УДК 519.613:681.326

TAMER BANI AMER, И.В. ХАХАНОВ, Е.И. ЛИТВИНОВА, И.В. ЕМЕЛЬЯНОВ

КУБИТНЫЕ МОДЕЛИ ОПИСАНИЯ ЦИФРОВЫХ УСТРОЙСТВ

Предлагается облачный сервис QuaSim для моделирования и верификации цифровых систем, основанный на транзакциях между адресуемыми компонентами памяти для реализации любой функциональности. Описывается новый подход к синтезу и анализу цифровых систем, использующий векторную форму (квант) задания комбинационных и последовательностных структур для их имплементации в элементы памяти, что существенно отличается от общепринятой теории проектирования дискретных устройств на основе таблиц истинности компонентов. Используются квантовые или кубитные структуры данных [1-5] для реализации вычислительных процессов в целях повышения быстродействия анализа цифровых систем и уменьшения объемов памяти на основе унарного кодирования состояний входных, внутренних и выходных переменных и имплементации кубитных векторов в элементы памяти FPGA, реализующих комбинационные и последовательностные примитивы.

1. Общая характеристика исследования

Цель – повышение надежности вычислительных устройств за счет адресуемости схемных элементов, что позволяет выполнять online ремонт, а также повышает быстродействия методов моделирования, тестирования и верификации сложных цифровых систем, благодаря уменьшению размерности моделей функциональных примитивов и адресной реализации всех компонентов структур данных.

Задачи: 1) Создание автоматной модели кубитного процессора. 2) Синтез кубитных моделей цифровых примитивов. 3) Синтез и анализ кубитных моделей цифровых схем. 4) Анализ цифровых систем на основе использования квантовых векторов описания примитивов.

Актуальность исследования: 1) Современная система на кристалле содержит 94% памяти и лишь 6% логики, которая доставляет более 90% проблем, связанных с верификацией, тестированием, диагностированием и восстановлением работоспособности [6-7]. Конечно, быстродействие логических схем на порядок выше, чем у памяти, однако большая доля вычислительных процессов приходится на обмен информацией в структурах памяти. Поэтому преимущества комбинационной логики в реальных вычислительных системах обработки больших данных компенсируются большими временными затратами (порядка 90%), связанными с транзакциями в памяти. 2) Реализация процессора только на основе использования элементов памяти делает его однородным по структуре и типам функциональных примитивов, что доставляет очевидные технологические удобства процессам проектирования, производства и эксплуатации, включая верификацию, встроенные тестирование и диагностирование, а главное – ремонт в режиме online с помощью использования на кристалле универсальных адресуемых spare-компонентов памяти. 3) Моделирование в процессе верификации проектируемых вычислителей на основе адресных моделей компонентов делает данную процедуру технологически простой из-за регулярных структур данных и применения единственной операции транзакции на элементах памяти, а также более быстродействующей, благодаря параллельной квантоподобной обработке больших массивов однотипной памяти [3-5, 8, 11, 12]. 4) Энергопотребление при замене логики на элементы памяти возрастает на несколько процентов, что на самом деле будет платой за перечисленные выше существенные преимущества, связанные с увеличением выхода годной продукции, повышением надежности вычислительных изделий, снижением стоимости проектирования и изготовления, а также автономным восстановлением работоспособности в режиме remote & online без участия человека. Однако энергосберегающие решения по вычислительным процессам на памяти дают основания полагать, что такого проигрыша вообще не будет [9-10, 13-14].

2. Кубитные структуры данных

На мировом рынке электронных технологий наблюдается конкуренция между базами имплементации идеи [1-4,12]: 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 – в технологичности модификации проекта, недостатки – в невысоком быстродействии функционирования цифровой системы. 2) Жесткая реализация ориентирована на использование компилятивных моделей при разработке программных приложений или на имплементацию проекта в кристаллы VLSI [6-7,13-14]. 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, а также на возможность online ремонта в процессе эксплуатации.

Кубитные структуры описания цифровых систем. Кубит (n-кубит) есть векторная форма унитарного (унарного) 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) [3,12]. При этом допускается суперпозиция (одновременное 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.

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. Синонимами кубита при задании двоичного вектора описания логической функции являются: Q-покрытие, Q-вектор, квантовый вектор [3-4,12,15] как унифицированная векторная форма суперпозиционного задания выходных состояний, соответствующих адресным кодам входных переменных логического элемента.

Кубит в цифровой системе используется в качестве формы задания структурного примитива, инвариантной к технологиям реализации функциональности (hardware, software). Более того, синтез цифровых систем на основе кубитных структур не привязан жестко к теореме Поста, определяющей пять условий (классов) существования функционально полного базиса. На предлагаемом уровне абстракции n-кубит дает более широкие возможности для векторного задания любой n-входовой функции из булеана мощностью $|B(A)| = 2^{2^n}$, которое непременно содержит все функциональности, удовлетворяющие пяти классам теоремы Поста. 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 $k = 2^n$.

3. Синтез кубитного покрытия комбинационной схемы

Кубит комбинационной схемы представляет собой вектор состояний выхода на упорядоченном множестве всех входных слов, который отождествляется с адресами ячеек памяти вектора. Синтез Q-покрытия схемной структуры (без таблиц истинности логических элементов) на основе примитивов, заданных Q-векторами, сводится к получению обобщенного кубит-вектора путем выполнения логической операции над разрядами кубитных векторов с помощью декартовой процедуры – для двух 4-разрядных кубитов по суперпозиции логической операцией or (and, xor):

$\vee,\wedge,\oplus$	b(0)	b(1)	b(2)	b(3)
a(0)	$c(0)=a(0)\vee b(0)$	$c(1)=a(0)\vee b(1)$	$c(2)=a(0)\vee b(2)$	$c(3)=a(0)\vee b(3)$
a(1)	$c(4)=a(1)\vee b(0)$	$c(5)=a(1)\vee b(1)$	$c(6)=a(1)\vee b(2)$	$c(7)=a(1)\vee b(3)$
a(2)	$c(8)=a(2)\vee b(0)$	$c(9)=a(2)\vee b(1)$	$c(10)=a(2)\vee b(2)$	$c(11)=a(2)\vee b(3)$
a(3)	$c(12)=a(3)\vee b(0)$	$c(13)=a(3)\vee b(1)$	$c(14)=a(3)\vee b(2)$	$c(15)=a(3)\vee b(3)$

Например, для логических суперпозиций двух кубитов при получении Q-векторов схемных структур $c_1=(a_1\wedge a_2)\vee(b_1\wedge b_2)$, $c_2=(a_1\wedge a_2)\wedge(b_1\vee b_2)$, $c_3=(a_1\wedge a_2)\oplus(b_1\vee b_2)$, имеет место таблица:

a(and) =	0001
b(or) =	0111
$c_1=a(and)\vee b(or)$	0111 0111 0111 1111
$c_2=a(and)\wedge b(or)$	0000 0000 0000 0111
$c_3=a(and)\oplus b(or)$	0111 0111 0111 1000

При построении Q-покрытия трех схем из трех элементов каждая используется суперпозиция двух логических примитивов с третьим элементом (or, and, xor), вследсвтие чего получаются три вектора размерности 16 бит каждый. Вычислительная сложность процедуры синтеза Q-покрытия комбинационной схемы определяется произведением длин Q-векторов p примитивов, входящих в нее: $\eta=\prod_{i=1}^{p}\mathrm{card}(Q_i)$.

