УДК 004.272.2

Особенности проектирования модулярных умножителей с помощью современных САПР

Д.В. Тельпухов, Р.А. Соловьев, Е.С. Балака, В.С. Рухлов, А.С. Михмель

Институт проблем проектирования в микроэлектронике РАН, nofrost@inbox.ru

***Аннотация*—В статье исследуются различные подходы к проектированию модулярных умножителей, которые требуют максимальной производительности для использования в задачах цифровой обработки сигналов. На основе сравнения характеристик различных реализаций умножителей, синтезированных в базисе ПЛИС и заказных СБИС, были выявлены наиболее эффективные методы их построения.**

***Ключевые слова* — модулярная арифметика, модулярный умножитель, ПЛИС, СБИС.**

I. Введение

Современные САПР для проектирования электроники заточены на эффективную реализацию устройств, построенных на базе позиционной системы счисления. В процессе синтеза устройства из внутренней библиотеки САПР выбирается оптимальный вариант архитектуры того или иного функционального блока, отвечающего заданным параметрам по быстродействию, площади, потребляемой мощности и т.п. [1]. В то время как для нетрадиционных систем счисления, в частности, модулярной системы счисления, требуется дополнительно оценивать, какая из множеств разработанных реализаций, является наиболее эффективной в том или ином случае.

Данная работа посвящена сравнительному анализу подходов к построению модулярных умножителей. Исследовано шесть наиболее популярных архитектур, проведена их оценка по аппаратным и временным затратам в двух технологических базисах ПЛИС (САПР Altera Quartus II) и СБИС (САПР Synopsys DC).

II. Умножители по модулю

Эффективная реализация модулярных умножителей напрямую зависит от того, в какой степени и насколько удачно они могут быть адаптированы к современным технологиям проектирования. Арифметическая операция умножения по модулю над остатками *x, y* может приводить к выходу результата операции $|x \cdot y|_p$ за диапазон Z_p, тогда требуется корректировка результата, т.е. взятие от числа $x \cdot y$ вычета по mod *p*. Операция взятия вычета $|x \cdot y|_p$ выражается формулой:

$$|x \cdot y|_p = x \cdot y - \left\lfloor \frac{x \cdot y}{p} \right\rfloor \cdot p \cdot$$

Эта формула непосредственно связана с аксиомой Архимеда теории действительных чисел [2], которая предопределяет аддитивный характер вычислений с остатками по mod p.

Наиболее распространенным методом адаптации является выбор оснований модулярной арифметики максимально близкими к степеням двойки. Достоинство выбора модулей в виде $p=2^t \pm 1$ состоит в том, что все модульные операции по этим модулям над остатками, представленными двоичным позиционным кодом, могут быть получены без формирования величины $\left\lfloor \frac{x}{p} \right\rfloor$. Модули специального вида имеют ряд неоспоримых преимуществ, позволяя лучше реализовывать модульные и немодульные устройства. Однако, использование алгоритмов для произвольных оснований, открывает перспективы для более гибкого выбора модулей, что в свою очередь позволяет глубже распараллелить всю вычислительную структуру.

Для аппаратной реализации модулярных умножителей в литературе предложено несколько эффективных методов.

A. *Индексный умножитель*

Реализация индексных умножителей основана на математических свойствах полей Галуа. Одним из наиболее важных свойств полей Галуа является то, что любой ненулевой элемент поля может быть выражен через некоторое число, называемое первообразным или примитивным корнем. Данное свойство может быть использовано для осуществления операции умножения над элементами поля [3]. Каждому целому числу q из диапазона $(0, p)$ ставится в соответствие число *i* такое, что $q = w^i \bmod p$, откуда

$$|x \cdot y|_p \longleftrightarrow w^{|i_x + i_y|_{p-1}}.$$

Структура индексного умножителя представлена на рис. 1.

