СИНТЕЗ КУБИТНЫХ МОДЕЛЕЙ ЛОГИЧЕСКИХ ФУНКЦИЙ

Владимир Хаханов, Багхдади Аммар Авни Аббас, Олеся Гузь, Ирина Хаханова
Харьковский национальный университет радиоэлектроники, Украина
hahanov@kture.kharkov.ua

Abstract. *A superposition method for synthesis of functionality cube is proposed. It is characterized by using the structure of primitive elements and allows reducing the time of generating digital device model, focused on implementation in PLD. The method for generating the functionality cube allows simplifying solving the problem of synthesis of digital structure on chip through the use of components (LUT, Slice, CLB).*

Ключевые слова: *квантовые вычисления, кубитный процессор, диаграмма Хассе.*

I. Введение

Квантовый компьютер представляет собой вычислительное устройство, использующее при работе квантово механические явления: суперпозицию состояний (superposition), квантовую запутанность (entanglement), интерференцию (interference), параллелизм (parallelism) и обратимость вычислений (reversible computation) [1-5], что позволяет преодолеть ограничения по быстродействию классических компьютеров.

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

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

II. Синтез куба функциональности методом суперпозиции

Формально существует два пути синтеза функциональности цифровой схемы, имеющей n входов. Первый формирует вектор длиной n, путем заполнения нулевыми и единичными значениями функции на основе прямого моделирования всех $q = m \times 2^n$ входных воздействий на m примитивах. Второй – метод суперпозиции кубов примитивных элементов – также осуществляет доопределение всех координат вектора состояний выходов, но путем суперпозиции m кубических покрытий, входящих в состав схемной структуры. В этом случае вычислительная сложность получения функционального покрытия равна $q = 2 \times m$, при условии, что структурные компоненты схемы предварительно были ранжированы в соответствии с порядком распространения сигналов, а покрытие каждого примитива имеет 2 куба.

Пример получения покрытия вторым путем для функциональности

$f(X)=(X_1X_2)\vee(X_3X_4)$, имеющей три двухвходовых примитива (and, or, or), представлен ниже. Для этого используются два покрытия: C(and)=0001, C(or)=0111. Их взаимодействие как декартово произведение относительно бинарной операции or формирует следующий результат:

$$f(X)=(X_1X_2)\vee(X_3X_4)=(0001)\vee(0111)=\begin{array}{c|cccc}\vee & 0 & 1 & 1 & 1\\ \hline 0 & 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{array}=(0111\ \ 0111\ \ 0111\ \ 1111).$$

Более сложный вариант взаимодействия представлен логической функцией:

$$f(X)=(X_1X_2)\vee X_3=0001\vee 01=\begin{array}{c|cc}\vee & 0 & 1\\ \hline 0 & 0 & 1\\ 0 & 0 & 1\\ 0 & 0 & 1\\ 1 & 1 & 1\end{array}=(01\ \ 01\ \ 01\ \ 11).$$

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

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

Здесь процедура получения куба выходных значений дополнена шагом минимизации, которая может существенно упростить логическую структуру за счет исключения несущественных переменных.

Процесс модель получения покрытия логической функциональности, путем ее последовательного разложения по n переменным, включает следующие пункты:

Выполнение декартова произведения $\{\vee_\times,\wedge_\times,\oplus_\times\}$ логической функции от двух (n) переменных в целях формирования вектора выходных значений, размерностью $p=2^n$. Здесь каждый бит вектора одной переменной X_i взаимодействует по логической операции с каждым битом вектора другой переменной X_j: $\begin{bmatrix}X_1 & & X_2\\ 0 & \wedge_\times & 0\\ 1 & & 1\end{bmatrix}=\begin{matrix}X_1\wedge X_2\\0\\0\\0\\1\end{matrix}$. Размерность полученного вектора равна $p=p_i\times p_j$, где p_i,p_j – разрядности двоичных векторов при соответствующих переменных. Последовательное выполнение декартовых (векторных) логических операций над всеми примитивами (логическими переменными): $P=\prod_{i=1}^{n}P_i$ для получения куба логической функциональности максимальной размерностью $p=2^n$.