Проблема синтеза Q-покрытия схемы, входные линии/сходящиеся разветвления которой имеют гальванические/проводные соединения (здесь по переменной a_2): $c=(a_1\wedge a_2)\vee(a_2\vee a_3)$, является более сложной задачей. В данном случае после синтеза Q-покрытия схемы следует выполнить его верификацию относительно существования противоречивых адресов на переменных a_2 в целях минимизации Q-вектора путем последующего исключения упомянутых адресов из рассмотрения. При этом размерность Q-покрытия уменьшается до $\mathrm{card}(Q)=2^q$ координат, где q – общее число входных переменных схемы:

Q =	0111 0111 0111 1111
a_1 =	0000 0000 1111 1111
a_2 =	0000 1111 0000 1111
a_2 =	0011 0011 0011 0011
a_3 =	0101 0101 0101 0101

=

Q =	0111 0111 0111 1111
a_1 =	0000 0000 1111 1111
a_2 =	00xx xx11 00xx xx11
a_2 =	00xx xx11 00xx xx11
a_3 =	0101 0101 0101 0101

=

Q =	0111 0111
a_1 =	0000 1111
a_2 =	0011 0011
a_2 =	0011 0011
a_3 =	0101 0101

=

Q =	0111 0111
a_1 =	0000 1111
a_2 =	0011 0011
a_3 =	0101 0101

Синтез Q-покрытия включает: 1) построение таблицы соответствия адресов разрядам Q-вектора схемы, 2) отметку символами x противоречивых координат по двум строкам a_2, 3) исключение из таблиц всех столбцов с данными символами, 4) объединение в одну получившихся двух идентичных строк a_2, 5) результирующий Q-вектор комбинационной схемы имеет существенно меньшую размерность. Преимущества предложенного Q-метода синтеза вычислительных устройств заключаются в компактности их описания Q-векторами и в высоком быстродействии адресного моделирования логических элементов, создаются условия для рыночно привлекательной «квантовой» теории проектирования цифровых систем на кристаллах, использующей векторно-кубитную форму задания структурных компонентов.

4. Минимизация кубитного покрытия схемы

Синтез кубит-вектора схемы по Q-покрытиям компонентов связан с уменьшением размерности Q-вектора путем исключения несущественных переменных. Существенность зависит от гальванических соединений входных и внутренних линий цифрового устройства, которые накладывают ограничения, связанные с противоречивостью сигналов на линиях связи. Правило минимизации адресного пространства заключается в устранении адресных кодов, которые создают противоречия по соединенным переменным.

Пусть имеется Q-вектор схемы и его адресное пространство, где переменные b,c,d (a,b,c) соединены гальванически. Ниже приведены таблицы минимизации адресного пространства для получения уменьшенного Q-вектора:

Q =	0111 0111 0111
a =	0000 0000 1111
b =	0000 1111 0000
c =	0011 0011 0011
d =	0101 0101 0101

Q =	0xxx xxx1 0xxx xxx1
a =	0000 0000 1111 1111
b =	0xxx xxx1 0xxx xxx1
c =	0xxx xxx1 0xxx xxx1
d =	0xxx xxx1 0xxx xxx1

=

Q
a =
b =

Q =	0111 0111 0111 1111
a =	00xx xxxx xxxx xx11
b =	00xx xxxx xxxx xx11
c =	00xx xxxx xxxx xx11
d =	0101 0101 0101 0101

=

Q	0111
a =	0011
d =	0101

В таблицах наблюдается зеркальная осевая симметрия с инверсией сигналов на координатах адресного пространства, которая создает свойство, описываемое следующим выражением: $L \oplus R = 1 \to L_{ij} \oplus R_{ij} = 1$. Данное обстоятельство следует использовать для уменьшения размерности анализируемого пространства в два раза и соответствующего снижения вычислительной сложности задачи синтеза квантовой вектор-функциональности цифровой схемы.

Количество различных вариантов взаимодействий на q входных переменных, связанных с гальваническим соединением сочетаний входных линий, определяется функциональной зависимостью, граничные значения которой находятся в интервале: $\mathrm{card}(Q) = [2^q - 3^q]$.

Существует эффективная процедура для минимизации размерности Q-вектора путем выявления противоречий в кодах-столбцах, на координатах (Aij), соответствующих гальванически связанным w-переменным по j-параметру. Очевидно, такую процедуру достаточно выполнить на половине адресного пространства $\mathrm{card}(Q) = 2^q / 2$, тогда остальная часть противоречивых столбцов удаляется в соответствии с зеркальным отображением номеров тех столбцов, которые были удалены из первой половины таблицы кодов адресов:

$$\{Q_i, Q_{2^q - i}\} = \varnothing \leftrightarrow (\bigwedge_{j=1}^{w} A_{ij}) \oplus (\bigvee_{j=1}^{w} A_{ij}) = 1,\ i \le 2^q / 2.$$

Если в столбце A_i на группе из w связанных переменных зафиксировано, что конъюнкция их состояний равна нулю, а дизъюнкция имеет значение единицы, то i-столбец и его зеркальное отображение $2^q - i$ удаляются из адресного пространства A. Это автоматически приводит к исключению из Q-вектора двух полученных $\varnothing$-координат (в таблицах обозначены символами x), соответствующих данным столбцам.

Наблюдается также симметрия пространства векторов-расстояний по Хэммингу, полученных путем xor-взаимодействия между соседними строками таблицы адресного пространства, для которых суперпозиция левой и правой частей дает результат $L \oplus R = 0 \to L_{ij} \oplus R_{ij} = 0$:

Q =	0111	0111	0111	1111
a ⊕ b	0000	1111	1111	0000
b ⊕ c	0011	1100	0011	1100
c ⊕ d	0110	0110	0110	0110
d ⊕ a	0101	0101	1010	1010

$= (L, R);\ (L \oplus R) =$

Q =	0111	0111	0111	1111
a ⊕ b		0000	0000	
b ⊕ c		0000	0000	
c ⊕ d		0000	0000	
d ⊕ a		0000	0000	

$\leftrightarrow (L = \overline{R})$

Целесообразность минимизации логической функции, описанную квант-вектором: минимизация Q-векторов для получения нормальных или скобочных форм не имеет практического значения, существенно только уменьшение размерности вектора функционального описания, что может быть лишь следствием определения несущественности некоторых

входных (адресных) переменных. Тем не менее, существует проблема разбиения квант-вектора на составляющие части меньшей размерности, что связано с имплементацией функциональности в конструктивные компоненты LUT FPGA. В этом случае выполняется разбиение Q-вектора на два равных подвектора Q=(L,R), которые соединяются в структурно-адресную организацию функциональности с помощью мультиплексора $Q = (\bar{a} \wedge L) \vee (a \wedge R)$. Если переменная мультиплексирования a=0, то функциональность Q формируется с помощью ячеек левого L-вектора, в противном случае, когда a=1, значение функции Q формируется битами правого R-вектора. Алгоритмы разбиения и имплементации сложных логических функций имеются в каждой промышленной системе синтеза, моделирования и верификации компонентов SoC.