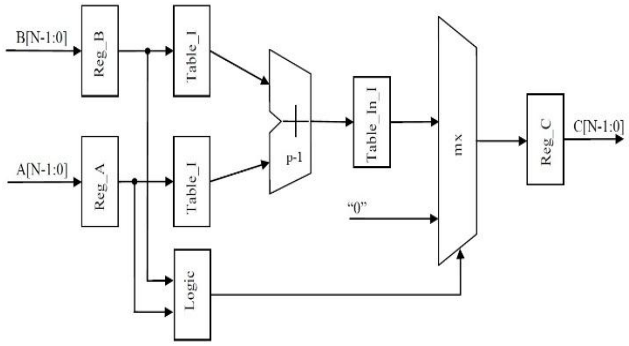

Рис. 1. Структура модульного канала индексного умножителя

B. Умножитель на базе метода разности квадратов

Метод построения умножителя основан на арифметическом выражении:

$$A \cdot B = \frac{(A+B)^2}{4} - \frac{(A-B)^2}{4}.$$

Умножение по этому методу может быть реализовано напрямую, с использованием трех сумматоров (один для сложения и два для вычитания) и двух таблиц подстановок, которые производят четверть-квадрирование (рис. 2). Каждая из таблиц подстановок содержит $2p-1$ строк и реализует функцию $\left|Q^2|4^{-1}|_p\right|_p$, где $Q = A + B$ или $Q = A - B$ соответственно. Отметим, что метод работает только для нечетных модулей.

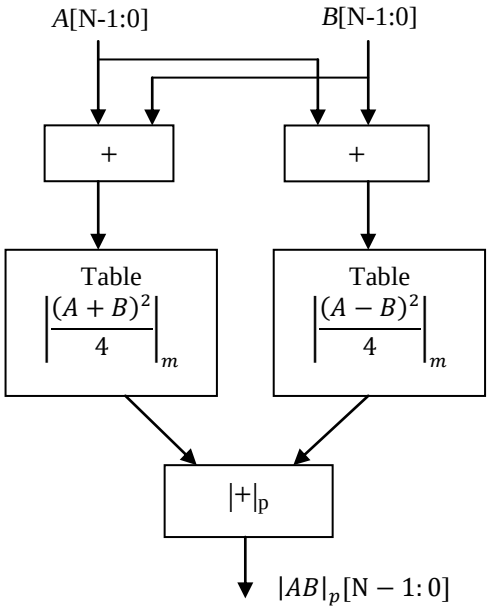

Рис. 2. Умножитель на базе разности квадратов

C. Умножитель на основе позиционного умножителя с прямым преобразователем

Из-за небольшой разрядности входных данных оба блока получаются довольно компактными. Умножение реализуется посредством двоичного умножителя, для реализации которого есть множество очень быстрых реализаций [4]. Результат умножения подается на вход модуля, который вычисляет значение вычета по модулю p (рис. 3).

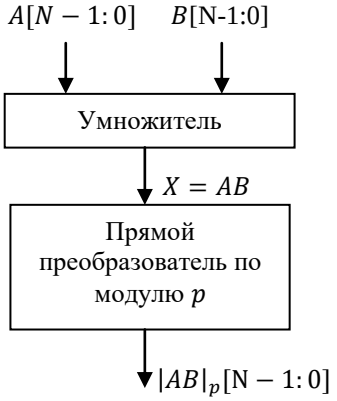

Рис. 3. Умножитель с преобразователем по модулю *p*

Рассмотрим подробнее преобразователь (рис. 4). На вход преобразователя приходит результат умножения (двоичный умножитель) число X в позиционном виде разрядности N-бит. Требуется найти остаток от его деления (вычет) на число p. Пусть p имеет размерность k бит.

