений с переходом от вычислительных процедур над байт-операндом, определяющим в дискретном пространстве одно решение (точку) к логическим регистровым параллельным процессам над кубит-операндом, одновременно формирующим булеан решений, дает возможность определить новые перспективы на пути создания высокопроизводительных компьютеров параллельного анализа и синтеза структур и сервисов дискретного киберпространства.

5. Представленная концепция системного проектирования программных и аппаратных облачных сервисов киберпространства на основе *MQT*-автомата с применением транзакций на кубитных адресно связанных компонентах ВС с помощью вектора моделирования дает возможность программировать логические функциональности в кубитных примитивах интерпретативной структуры ЦУ.

6. Рыночная привлекательность *MQT*-компьютера определяется примитивизмом его реализации в программном и аппаратном исполнении, высоким уровнем использования памяти в структуре современных ВС, достигающим 94% площади кристалла, и как следствие — уменьшением влияния комбинационной логики на быстродействие системы в целом, повышением надежности компьютеров с помощью онлайн- ремонта адресуемых элементов памяти, включая логические примитивы. Существующее неприятие *MQT*-компьютера рынком связано с лоббированием ВС на основе жесткой комбинационной логики в VLSI-проектах со стороны компаний, разрабатывающих процессоры на основе технологий reusable logic.

СПИСОК ЛИТЕРАТУРЫ

1. *Michael A. Nielsen, Isaac L. Chuang* Quantum Computation and Quantum Information. — Cambridge University Press. — 2010. — 676 p.
2. *Stig Stenholm, Kalle-Antti Suominen* Quantum approach to informatics. — John Wiley and Sons, Inc. — 2005. — 249 p.
3. *Whitney M. G.* Practical Fault Tolerance for Quantum Circuits. PhD dissertation. University of California, Berkeley. — 2009. — 229 p.
4. *Mikio Nfrfhara* Quantum Computing. An Overview. — Higashi-Osaka: Kinki University, 2010. — 53 p.
5. *Yervant Z.* Embedded Memory Test and Repair: Infrastructure IP for SOC Yield // IEEE Conf. Publications. — Intern. Test Conf. 2002. — DOI: 10.1109/TEST.2002.1041777 . — P. 340—349.
6. *Горбатов В.А.* Основы дискретной математики. — М. : Высшая школа, 1986. — 311 с.
7. *Hahanov V.I., Litvinova E.I., Chumachenko S.V. et al.* Qubit Model for solving the coverage problem // IEEE Conf. Publications. East-West Design and Test Symposium. Kharkov, 14—17 September, 2012. — DOI: 10.1109/EWDTS.2013.6673167 . — P. 142—144.
8. *Хаханов В.И., Мурад Али Аббас, Литвинова Е.И. и др.*. Квантовые модели вычислительных процессов // Радиоэлектроника и информатика. — 2011. — № 3. — С. 35—40.
9. *Курош А.Г.* Курс высшей алгебры. — М. : Наука, 1968. — 426 с.
10. *Хаханов В.И., Литвинова Е.И., Гузь О.А*. Проектирование и тестирование цифровых систем на кристаллах. — Харьков: ХНУРЭ, 2009. — 484 с.
11. *Хаханов В.И.* Техническая диагностика цифровых и микропроцессорных структур. — Киев: ИСИО, 1995. — 242 с.
12. *Roth J.P.* Diagnosis of automata failures: a calculus and method // IBM Journal of Research and Development. — 1966. — № 7. — P. 18—32.
13. *Карибский В.В., Пархоменко П.П., Согомонян Е.С., Халчев В.Ф.* Основы технической диагностики. Кн. 1. — М. : «Энергия», 1976. — 346 с.
14. *Бондаренко М.Ф., Хаханов В.И., Литвинова Е.И.* Структура логического ассоциативного мультипроцессора// Автоматика и телемеханика. — 2012. — № 10. — С. 71—92.
15. *Борисовец Б.Э., Шаршунов С.Г.* Общая модель и синтез тестов для механизмов управления межрегистровым обменом данными в микропроцессорах // Там же. — 1992. — № 8. — С. 142—149.
16. *Koal T., Scheit D., Vierhaus H.T.* A comprehensive scheme for logic self repair // IEEE Conf. Publications. — Signal Processing Algorithms, Architectures, Arrangements and Applications. 2009. — E-ISBN : 978-83-62065-06-6. — P. 15—18.