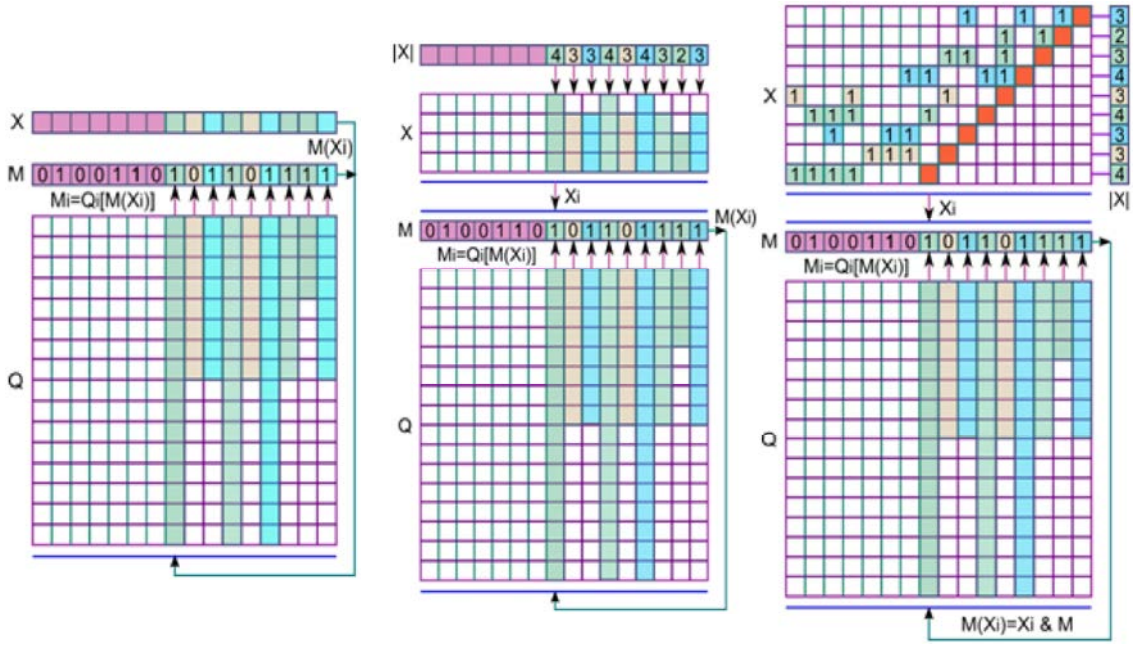
5. Модель кубитного процессора

Квантовый процессор может быть любой конечной размерности: вектор, матрица, куб. Для структуры, содержащей два измерения, он представлен матрицей столбцов или Q-векторов, которые формируют соответствующие им ячейки M-вектора моделирования (рис. 1, а). Вектор M совместно с X-вектором кортежей входных переменных примитивов создает структуру взаимных связей между столбцами-элементами. Адрес ячейки Q-покрытия, формирующей состояние невходного i-разряда M-вектора, определяется содержимым ячеек M-вектора, найденным по адресам, заданным i-кортежем вектора входных переменных. Каждый вектор Q_i, равно как и кортеж X_i вектора номеров входных линий, имеет адресную связь с M_i-ячейкой вектора моделирования. Квантовый процессор может входить компонентом в состав более сложной системы. Квантовая модель процессора имеет следующую структуру:

$$
\begin{aligned}
&W = <Q, M, X>,\\
&Q = (Q_1, Q_2, \ldots, Q_i, \ldots, Q_n),\\
&Q_i = (Q_{i1}, Q_{i2}, \ldots, Q_{ij}, \ldots, Q_{ik_i});\\
&M = (M_1, M_2, \ldots, M_i, \ldots, M_n);\\
&X = (X_1, X_2, \ldots, X_i, \ldots, X_n),\\
&X_i = (X_{i1}, X_{i2}, \ldots, X_{ij}, \ldots, X_{im_i});\\
&M_i = Q_i[M(X_i)];\ k_i = 2^{m_i}.
\end{aligned}
$$

В аналитической модели W представлены: 1) Упорядоченная адресно-доступная Q-совокупность квантовых примитивов, формирующих функциональность системы. 2) Вектор моделирования M, связывающий все примитивы в единую систему на основе идентификации эквипотенциальных линий, создающих формат из существенных переменных: входных, внутренних и выходных. 3) Вектор X кортежей упорядоченных номеров входных переменных для каждого квантового примитива, которые формируют адреса доступа к ячейкам Q-векторов примитивов (рис. 1, а). Вектор количества входных переменных примитива $|X|$ формирует адресное пространство или длину каждого Q-покрытия. Его можно представить в виде таблицы кортежей входных переменных, которые формируют номера линий вектора моделирования для вычисления адресов доступа к квантовым покрытиям (рис. 1, б). Таблицу кортежей можно также представить в виде матрицы масок входов, определенных в формате вектора моделирования, для параллельного формирования адресов и одновременного считывания выходных состояний примитивов из матрицы Q-покрытий (рис. 1, в). Из структуры X-матрицы входных линий видно, что кванты, формирующие выходы: (8, 9, 10), (11, 12) и (13, 14), можно обрабатывать параллельно. 4) Характеристическое уравнение, задающее алгоритм функционирования квантового процессора на основе использования только операций транзакции (считывание-запись) между Q-векторами примитивов и вектором моделирования.

Схема цифрового устройства, соответствующая приведенному выше описанию структур данных: M-вектор моделирования, X-матрица входов и Q-матрица покрытий, представлена на рис. 2. Она содержит 9 примитивов, каждый из которых имеет Q-покрытие в форме квант-вектора, реализующего некоторую функциональность. Особенность квантовых структур данных, представляющих модель цифровой схемы, заключается в полной адресуемости всех компонентов устройства без гальванических проводных соединений.



а б в

Рис. 1. Кубитные структуры данных кубитного процессора

Для кубитного(only memory-based) процессора имеют место следующие аксиомы: 1) В квантовом процессоре нет ничего, кроме адресуемой памяти. 2) Вычислительный процесс представлен единственной универсальной транзакцией между адресуемыми компонентами памяти $M_i = Q_i[M(X_i)]$. 3) Транзакция есть универсальная процедура считывания-записи данных на непустом множестве адресуемых элементов памяти. 4) Все компоненты памяти являются online-repaired, благодаря их псевдогальванической адресной (address-connected) связности. 5) Комбинационные логические элементы (reusable logic), равно как и последовательностные (sequential components), исполняются на элементах памяти. 6) Связывание всех компонентов в вычислительную систему осуществляется посредством (цифровой) идентификации псевдо-гальванических соединений вход-выходных переменных компонентов схемы, формирующих вектор моделирования, который хранит состояния всех существенных линий цифровой системы. 7) Все компоненты кубитной модели цифровой системы: W=<Q,M,X>, включая функциональные модули, вектор моделирования, вектор адресов входных переменных, являются online перепрограммируемыми, а значит – online ремонтопригодными. 8) Примитив цифровой системы имеет формат W=<Q,Y,X>, поскольку отдельный элемент не имеет связей и вектора M, создающих из отдельных компонентов систему.

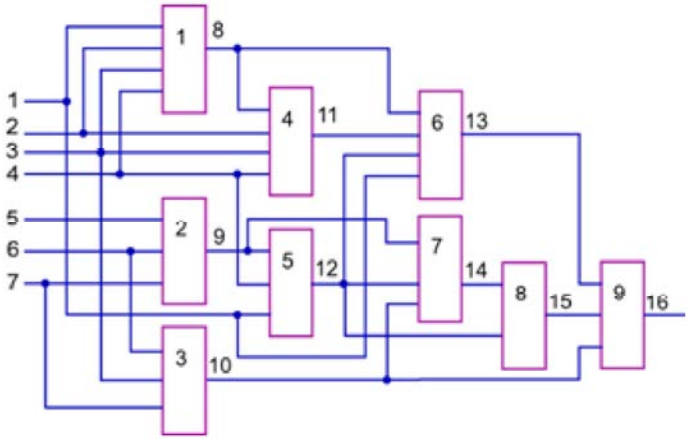


Рис. 2. Схема цифрового устройства

Согласно введенной квантовой модели, описания последовательностных примитивов (триггеры, регистры, счетчики) можно представлять Q-покрытиями или кубитными векторами, которые имеют псевдопеременные для задания внутреннего состояния. Например, функциональное описание SR-триггера трансформируется в квантовый примитив, заданный Q-покрытием, а затем реализуется на адресуемом элементе памяти FPGA с диаграммами проверки, что представлено на рис. 3.

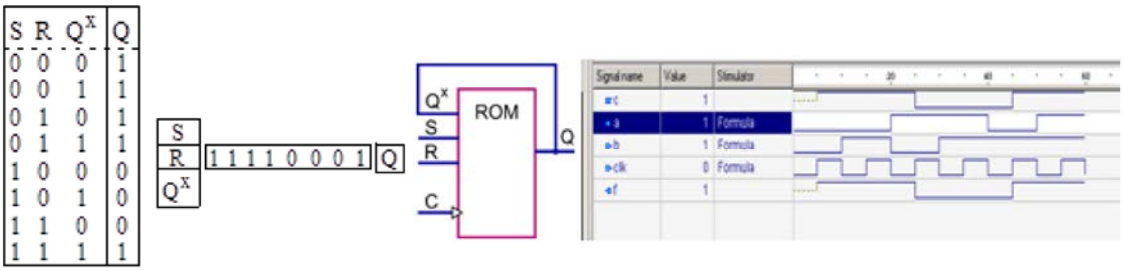


Рис. 3. SR-триггер на элементе памяти

Таблица истинности триггера представлена в форме вектора выходных состояний $Q(S,R,Q^X)=(11110001)$, который записывается в элемент постоянной памяти, имеющий три адресных входа, сигнал синхронизации, а также обратную связь, которая соединяет выход элемента памяти с одним адресным входом. HDL-реализация в системе проектирования Active HDL 9.1 (Aldec Inc.), а также результаты верификации синтезированного SR-триггера подтверждают корректность схемотехнического решения.