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

Исключение противоречивых входных воздействий, если термы логической функции имеют одни и те же переменные. Демонстрацией данной процедуры может служить синтез

куба функциональности для выражения:

$$Y = X_1X_2 \vee \overline{X}_2X_3 = \begin{bmatrix} X_1X_2 & & \overline{X}_2X_3 \\ 0 & & 0 \\ 0 & \vee_{\times} & 1 \\ 0 & & 0 \\ 1 & & 0 \end{bmatrix} = \begin{matrix} Y \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{matrix} \begin{bmatrix} X_1^1X_2^2X_2^3X_3^4 \\ 0000 \\ 0001 \\ 0010 \\ 0011 \\ 0100 \\ 0101 \\ 0110 \\ 0111 \\ 1000 \\ 1001 \\ 1010 \\ 1011 \\ 1100 \\ 1101 \\ 1110 \\ 1111 \end{bmatrix} \xrightarrow{X_2^2=X_2^3} \begin{matrix} Y \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{matrix} \begin{bmatrix} X_1X_2X_3 \\ 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix}$$

После получения куба функциональности от «четырех» переменных, среди которых одна переменная на позициях 2 и 3 повторяется $X_1^1X_2^2X_2^3X_3^4$, необходимо преобразовать таблицу истинности к трем фактически существующим переменным. Для этого необходимо исключать те строки таблицы истинности, которые имеют неравные значения координат для одной и той же переменной $X_2^2 \neq X_2^3$. В результате выполнения такой процедуры получается функция от фактического числа переменных, заданная, в данном случае, кубом C=(01000111). В целях уменьшения объема памяти для хранения промежуточных результатов можно исключать противоречивые состояния переменных непосредственно при выполнении декартовых логических операций. Сама декартова операция для любого числа координат взаимодействующих векторов также может выполняться за один такт.

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

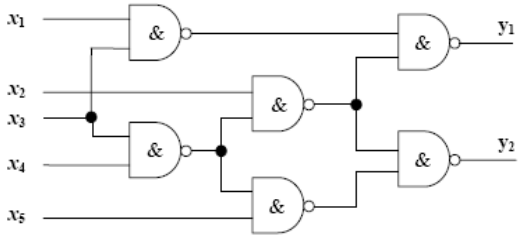

Рис. 1. Схемная тест-структура из библиотеки ISCAS

Реализация процесс модели синтеза общей функциональности представлена следующими пунктами:

1. Создание библиотеки примитивов в форме кубов их функциональностей:

$C_i = (0001), i = \overline{1,6}$. Для данной схемы имеет место один тип двухвходового примитива:

$$C_{\text{and–not}} = \begin{bmatrix} \overline{X_1X}_2 & Y \\ 00 & 1 \\ 01 & 1 \\ 10 & 1 \\ 11 & 0 \end{bmatrix} \begin{matrix} C \\ 1 \\ = 1 = (1110). \\ 1 \\ 0 \end{matrix}$$

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

$$C_5(Y_1) = C_1(\overset{C_3}{\overline{X_2C_2}}) = \overline{\overline{(\overline{X_1X_3})[X_2(\overline{X_3X_4})]}};$$

$$C_6(Y_2) = \overset{C_3}{\overline{X_2C_2}}\,\overset{C_4}{\overline{(C_2X_5)}} = \overline{\overline{X_2\overline{X_3X_4}}\;\overline{X_3X_4X_5}}$$