V.I. Hahanov, Baghdadi Ammar Awni Abbas, E.I. Litvinova, A.S. Shkil

QUBIT DATA STRUCTURES FOR COMPUTER DEVICES

Qubit models and methods for improving the performance of software and hardware for analyzing digital devices by increasing the dimension of data structures and memory are developed. The main concepts, terminology and definitions are introduced for the implementation of quantum computation in practice and analysis of virtual computers. The results of research in the design and modeling computer systems in cyberspace by using a two-component automaton <memory, transaction> are presented.

K e y w o r d s: Qubit structure, quantum computing, automaton «memory-transaction», simulation of digital system, processor array.

REFERENCES

1. *Michael A. Nielsen, Isaac L. Chuang* Quantum Computation and Quantum Information. — Cambridge University Press, 2010. — 676 p.
2. *Stig Stenholm, Kalle-Antti Suominen* Quantum approach to informatics. — John Wiley and Sons, Inc., 2005. — 249 p.
3. *Whitney M. G*. Practical Fault Tolerance for Quantum Circuits : PhD dissertation. University of California, Berkley. — 2009. — 229 p.
4. *Mikio Nfrfhara* Quantum Computing. An Overview. — Higashi-Osaka: Kinki University, 2010. — 53 p.
5. *Yervant Z.* Embedded Memory Test and Repair: Infrastructure IP for SOC Yield // IEEE Conf. Publications. — Intern. Test Conf. 2002. — DOI: 10.1109/TEST.2002. 1041777. — P. 340—349.
6. *Gorbatov V.A.* Fundamentals of Discrete Mathematics. — Moscow: Higher School, 1986. — 311 p. (in Russian).
7. *Hahanov V.I., Litvinova E.I., Chumachenko S.V. et al*. Qubit Model for solving the coverage problem // IEEE Conf. Publications. East-West Design and Test Symposium. Kharkov, 14—17 September, 2012. — DOI: 10.1109/EWDTS.2013.6673167. — P. 142—144.
8. *Hahanov V.I., Murad Ali Abbas, Litvinova E.I. et al.* Quantum models of computer process // Radioelectronics and Informatics. — 2011.— No 3. — P. 35—40 (in Russian).
9. *Kurosh A.G.* Course of Higher Algebra. — Moscow: Nauka, 1968. — 426 p. (in Russian).
10. *Hahanov V.I., Litvinova E.I., Guz O.A*. Design and Test of Digital System-on-Chip. — Kharkov: KhNURE, 2009. — 484 p. (in Russian).
11. *Hahanov V.I.* Technical Diagnosis of Digital and Microprocessor Structures. — Kiev: ICIO, 1995. — 242 p. (in Russian).
12. *Roth J.P.* Diagnosis of automata failures: a calculus and method // IBM Journal of Research and Development. — 1966. — No 7. — P. 18—32.
13. *Karibsky V.V., Parkhomenko P.P., Sogomonyan E.S., Khalchev V.F.* Fundamentals of technical diagnostics. Book 1. — M.: «Energy», 1976. — 346 p. (in Russian).
14. *Bondarenko M.F., Hahanov V.I., Litvinova E.I.* Structure of Logical Associative Multiprocessor // Automation and Remote Control. — 2012. — No 10. — P. 71—92 (in Russian).
15. *Borisovets B.E. , Sharshunov S.G.* General Model and Test Synthesis for Control Engines of Interregister Data Exchange in Microprocessors // Ibid. — 1992. — No 8. — P. 142—149 (in Russian).
16. *Koal T., Scheit D., Vierhaus H.T.* A comprehensive scheme for logic self repair // IEEE Conf. Publications. — Signal Processing Algorithms, Architectures, Arrangements and Applications. — 2009. — E-ISBN : 978-8-362065-06-6. — P. 15—18.

Поступила 16.10.13;
после доработки 11.11.14

ХАХАНОВ Владимир Иванович, д-р техн. наук, декан факультета компьютерной инженерии и управления, профессор кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1978 г. окончил Харьковский ин-т радиоэлектроники. Область научных исследований — проектирование и тестирование вычислительных систем, сетей и программных продуктов.

БАГХДАДИ АММАР АВНИ АББАС, аспирант кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1978 г. окончил Багдадский университет. Область научных исследований — проектирование и тестирование вычислительных систем.