Другой пример связан с синтезом на элементе постоянной памяти синхронного DV-триггера. Таблица истинности триггера трансформирована в вектор выходных состояний $Q(D,V,Q^X)=(01000111)$, который записывается в элемент памяти, имеющий три адресных входа, сигнал синхронизации, а также обратную связь, которая соединяет выход примитива памяти с одним адресным входом. Все упомянутые компоненты, включая временные диаграммы верификации HDL-кода модели DV-триггера, представлены на рис. 4.

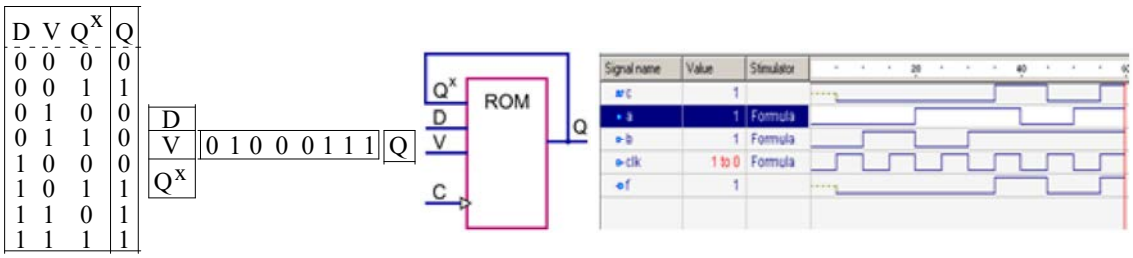


Рис. 4. DV-триггер на элементе памяти

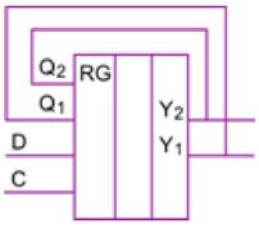
На рис. 5 представлены модели двух последовательностных примитивов: двухразрядных регистра и счетчика. Их отличие заключается в задании двух выходов, состояния которых формируются одним и тем же множеством входных переменных.

Регистр на переменных (C,D,Q1,Q2,Y1,Y2) выполняет функцию сдвига вправо информации от входа D по разрядам: (D-Y1-Y2), при R=1, и сохранение данных при C=0. Счетчик, определенный на переменных (R,C,Q1,Q2,Y1,Y2), реализует функцию инкремента по разрядам (Y1,Y2), при RC = (11), а также режим хранения информации, при (R or C = 0). Таким образом, для реализации двухразрядного регистра или счетчика необходимо два 16-битовых элемента пямяти, работающих синхронно от одних и тех же входов:

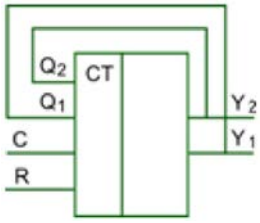
C		
D	0 0 1 1 0 0 1 1 0 0 0 0 1 1 1 1	Y1
Q1	0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1	Y2
Q2		

R		
C	0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0	Y1
Q1	0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0	Y2
Q2		

Здесь каждая квантовая модель представлена двумя векторами, где каждый из них формирует функцию разряда регистра или счетчика, как состояние ячейки вектора, получаемое при формировании адреса A входными переменными: {Y1, Y2} = A(C, D, Q1, Q2), {Y1, Y2} = A(R, C, Q1, Q2) соответственно. Моделирование примитива сводится к тривиальной процедуре формирования адреса, по которому находится состояние выхода примитива, как содержимое ячейки квантового вектора.



C	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
D	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Q1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Q2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Y1	0	0	1	1	0	0	1	1	0	0	0	0	1	1	1	1
Y2	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1



R	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
C	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Q1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Q2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Y1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
Y2	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0

Рис. 5. Memory-based модели регистра и счетчика

6. Алгоритм моделирования кубитных покрытий цифровых компонентов

Использует memory-based only модели для адресного анализа цифровых систем в целях их верификации. 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. Программная реализация алгоритма моделирования таких структур становится конкурентоспособной по быстродействию на рынке проектирования цифровых систем на кристаллах за счет адресации функциональных примитивов.

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 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. 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 параллелизме 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 цифрового устройства или всей схемы в целях их параллельной обработки на одном временном такте. Модель функционирования цифровой структуры упрощается до вычисления двух адресов при формировании вектора моделирования $M_i = Q_i[M(X_i)]$ путем исключением сложного адреса выхода примитива в процессе записи состояний выходов в координаты M-вектора.

Алгоритм моделирования квантовых примитивов цифровой системы использует аналитическую структуру (k – число входных переменных i-примитива, * – операция конкатенации битов, A – адрес бита Q-вектора):

$$M_i = Q_i(A) \leftarrow A = \overset{k}{\underset{j=1}{*}} M(X_{ij}) \leftarrow \begin{matrix} M \\ X_i \\ Q_i \end{matrix}$$

Данному аналитическому выражению можно поставить в соответствие следующие пункты алгоритма формирования двоичных состояний M-вектора моделирования цифровой схемы, изображенные на рис. 6:

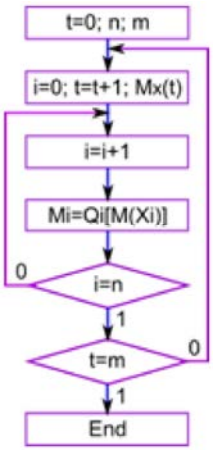
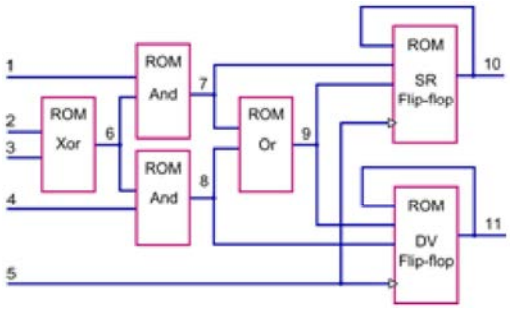


Рис. 6. Алгоритм моделирования квантовых покрытий цифровой системы

0) Инициирование начальных условий и параметров. 1) Задание очередного набора двоичных состояний на входных координатах вектора моделирования. 2) Определение i-номера очередного обрабатываемого примитива путем выполнения операции инкрементирования. 3) Выполнение процедуры конкатенации состояний битов M-вектора, соответствующих номерам вектора входных переменных Xi. Считывание соответствующего бита из функционального кубит-покрытия Qi по двоичному вектор-адресу сконкатенированных битов M-вектора. Занесение считанного из кубита бита в вектор моделирования M по адресу i. (M-вектор может иметь координаты с символами X, что дает возможность выполнять троичное моделирование цифровых устройств для решения задач тестирования и верификации.) 4) Если не все примитивы обработаны i<n, выполняется переход к пункту 2 алгоритма. 5) Если не все входные наборы обработаны t<m, выполняется переход к пункту 1. 6) Конец моделирования.

Исходя из характеристического уравнения квантовой модели цифровой системы можно сделать вывод, что современный <MQT> (Memory-Quant-Transaction) процессор следует представлять как адресную организацию структуры функциональных примитивов памяти без гальванических или проводных связей, на которых определены адресные транзакции данных во времени и пространстве для достижения поставленной цели.

На рис. 7 представлена схема с триггерами и комбинационной логикой, которая также описана в виде элементов памяти, куда занесены выходные состояния таблицы истинности каждого логического элемента. Структуры данных, необходимые для моделирования цифрового устройства, сведены в таблицу, где основными компонентами являются: M – вектор моделирования или состояния занумерованных линий, который в данном случае имеет 5 входных, 6 внутренних и выходных линий, состояния которых подлежат определению; X – вектор кортежей номеров входных линий примитивов, которые необходимы для формирования адреса в целях извлечения по нему состояния выхода элемента Q_i, функциональность которого задается Q-вектором.