$$C_5 = \begin{matrix} \overline{X_1X_3} \\ 1 \\ 1 \\ 1 \\ 0 \end{matrix} \overline{\wedge}_{\times} \begin{bmatrix} & \overline{X_3X_4} \\ X_2 & 1 \\ 0 \;\overline{\wedge} & 1 \\ 1 & 1 \\ & 0 \end{bmatrix} = \begin{bmatrix} \overline{X_1X_3} & X_1X_3 \\ 1 & 00 \\ 1 & 01 \\ 1 & 10 \\ 0 & 11 \end{bmatrix} \overline{\wedge}_{\times} \begin{bmatrix} \overline{X_2\overline{X_3\overline{X}_4}} & X_2X_3X_4 \\ 1 & 000 \\ 1 & 001 \\ 1 & 010 \\ 1 & 011 \\ 0 & 100 \\ 0 & 101 \\ 0 & 110 \\ 1 & 111 \end{bmatrix} = (0001\ 0010\ 0101\ 1111);$$

$$C_6 = \begin{bmatrix} & \overline{X_3X_4} \\ X_2 & 1 \\ 0 \;\overline{\wedge}_{\times} & 1 \\ 1 & 1 \\ & 0 \end{bmatrix} \overline{\wedge}_{\times} \begin{bmatrix} \overline{X_3X_4} & \\ 1 & X_5 \\ 1 \;\overline{\wedge}_{\times} & 0 \\ 1 & 1 \\ 0 & \end{bmatrix} = \begin{bmatrix} \overline{X_2\overline{X_3\overline{X}_4}} & X_2X_3X_4 \\ 1 & 000 \\ 1 & 001 \\ 1 & 010 \\ 1 & 011 \\ 0 & 100 \\ 0 & 101 \\ 0 & 110 \\ 1 & 111 \end{bmatrix} \overline{\wedge}_{\times} \begin{bmatrix} \overline{\overline{X_3\overline{X}_4X}_5} & X_3X_4X_5 \\ 1 & 000 \\ 0 & 001 \\ 1 & 010 \\ 0 & 011 \\ 1 & 100 \\ 0 & 101 \\ 1 & 110 \\ 1 & 111 \end{bmatrix} =$$

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

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

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

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

III. Имплементация функциональности в технологическую структуру PLD

Решение задачи сводится к разбиению куба функциональности на сегменты, которым можно поставить в соответствие библиотечные примитивы конкретного кристалла. На кристалле PLD (FPGA) имеется в наличии определенное количество четырехвходовых примити-

вов – элементов памяти для хранения таблицы истинности LUT (Look Up Table). Примитивы могут объединяться в логические секторы (Slice), состоящие из двух примитивов, объединенных мультиплексором, что дает возможность увеличить разрядность входных переменных до пяти. Последующее объединение двух секторов с помощью мультиплексора увеличивает разрядность входных переменных до шести. При этом на кристалле технологически предусмотрено, что каждые 2 пары секторов мультиплексируются в блоки CLB (Configurable Logic Block), что дает возможность увеличить число входов синтезируемой логической функции, в пределах одного CLB, до семи переменных. Структура упомянутых выше модулей (LUT, Slice, CLB) представлена на рис. 2.