ЛИТВИНОВА Евгения Ивановна, д-р техн. наук, профессор кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1985 г. окончила Харьковский ин-т радиоэлектроники. Область научных исследований — проектирование и тестирование цифровых систем и сетей на кристаллах.

ШКИЛЬ Александр Сергеевич, канд. техн. наук, доцент, доцент кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1979 г окончил Харьковский ин-т радиоэлектроники. Область научных исследований — проектирование и тестирование цифровых систем и сетей на кристаллах.

Отзыв о статье В.И. Хаханова, Багхдади Аммар Авни Аббас, Е.И. Литвиновой, А.С. Шкиля «Кубитные структуры данных вычислительных устройств»

Более двух десятилетий муссируется в мире идея создания квантового компьютера. Серьезные фирмы и государственные фонды вкладывают инвестиции в лаборатории, занимающиеся построением физической модели квантового компьютера. Особую озабоченность выражают специалисты в области защиты данных, поскольку ученые обещают взломать с помощью квантового компьютера коды, которые считаются защищенными. В геометрической прогрессии увеличивается число публикаций, посвященных вопросам не только построения квантового компьютера, но и организации в нем вычислительного процесса. Например в [1] приведено более восьмидесяти источников, освещающих успехи в построении физической модели квантового компьютера.

Пока отсутствует в коммерческой эксплуатации квантовый компьютер, предлагают пользоваться его математической и алгоритмической моделями [2,3]. У многих ученых появляется мысль разработать теорию создания алгоритмов решения тех или иных задач, предназначенных для реализации в архитектуре квантового компьютера, с надеждой на его появление на рынке через некоторое время.

Авторы статьи «Кубитные структуры данных вычислительных устройств» также надеются, что в области верификации дискретных систем квантовый компьютер способен ускорить процесс поведенческого моделирования в не-

мыслимое число раз или что принципы, принятые в архитектуре квантового компьютера, могут стать основой новых эффективных методов проектирования и верификации дискретных систем. По крайней мере, они без ограничений используют соответствующую терминологию, например кубит, кубит-вектор, кубитная структура данных, квантовая матрица, квантово-регистровый граф, суперпозиция, перепутывание, реверсность отношений. Теория, представленная в статье, частично построена на предположении, что кубитная структура данных содержит в себе суперпозицию дискретных состояний, как будто это вектор кубитов некоторого квантового компьютера или его модели. Поэтому следует определить, в чем сущность квантового компьютера, какие особенности имеет его архитектура и каковы перспективы его практического применения.

Кубит (квантовый бит) и вектор кубитов являются конструктивными объектами квантовых алгоритмов. Квантовый вентиль — это операция, исполняемая над кубитами и векторами кубитов. Квантовый алгоритм представляется последовательностью квантовых вентилей. Алгоритм выполняется бесконечно долго, но может попадать в стабильное состояние, в котором считывается состояние квантового вектора как результат счета.

В конструкции квантового компьютера кубиты и их векторы исполнены как замкнутые системы на основе квантовых элементов той или иной природы, например групп атомов, охлажденных почти до абсолютного нуля. Квантовый вентиль реализуется как некоторое окружение вектора кубитов, влияющее на его состояние. Частота изменения состояния вектора кубитов определяется его конструкцией, физическими свойствами и может быть очень высокой. Собственно, это и определяет быстродействие квантового компьютера.

Поскольку вектор кубитов представляет собой замкнутую систему, по закону сохранения энергии состояние этого вектора может изменяться только линейными операторами-вентилями. Поэтому разрабатываются квантовые алгоритмы, с помощью которых вычисляются, в основном, линейные функции, например разложения чисел на сомножители, а реализация нелинейных логических алгоритмов связана с серьезными техническими трудностями.

В квантовых компьютерах осуществляются принцип суперпозиции и эффект запутанности, используемые в квантовой теории физики. Согласно фундаментальному принципу неопределенности состояние вектора кубитов невозможно измерить точно, оно становится случайным после каждого измерения. Поэтому для простоты суждений считается, что оно представляет собой суперпозицию многих состояний, каждое из которых имеет свою долю вероятности в измеренном результате. Этот принцип традиционно объясняется притчей о коте Шредингера, который, согласно суперпозиции, оказывается и мертвым, и живым одновременно.