L	1	2	3	4	5	6	7	8	9	10	11
M	0	0	1	1	1	1	0	1	1	1	1
X	.	.	.	.	.	2	1	6	7	10	11
	.	.	.	.	.	3	6	4	8	7	9
	.	.	.	.	.	.	.	.	.	9	8
Q	.	.	.	.	.	0	0	0	0	1	0
	.	.	.	.	.	1	0	0	1	1	1
	.	.	.	.	.	1	0	0	1	1	0
	.	.	.	.	.	0	1	1	1	1	0
	.	.	.	.	.	.	.	.	.	0	0
	.	.	.	.	.	.	.	.	.	0	1
	.	.	.	.	.	.	.	.	.	0	1
	.	.	.	.	.	.	.	.	.	1	1

Рис. 7. Memory-based комбинационная схема с триггерами

Пример выполнения алгоритма моделирования схемной квантовой структуры. Все примитивы должны быть упорядочены по принципу: очередной элемент анализируется, если все предшественники для него были обработаны. В процессе моделирования адресно извлеченное состояние ячейки текущего Q-покрытия заносится в разряд Mi вектора моделирования. Результаты последовательной обработки всех Q-векторов схемной структуры формируют состояния линий M-вектора для приведенного выше примера ячейки (6 – 11). Первоначальные состояния неопределенностей на псевдовходах функциональных примитивов доопределяются сигналами нуля или единицы в зависимости от внутренней технологической культуры компании, предоставляющей промышленные средства моделирования и верификации. Количество входных переменных примитива q связано с длиной Q-вектора соотношением: $\mathrm{card}(Q) = 2^q$. Правильность работы алгоритма моделирования была верифицирована на тестовых и реальных схемах с привлечением средств Active HDL 9.1 (Aldec Inc.). Особенность структурно-функционального задания цифровой системы заключается в представлении всех примитивов элементами памяти, куда записываются Q-векторы выходных состояний.

Таким образом, можно сделать следующие выводы: 1) Любые структурные компоненты вычислительных устройств, комбинационные и/или последовательностные, а также системы в целом можно описывать кубитными Q-векторами и реализовывать в элементах памяти FPGA, CPLD или VLSI. Это предоставляет рынку электронных технологий возможность не использовать комбинационную reusable логику при синтезе вычислительных устройств, которая доставляет разработчикам серьезные проблемы, связанные с тестированием, верификацией и ремонтом жесткой проводной реализации цифровых изделий. 2) Memory-based интерпретативное адресно-ориентированное моделирование комбинационных и последовательностных примитивов цифровых устройств становится соизмеримым по быстродействию с компилятивным анализом дискретных объектов. Кроме того, становится возможным реализовывать на программируемых логических устройствах аппаратное моделирование цифровых систем, где комбинационные и последовательностные функциональные примитивы будут представлены стандартными элементами памяти, в которые зашиваются Q-векторы.

7. Анализ вычислительных структур

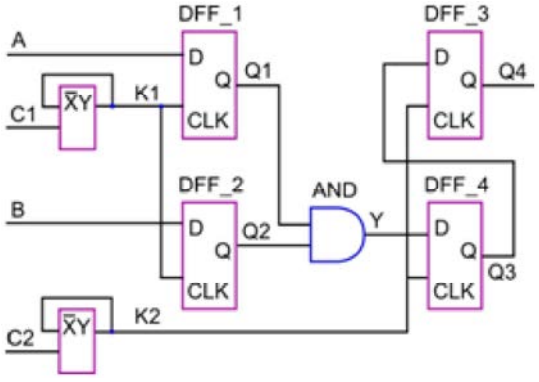
Сигналы синхронизации доставляют определенные неудобства для описания моделей последовательностных компонентов (триггеры, регистры, счетчики) и реализации алгоритмов анализа. Это связано со схемотехническим исполнением управления по переднему или заднему фронту, которые разрешают выполнение транзакций между master-slave компонентами. Другими словами, синхронные примитивы имеют два последовательно соединенных элемента, ориентированные на низкий и высокий уровни сигналов записи данных в первую и вторую ступени соответственно. Однако для логического моделирования учет подробностей, связанных со схемотехническими решениями, может существенно замедлить время анализа цифровых схем. Поэтому здесь необходимы логически адекватные модели реальных процессов, приводящие к повышению быстродействия алгоритмов обработки компонентов. При этом накладывается ограничение, связанное только с адресным

характером анализа всех компонентов схемы. Для обеспечения возможности рассмотрения синхровхода, как логической переменной, формирующей адрес квантового вектора, предлагается модель разбиения последовательностного примитива на два элемента: 1) Логический квант выдачи разрешающего сигнала при формировании переднего (заднего) фронта в двух временных модельных тактах. 2) Квант реализации штатной функциональности (триггера регистра, счетчика) последовательностного компонента. Таким образом, модель синхронного D-триггера может быть описана в форме двух Q-покрытий, адресно вычисляющих состояния выходов:

CLK(t 1)	0 1 0 0	C(t)
CLK(t)		

Q(t 1)		
D(t)	0 0 0 1 0 0 1 1	Q(t)
C(t)		

С учетом изложенного выше модель цифровой схемы с двумя сигналами синхронизации, представленная на рис. 8, будет иметь структуры данных, состоящие из совокупности Q-покрытий, которые формируют текущий вектор моделирования M, но с учетом значений координат данного вектора в предыдущий момент времени M(t-1). Увеличение числа переменных за счет введения двух элементов синхронизации уменьшает совокупную размерность таблицы квантовых векторов, которая при 7 переменных будет иметь 56 координат.



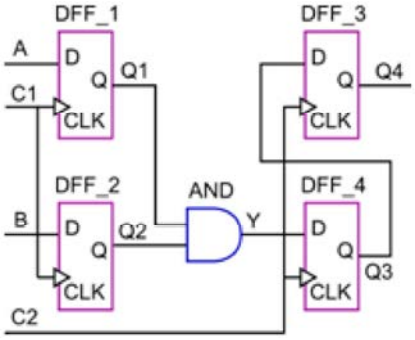
L	A	C_1	B	C_2	K_1	Q_1	Q_2	Y	K_2	Q_3	Q_4
$M^{t\ 1}$	1	0	1	0	0	0	0	0	0	0	0
M^t	1	1	1	1	1	1	1	1	1	1	1
X	.	.	.	.	$C_1^{t\ 1}$	$Q_1^{t\ 1}$	$Q_2^{t\ 1}$	Q_1	$C_2^{t\ 1}$	$Q_3^{t\ 1}$	$Q_4^{t\ 1}$
	.	.	.	.	C_1^t	A	B	Q_1	C_2^t	Y	Q_3
	.	.	.	.	.	K_1	K_1	.	.	K_2	K_2
Q	.	.	.	.	0	0	0	0	0	0	0
	.	.	.	.	1	0	0	0	1	0	0
	.	.	.	.	0	0	0	0	0	0	0
	.	.	.	.	0	1	1	1	1	1	1
	.	.	.	.	.	0	0	.	.	0	0
	.	.	.	.	.	0	0	.	.	0	0
	.	.	.	.	.	1	1	.	.	1	1
	.	.	.	.	.	1	1	.	.	1	1

Рис. 8. Синхронизированная структура триггера

Если не вводить две дополнительные переменные (элементы синхронизации), то объем памяти для Q-покрытий будет равен 80 ячейкам (рис. 9). Данная схемная реализация максимально ориентирована на структуры данных промышленных средств моделирования и верификации. Однако квантовые векторы для задания функциональностей триггеров создают необходимые условия для повышения быстродействия интерпретативного анализа, тестирования и диагностирования схемных компонентов.