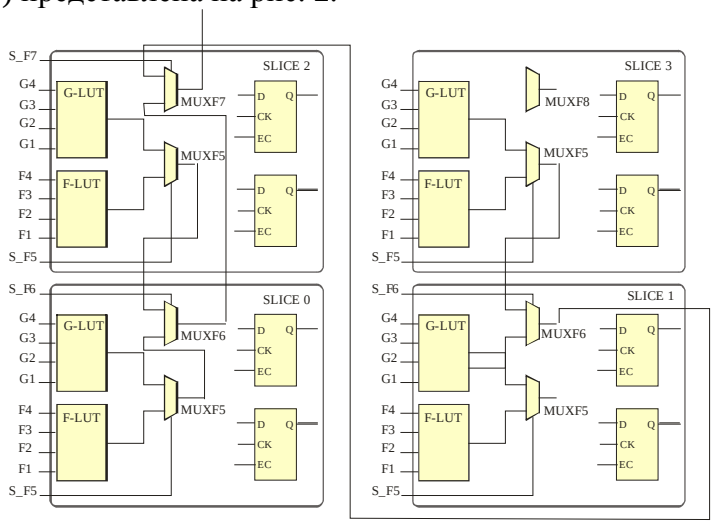

Рис. 2. Структура моделей PLD

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

Для оценки аппаратной сложности при синтезе или реализации вычислительной структуры необходимо иметь обратную зависимость числа логических блоков CLB от сложности логической функции, приведенной к количеству переменных n или мощности куба функциональности $N = 2^n$. Например, для функции от n=10 переменных необходимо иметь на кристалле 8 CLB. В общем случае число логических блоков CLB, в зависимости от числа переменных синтезируемой функции, равно $N = 2^{n-7}$, $n = 7,8,9, \ldots$.

Что касается количества примитивов LUT, то здесь оценка аппаратной сложности будет представлена следующим выражением: $N^* = 8L \times 2^{n-7} + 4M \times 2^{n-7} = (8L + 4M) \times 2^{n-7}$. Здесь 8L, 4M – есть аппаратная сложность реализации LUT и мультиплексора, входящих в состав CLB. При этом важные характеристики реализации функциональности – структурная глубина, как число уровней иерархии, выраженное в количестве LUT и/или мультиплексорах на самом длинном логическом пути и его задержка D – также зависят от числа переменных (задержки мультиплексора): $S = (n - 3)$; $D = (n - 3) \times D_M$.

IV. Кубит-процессор (Квантовый процессор)

В качестве примера предлагается решить задачи поиска оптимального единичного покрытия всех столбцов минимальным числом строк матрицы M, представленной на рис.3. Для этого необходимо сделать перебор всех 255 сочетаний: из восьми по одной строке, по двум, трем, четырем, пяти, шести, семи и восьми. Минимальное количество примитивов (строк), формирующее покрытие, есть оптимальное решение. Таких решений может быть несколько. Диаграмма Хассе есть компромиссное предложение, относительно времени и памяти, или такая стратегия решения задачи покрытия, когда ранее полученный результат впоследствии используется для создания более сложной суперпозиции. Поэтому для каждой таблицы покрытия, содержащей n примитивов (строк), необходимо генерировать собственную мультипроцессорную структуру в форме диаграммы Хассе, которая далее должна быть использована для почти параллельного решения NP-полной задачи. Например, для четырех строк таблицы покрытия диаграмма Хассе – структура мультипроцессора – показана на рис. 3.

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

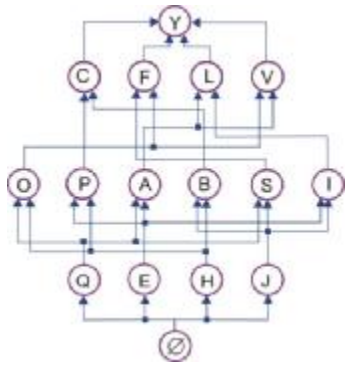

Рис. 3. Квантовая структура вычислительных процессов

Оптимальные решения задачи покрытия для матрицы M, которая генерирует 255 вариантов возможных сочетаний, представлены строками в форме ДНФ: $C = fgh \vee efg \vee cdf$.

V. Заключение

Предложен новый суперпозиционный метод синтеза куба функциональности, который характеризуется использованием структуры примитивных элементов и для определенного класса устройств позволяет существенно уменьшить время получения модели цифрового устройства, ориентированной на имплементацию в кристаллы PLD. Метод построения куба функциональности позволяет существенно упростить решение задачи синтеза цифровой структуры в кристалле на основе использования компонентов (LUT, Slice, CLB), что представляет определенный интерес для его промышленного использования.

VI. Библиография

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

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

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

4. Glassner A. Quantum computing. 3. // IEEE Computer Graphics and Applications.– Nov/Dec 2001.– Vol. 21, Issue 6.– P. 72 – 82.

5. Marinescu D.C. The Promise of Quantum Computing and Quantum Information Theory – Quantum Parallelism // Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05).– 2005.– P. 112-114.