Проявление этого принципа в квантовых компьютерах часто приводит к заблуждению, что такой компьютер параллельно решает множество задач. Но

суть состоит как раз в противоположном эффекте: принцип суперпозиции для получения достоверного результата вынуждает многократно повторять алгоритм и(или) выполнять его копии на параллельных квантовых процессорах.

Влияние внешних факторов не позволяет векторам кубитов долго (порядка миллисекунд) находиться в состоянии когерентности, когда проявляются их квантовые свойства. Это является физическим ограничением на сложность реализуемых алгоритмов. Эффект запутанности проявляется в том, что если поток квантовых частиц расщепить надвое, то в каждом из двух выходных потоков частицы будут иметь противоположные спины. На этом эффекте строят системы защиты передачи данных, в которых вмешательство в один из потоков может быть зарегистрировано посредством измерений квантовых частиц в другом потоке. Этот эффект используется также для передачи информации (телепортации квантов) внутри квантового компьютера.

Исследуются также архитектуры квантовой матрицы вентилей, адиабатического квантового компьютера и квантовой машины Тьюринга. Первая из них — это параллельная форма квантового компьютера, в которой квантовые вентили могут быть исполнены в обратном направлении, т.е. реверсно. Вторая представляет собой модель, которая на микроуровне физически реализует алгоритм симулируемого отжига. При этом получается достоверный результат, поскольку ход алгоритма заканчивается в наиболее вероятном конечном состоянии [4]. Третья архитектура — это вариант машины Тьюринга с вероятностными переходами. Считается, что все перечисленные архитектуры эквивалентны [5].

Таким образом, можно сделать следующие выводы о применимости архитектуры квантового компьютера.

Квантовый компьютер — это вероятностный процессор. Ни он, ни его модели не предназначены для решения точных, особенно логических задач, каковыми являются задачи поведенческого моделирования дискретных систем.

Квантовый компьютер предназначен для исполнения одного или ограниченного множества квантовых алгоритмов, поскольку алгоритм в нем задается в конструкции вектора кубитов и его окружения. Динамическое изменение программы такого компьютера – сложная инженерная задача, которая будет решена нескоро.

Быстродействие квантового компьютера не связано с принципом суперпозиции и определяется его конструкцией, скоростью изменения состояний кубитовых векторов, которая ограничена законами и константами квантовой физики.

Квантовый компьютер может эффективно решать задачи стохастического моделирования, подобные методу Монте-Карло и методу моделируемого отжига. Такой компьютер — это по своей сути эффективная аналоговая вычислительная машина для моделирования молекул и химических реакций.

Квантовые компьютеры еще не скоро покинут стены лабораторий. Сегодня на экспериментальном квантовом компьютере эффективно решаются лишь оптимизационные задачи моделируемого отжига. Квантовый компьютер и квантовые алгоритмы успешно моделируются на современных цифровых компьютерах в замедленном масштабе времени с целью их исследования. Перевод алгоритмов, предназначенных для неймановских процессоров, в квантовые алгоритмы возможен, но не имеет практического смысла.

Из всего изложенного можно сделать следующий вывод. Квантовые компьютеры по своей архитектуре больше похожи на аналоговые вычислительные машины, чем на цифровые. Они выполняют стохастические, недетерминированные алгоритмы. Поэтому адаптация под их архитектуру алгоритмов, предназначенных для неймановской архитектуры или реализуемых в конечных автоматах, скорее всего, обречена на неудачу.

СПИСОК ЛИТЕРАТУРЫ

1. *Monroe C. et al*. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects // Physical Review. —2014. — A 89. — P. 022317 (16)
2.. *Selinger P.* Towards a Quantum Programming Language // Math. Struct. in Comp. Science. — 2004. — Vol. 14, № 4. — P. 527—586.
3. *Gay S.J.* Quantum Programming Languages. Survey and Bibliography // Ibid. — 2006. — Vol. 16, № 4. — P. 581—600.
4. *Boixo S., Ronnow T., Isakov S. et al*. Evidence of Quantum Annealing with More Than One Hundred Qubits // Nature Physics. — 2014. — Vol. 10. — P. 218—224.
5. *Perdrix S., Jorrand P.* Classically-controlled Quantum Computation // Electronic Notes in Theoretical Computer Science. — 2006. — Vol. 135. — P. 119—128.

Ст. науч. сотр. кафедры вычислительной техники
Национального технического университета
«Киевский политехнический ин-т», д-р техн. наук
А.М. СЕРГИЕНКО