Проблема уменьшения совокупного объема Q-покрытий схемы связана с количеством переменных, формирующих адреса координат Q-вектора. Естественно, что любое разбиение числа переменных на два равных упорядоченных подмножества дает возможность существенно уменьшить размерность памяти для записи уже двух Q-векторов. В общем случае функциональная зависимость уменьшения размерности исходного Q-вектора, определенного на n-переменных, при делении на 2 подсхемы с равным числом результирующих переменных (n/2), имеет следующий вид:

$$Q = \frac{2^n}{2^{n/2} + 2^{n/2}} = \frac{2^n}{2x2^{n/2}} = \frac{2^n}{2^{n/2+1}} = 2^{n-(n/2+1)} = 2^{n/2-1}.$$



L	A	C_1	B	C_2	Q_1	Q_2	Y	Q_3	Q_4
M^{t-1}	1	0	1	0	0	0	0	0	0
M^t	1	1	1	1	1	1	1	1	1
X	.	.	.	.	Q_1^{t-1}	Q_2^{t-1}	Q_1	Q_3^{t-1}	Q_4^{t-1}
	.	.	.	.	A	B	Q_2	Y	Q_3
	.	.	.	.	C_1^{t-1}	C_1^{t-1}	.	C_2^{t-1}	C_2^{t-1}
	.	.	.	.	C_1^t	C_1^t	.	C_2^t	C_2^t
Q	.	.	.	.	0	0	0	0	0
	.	.	.	.	0	0	0	0	0
	.	.	.	.	0	0	0	0	0
	.	.	.	.	0	0	1	0	0
	.	.	.	.	0	0	.	0	0
	.	.	.	.	1	1	.	1	1
	.	.	.	.	0	0	.	0	0
	.	.	.	.	0	0	.	0	0
	.	.	.	.	1	1	.	1	1
	.	.	.	.	0	0	.	0	0
	.	.	.	.	1	1	.	1	1
	.	.	.	.	1	1	.	1	1
	.	.	.	.	1	1	.	1	1
	.	.	.	.	1	1	.	1	1
	.	.	.	.	1	1	.	1	1
	.	.	.	.	1	1	.	1	1

Рис. 9. Схема с D-триггерами на основе внутренней синхронизации

Например, разбиение вектора из восьми переменных на два Q-покрытия из 4 переменных уменьшает объем памяти в 8 раз. Однако следует иметь ввиду, что каждое разбиение функционального модуля на k подсхем относительно внешних входов потребует k дополнительных d-циклов для вычисления состояния выхода всей схемы $T = d(k+1)$. Снижение быстродействия разбиенной функциональности является платой за существенное уменьшение объема памяти для хранения структур данных цифровой системы. В общем случае разбиение функциональности на k одинаковых частей приводит к получению следующей зависимости выигрыша объема памяти от числа разбиений на подмножества вектора входных переменных: $Q = \frac{2^n}{k \times 2^{n/k}} = \frac{1}{k} \times 2^{n-n/k}$.

Здесь параметр разбиения k принимает значения, кратные степени двойки: 2, 4, 8, 16. Однако значение k не должно быть более, чем n/2. Следует заметить, что необязательно количество разбиений не обязательно должно принимать значения, кратные степени двойки. В общем случае, на векторе входных переменных может существовать m разбиений, каждое из которых имеет более одной переменной. При этом выполняется условие разбиения, что сумма всех переменных, участвующих в разбиениях, не может быть больше n:

$$Q = \frac{2^n}{2^{k_1} + 2^{k_2} + \ldots + 2^{k_i} + \ldots + 2^{k_m}}, \; k_1 + k_2 + \ldots + k_i + \ldots + k_m = n.$$

Формула показывает выигрыш от разбиения функциональности в виде отношения размерности исходного квант-вектора к совокупному объему Q-векторов, полученных после разбиения. Чтобы оценить эффективность разбиения функциональности на схемные фрагменты, необходимо учитывать не только уменьшение объема памяти для хранения структур данных, но и негативные последствия, связанные со стоимостью анализа увеличенного

количества схемных компонентов, которое регламентируется в каждом конкретном случае коэффициентом d:

$$Q = \frac{2^n}{d \times m \times (2^{k_1} + 2^{k_2} + \ldots + 2^{k_i} + \ldots + 2^{k_m})}.$$

Подводя итог в части модификации теории исправного моделирования (fault-free) цифровых систем, можно отметить следующие факты. Автомат моделирования синхронных цифровых устройств, как правило, представлен моделью Мура:

$$S(t) = f[X(t), S(t-1)];$$
$$Y(t) = f[X(t), S(t)].$$

Здесь фигурируют входные (X) и внутренние состояния автомата в двух соседних временных фреймах S(t), S(t-1), а также правила определения выходных значений Y(t) для инициирования вычислительных процедур. Предлагается модификация упомянутой модели автомата Мура для анализа цифровых систем, суть которой заключается в замене функциональных отношений адресными (A) операциями:

$$S(t) = A[X(t), S(t-1)];$$
$$Y(t) = A[X(t), S(t)].$$

Определенный выше адресный или квантовый автомат позволяет: 1) Избежать жестких гальванических межсоединений между элементами комбинационных и последовательностных схем при их аппаратной имплементации только в элементы памяти. 2) Получить свойство гибкой заменяемости компонентов цифровой системы в режиме online, благодаря их адресуемости. 3) Существенно упростить все процессы моделирования, верификации и тестирования путем использования только процедур вычисления адреса компонента схемы или ячейки его памяти. 4) Унифицировать процессы проектирования цифровых изделий путем их сведения к формированию функциональностей на основе вычисления адресов или к транзакциям на элементах памяти. 5) Повысить эффективность процедур моделирования цифровых схем за счет уменьшения объема интерпретативных моделей и упрощения способа их обработки, когда вместо исчерпывающего анализа таблиц предлагается вычисление адреса ячейки квантового вектора. 6) Выполнять все вычислительные процедуры на основе использования квантовых покрытий и вектора моделирования, заданного в двух соседних автоматных тактах, согласно определению квантового автомата. 7) Технологически проще становится использовать инфраструктуру [6, 7] стандартов тестопригодного проектирования (IEEE 1500, 1149) для покомпонентного тестирования, диагностирования и восстановления работоспособности адресно доступных функциональных блоков в режиме online.

8. Структура облачного сервиса QuaSim для моделирования цифровых устройств

QuaSim представляет собой средство для анализа, тестирования и верификации цифровых проектов небольшой размерности и предназначено для использования в учебном процессе в качестве облачного сервиса, доступного для студентов с любого мобильного устройства или компьютера.

Цель – существенное повышение качества учебного процесса путем предоставления технологичных микросервисов анализа цифровых устройств с одновременной визуализацией схем, тестов, результатов моделирования и таблиц истинности функциональных элементов.

Задачи: 1) Создание структуры облачного моделирования цифровых устройств на платформе компьютинговых сервисов Google. 2) Разработка модуля (микросервиса) Q-element, реализующего создание модели и визуализацию логического или функционального примитива схемы. 3) Проектирование модуля генерирования кубитных моделей примитивов и более сложных цифровых устройств, а также средств их оперативной визуализации. 4) Разработка модуля или панели управления, интегрирующего симулятор, генератор логических элементов, схемных конструкций и вход-выходных портов. 5) Разработка собственно модуля для анализа цифровых схем на основе рекурсивной обработки логических

элементов. 6) Создание служебных библиотек для хранения: готовых функциональных элементов, сложных цифровых схем, тестов и результатов их анализа, а также служебной информации. 7) Тестирование и верификация облачного сервиса Q-simulator, предназначенного для интерпретативного моделирования цифровых устройств.

Сущность квантового метода анализа заключается в адресной реализации всех функциональных компонентов цифровых систем и структур данных, что дает возможность существенно повысить быстродействие интерпретативного моделирования и качество обслуживания проектов за счет быстрой замены некондиционных логических элементов путем их переадресации.

Структура облачного сервиса включает следующие основные микросервисы: 1) Q-element генерирует квантовые описания логических элементов в структуре цифровой функциональности. 2) Модуль View выполняет визуализацию схемных элементов, портов входов и выходов на экране монитора. 3) Модуль Collapse управляет окнами монитора и их размерами при помощи соответствующих иконок. 4) Контроллер Split визуализирует работу всех контроллеров при сборке схемы на экране, а также осуществляет масштабирование деталей проекта. 5) Функция Evaluate формирует состояние выхода текущего элемента путем выбора содержимого из ячейки кубита по ее адресу. 6) Модуль Q-sim реализует собственно алгоритм моделирования всех линий схемы путем построения на первом шаге рекурсивной модели для последовательно-параллельной обработки элементов. На втором шаге вычисляются состояния всех выходов логических элементов. Моделирование заканчивается после обработки всех тестовых входных последовательностей. Если схема имеет глобальные или локальные обратные связи, то моделирование осуществляется до фиксации одинаковых значений сигналов на всех линиях схемы. Если схема не устанавливается в устойчивое состояние на входных наборах, то фиксируется генераторный режим после выполнения n (=20) итераций. В этом случае всем изменяющимся линиям присваивается значение двоичной неопределенности X={0,1}. 7) Модуль управления библиотекой элементов, схем и проектов осуществляет считывание, запись и подключение фрагментов. 8) Модуль временных диаграмм осуществляет визуализацию теста с выходными сигналами на мониторе в форме непрерывных сигналов, разделенных на такты в абсолютном или модельном времени. 9) Все модули облачного сервиса запрограммированы на языке Swift, операционная система OSX 10.9, компилятор XCode 7. Количество исходных файлов 36, общее число строк кода – 1450.

На рис. 10 представлена визуализация результатов графического проектирования схемы с триггерами на мониторе компьютера. Данная схема полностью соответствует функциональности, представленной на рис. 7. Она содержит четыре входных порта для подачи рабочих или тестовых воздействий, а также два выходных порта. Структура содержит четыре логических элемента и два триггера. Мнемоническое описание компонентов схемы приведено к универсальной форме прямоугольника и различается только номером примитива в составе устройства, а также типом функциональности, которая задается кубит-вектором, представленным десятичным числом.

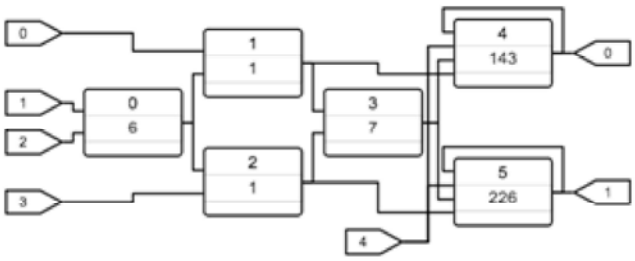


Рис. 10. Скриншот структуры с триггерами

В схеме, представленной на рис. 10, элементы имеют порядковые номера (в верхней части) и целые числа для идентификации функциональностей: 0/6 – 0110, 1/1 – 0001, 2/1 – 0001, 3/7 – 0111, 4/143 – 11110001, 5/226 – 01000111. Здесь двоичный вектор соответствует десятичному эквиваленту числа для задания функциональности. Поскольку кубит-вектор не имеет в явном виде задания входных наборов, то его можно рассматривать как неявную или компактную форму теоретико-множественной по сути таблицы истинности. Зачем явно указывать входные значения, если они составляют строго последовательную адресацию выходных значений? Таким образом, таблица истинности, как совокупность входных сигналов и соответствующих им выходных значений всегда проигрывает перед кубит-векторной формой представления функциональностей в плане объема и быстродействия анализа данных. Не существует принципиальных различий между описаниями комбинационного элемента, схемы или последовательностного примитива, поскольку все они формально представлены кубит-векторами, которые помещаются в адресуемую память. Более того, все примитивы схемы также являются адресуемыми, а структура схемы может быть описана в виде кубит-вектора. Таким образом, можно прийти к такой реализации вычислительного устройства, где нет ничего, кроме адресной памяти или кубитных векторов различной длины, в которых функциональности определяются упорядоченными наборами нулевых и единичных сигналов. Преимущества данного сервиса QuaSim кубитного описания и моделирования цифровых устройств заключаются в следующем: 1) Все функциональные элементы и схемы задаются Q-векторами, что унифицирует процедуры синтеза и анализа цифровых устройств. 2) Технологически просто менять или корректировать функциональность схемы или любого примитива путем замены отдельных битов Q-вектора. 3) Унификация кубитной формы описания примитивов схемы дает возможность применить к ним единственную процедуру анализа функциональностей, которая сводится к вычислению адреса $M_i = Q_i[M(X_i)]$, что делает процесс программирования облачного сервиса QuaSim технологически простым в исполнении и не зависящим от функциональной и структурной сложности цифровых структур. 4) Простой и понятный начинающему пользователю графический интерфейс делает облачный сервис конкурентоспособным на рынке образовательных услуг, где сложные и тяжеловесные средства моделирования от ведущих компаний планеты являются недоступными для университетов из-за их высокой стоимости, а для студентов – времязатратными по сложности подготовки HDL-спецификаций при рассмотрении небольших учебных проектов. 5) Унификация формы описания примитивов создает условия для технологичного решения задач синтеза, моделирования неисправностей, тестирования, верификации и диагностирования, основанные на операциях с кубит-векторами. 6) Недостатком кубитной или квантовой технологии описания и анализа цифровых структур можно считать некоторое уменьшение быстродействия моделирования по сравнению с существующими промышленными компиляторами, для ASIC и VLSI проектов, где объем reusable logic является доминирующим для достижения высокого быстродействия.

Структура взаимодействующих компонентов облачного сервиса QuaSim представлена на рис. 11. Квантовое или кубитное представление модели цифрового устройства вместе с интерпретативным симулятором составляют ядро системы, интегрированной в большие даные киберпространства или Интернета. Это дает возможность использовать в качестве исходных данных открытые спецификации и тестбенчи, описанные на языках VHDL, Verilog. Такие данные и/или тестовые примеры имеются практически во всех ведущих компаниях, университетах и тематических конференциях IEEE, TTTC, ISCAS. Кроме того, погружение Q-sim сервиса в интернет-пространство предполагает также выгрузку результатов его работы, связанную с анализом и синтезом учебных или рыночно ориентированных проектов в сервисы хранения данных на платформах Google, Amazon, Microsoft, IBM, Facebook. Естественно, что интеграция облачного сервиса с киберпространством предполагает наличие парсер-микросервисов для преобразования спецификаций из языков описания аппаратуры во внутренний язык QuaSim. Должно также существовать и обратное преобразование данных из кубитного представления в стандарты HDL-языков. Парсеризация обеспечивает вомножность использования открытых в интернете проектов для их изучения и сравнения в системе моделирования Q-sim, а также делает доступными внутренние проектные решения QuaSim для всех желающих на рынке образовательных сервисов.

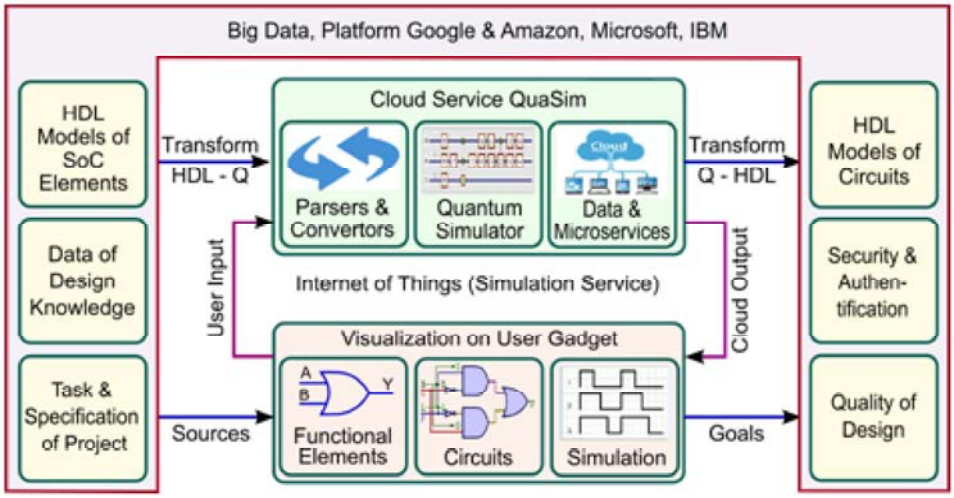


Рис. 11. Облачный сервис моделирования цифровых устройств

Блок Security контролирует доступ пользователей в целях их статистического учета и предполагает аутентификацию каждого на основе пароля, фамилии, имени, дополненной любым валидным (корпоративным) атрибутом из списка: {электронная цифровая подпись, e-mail, цифровой ключ, номер телефона}.

Тестирование и верификация облачного сервиса моделирования цифровых систем осуществлялись отдельно для каждого микросервиса, а затем во взаимодействии всех модулей.

1) Проверка правильности генерирования логических и более сложных функциональных элементов.

2) Проверка структурного синтеза цифровой схемы и средств визуализации.

3) Верификация и тестирование алгоритмов двоичного и троичного синхронного интерпретативного исправного моделирования входных воздействий на 40 схемах, комбинационного и последовательностного типов. При этом использовались тестовые наборы, алгоритмически генерируемые и составленные пользователем.

4) Проверка сервисных модулей, обеспечивающих работоспособность основных микросервисов: библиотеки элементов и схем, аутентификация пользователя, модуль формирования статистических данных по проектам и пользователям.

5) Верификация интерфейсных микросервисов, обеспечивающих интеракции между облачными back-end и пользовательскими front-end модулями.

9. Заключение

Сущность предлагаемого научно-технологического исследования заключается в создании векторных структур данных и кубитных методов синтеза, тестирования и моделирования, интегрированных в облачную инфраструктуру сервисного обслуживания компонентов цифровых систем на кристаллах в целях повышения качества изделий и выхода годной продукции за счет адресуемости всех вычислительных процессов и явлений. Основная инновационная идея Memory-Address-Transaction модели вычислений заключается в синтезе и анализе векторных цифровых структур на основе адресуемых элементов памяти, исключающих использование reusable or new logic. Трудно создать двумерный регистр, соответствующий матрице смежностей или таблице истинности, поэтому привести описание функции и структуры к единому одномерному формату, означает – технологично решать все задачи синтеза и анализа для функциональностей и графов в кубитно-векторной метрике, создающей memory-driven computing на основе выполнения параллельных логических операций.

Предлагается методология проектирования цифровых схем, на основе элементов памяти для синтеза компонентов операционного и управляющего автоматов, составляющих операционное устройство.

Показаны основы кубитно-векторного синтеза путем суперпозиции кубитных покрытий black box функциональностей, имплементируемых в элементы памяти, что дает возможность существенно повысить быстродействие средств моделирования, тестирования и верификации, а также упростить процессы создания реальных и виртуальных компьютерных систем.

Предложены кубитные структуры данных для моделирования и тестирования цифровых систем, которые дают возможность существенно упростить реализацию алгоритмов и повысить их быстродействие за счет адресуемости функциональных квантов и параллельности обработки примитивов.

Показана реализация вычислительных структур и процессов на основе использования адресного автомата, который дает возможность привлекать инфраструктуру стандартов тестопригодного проектирования для повышения выхода годной продукции, за счет online восстановления работоспособности функциональных примитивов.

Практическая значимость кубитной методологии синтеза и анализа цифровых систем заключается в имплементации процессора на основе элементов памяти, что делает его однородным по структуре функциональных примитивов и доставляет технологические удобства при реализации процессов проектирования, производства и эксплуатации, включая верификацию, встроенные тестирование, диагностирование и ремонт в режиме online за счет использования универсальных адресуемых spare-компонентов памяти. Кроме того, моделирование проектируемых вычислителей на основе адресуемых моделей элементов делает данную процедуру простой за счет регулярных структур данных и использования операции транзакции на элементах памяти, а также быстродействующей, благодаря параллельной обработке массивов однотипной памяти. Имплементация кубитных моделей описания цифровых компонентов и систем работает на увеличение выхода годной продукции, повышение надежности вычислительных изделий, снижение стоимости проектирования и изготовления, а также автономное восстановление работоспособности в режиме online без участия человека.

Предложен облачный сервис QuaSim для моделирования и верификации цифровых систем на основе транзакций между адресуемыми компонентами памяти для реализации любой функциональности. Описан новый подход к синтезу и анализу цифровых систем, использующий векторную форму (квант) задания комбинационных и последовательностных структур для их имплементации в элементы памяти, что существенно отличается от общепринятой теории проектирования дискретных устройств на основе таблиц истинности компонентов. Используются квантовые или кубитные структуры данных [1-5] для реализации вычислительных процессов в целях повышения быстродействия анализа цифровых систем и уменьшения объемов памяти на основе унарного кодирования состояний входных, внутренних и выходных переменных и имплементации кубитных векторов в элементы памяти FPGA, реализующих комбинационные и последовательностные примитивы.

Список литературы: 1. *Metodi T., Chong F.* Quantum Computing for Computer Architects. Synthesis Lectures on Computer Architecture. Morgan & Claypool. 2006. 154 p. **2.** *Stenholm Stig, Kalle-Antti Suominen.* Quantum approach to informatics. John Wiley & Sons, Inc. 2005. 249p. **3.** *Hahanov V.I., Wajeb Gharibi, Litvinova E.I., Shkil A.S.* Qubit data structure of computing devices // Electronic modeling. 2015. № 1. P.76-99. **4.** *Vladimir Hahanov, Tamer Bani Amer, Ivan Hahanov.* MQT-model for Virtual Computer Design // Proc. of Microtechnology and Thermal Problems in Electronics (Microtherm). 23-25 June 2015. P. 182-185. **5.** *Hahanov V.I., Litvinova E.I., Chumachenko S.V.* et al. Qubit Model for solving the coverage problem // Proc. of IEEE East-West Design and Test Symposium. Kharkov. 14-17 September, 2012. P.142 – 144. **6.** *Zorian Y. Shoukourian S.* Test solutions for nanoscale Systems-on-Chip: Algorithms, methods and test infrastructure. Computer Science and Information Technologies (CSIT), 2013. P. 1 – 3. **7.** *Zorian Y., Shoukourian S.* Embedded-memory test and repair: infrastructure IP for SoC yield. Design & Test of Computers, IEEE (Volume: 20, Issue: 3). P. 58 – 66. **8.** *Dugganapally I.P., Watkins S.E., Cooper B.* Multi-level, Memory-Based Logic Using CMOS Technology. VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on. Tampa, FL. P. 583-588. **9.** *Yueh W., Chatterjee S., Zia M., Bhunia S., Mukhopadhyay S.* A Memory-Based Logic Block With Optimized-for-Read SRAM for Energy-Efficient Reconfigurable Computing Fabric. Circuits and Systems II: Express Briefs, IEEE Transactions on. Vol. 62. Issue: 6. P. 593-597. **10.** *Matsunaga S., Hayakawa J., Ikeda*

S., Miura K., Endoh T., Ohno H., Hanyu T. MTJ-based nonvolatile logic-in-memory circuit, future prospects and issues. Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09.P. 433 – 435. **11.** *Harada S., Xu Bai, Kameyama M., Fujioka Y.* Design of a Logic-in-Memory Multiple-Valued Reconfigurable VLSI Based on a Bit-Serial Packet Data Transfer Scheme. Multiple-Valued Logic (ISMVL), 2014 IEEE 44th International Symposium on. P. 214 – 219. **12.** *Hahanov V.I., Tamer Bani Amer, Chumachenko S.V., Litvinova E.I.* Qubit technology analysis and diagnosis of digital devices // Electronic modeling. 2015. Vol. 37, № 3. P. 17-40. **13.** *Melikyan V.Sh.* A method of eliminating false paths during statistical static analysis of timing delays of digital circuits // Elektronica i svyaz. 2009. Vol. 2-3, No. 1. P. 93-96. **14.** *Melikyan V.Sh., Vatyan A.O.* Interconnections model delays for the logic analysis of ECL circuits //S UAB, Vol. 2, Computer Engineering, Moscow, 1997. P. 187-194. **15.** *Хаханов І.В., Литвинова Є.І.* Синтез та аналіз «квантових» моделей цифрових систем // АСУ та прилади інформатики. 2015. Вип. 172. С. 56–70.

Поступила в редколлегию 14.01.2016

Tamer Bani Amer, аспирант ХНУРЭ. Научные интересы: квантовые вычисления, тестирование и диагностика цифровых систем. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. +3805770-21-326.

Хаханов Иван Владимирович, студент факультета компьютерной инженерии и управления ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, программирование. Увлечения: горные лыжи, английский язык. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326.

Литвинова Евгения Ивановна, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326. E-mail: kiu@kture.kharkov.ua.

Емельянов Игорь Валерьевич, н.с. кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. +3805770-21-326.