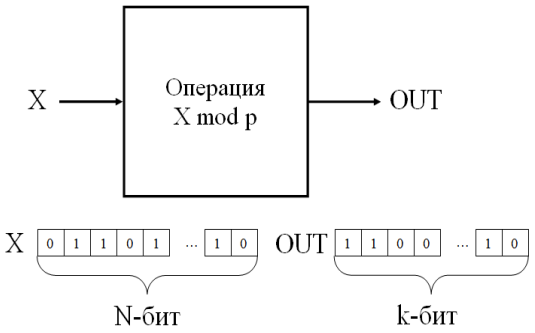

Рис. 4. Схема блока операции нахождения вычета по модулю

Здесь:

$0 \le X < 2^N$, $0 \le OUT < p$, где p известно на этапе проектирования устройства.

При некоторых соотношениях между входными данными и заданным p, модуль взятия остатка от деления можно реализовать простейшим образом. N незначительно превосходит разрядность модуля:

В этом случае достаточно проверить, в какой из диапазонов вида $[0; p), [p; 2p), [2p; 3p)$, ... попадает X и вычесть из него поправочный коэффициент,

соответствующий диапазону: 0 для $[0;p)$, p для $[p;2p)$, $2p$ для $[2p,3p)$ и так далее.

Рассмотрим общий случай. Представим входные данные X в следующем виде:

$$X = 2^0 \cdot X[0] + 2^1 \cdot X[1] + 2^2 \cdot X[2] + \ldots + 2^{N-1} \cdot X[N-1]$$

и воспользуемся следующими свойствами вычетов:

$$|A+B|_p = \left||A|_p + |B|_p\right|_p \text{ и} |A \cdot B|_p = \left||A|_p \cdot |B|_p\right|_p.$$

Тогда вычет X по модулю p можно записать следующим образом:

$$\begin{aligned} |X|_p = |2^0 \cdot X[0] + 2^1 \cdot X[1] + 2^2 \cdot X[2] + \ldots + 2^{N-1} \\ \cdot X[N-1]|_p \\ = \left||2^0|_p \cdot X[0] + |2^1|_p \cdot X[1] + |2^2|_p \right. \\ \left. \cdot X[2] + \ldots + |2^{N-1}|_p \cdot X[N-1]\right|_p \\ = |A_0 \cdot X[0] + A_1 \cdot X[1] + A_2 \\ \cdot X[2] + \ldots + A_{N-1} \cdot X[N-1]|_p \end{aligned}$$

Константы вида $A_i = \left|2^i\right|_p$ могут быть рассчитаны на этапе проектирования устройства и каждая константа не превосходит значение p . А общее значение выражения $SUM = (A_0 \cdot X[0] + \ldots + A_{N-1} \cdot X[N-1]) \leq (p-1) \cdot N$. Так как коэффициенты A_i известны, то оценку максимально возможного значения SUM можно сократить ещё больше.

Таким образом, чтобы посчитать значение вычета $|X|_p$ требуется выполнить две операции:

1) Найти сумму $SUM = (A_0 \cdot X[0] + \ldots + A_{N-1} \cdot X[N-1]) \leq (p-1) \cdot N$.

2) Определить в какой интервал вида $[0;p), [p;2p), [2p;3p), \ldots [(N-1) \cdot p; N \cdot p)$ попадает значение SUM и вычесть из него соответствующее значение 0 для $[0;p)$, p для $[p;2p)$, $2p$ для $[2p,3p)$ и так далее.

D. Умножитель на базе частичных сумм

Умножитель по модулю, реализованный на базе частичных сумм с редуцированными по модулю коэффициентами. Аналитическое представление:

$$\begin{aligned} X = |AB|_m = |(A_0 * 2^0 + A_1 * 2^1 + \ldots + A_k * 2^k) \\ * (B_0 * 2^0 + B_1 * 2^1 + \ldots + B_k \\ * 2^k)|_m \\ = |(A_0 * B_0 * 2^{0+0} + A_0 * B_1 * 2^{0+1} \\ + \ldots + A_k * B_k * 2^{k+k})|_m \end{aligned}$$

Структурная схема умножителя представлена на рис. 5.

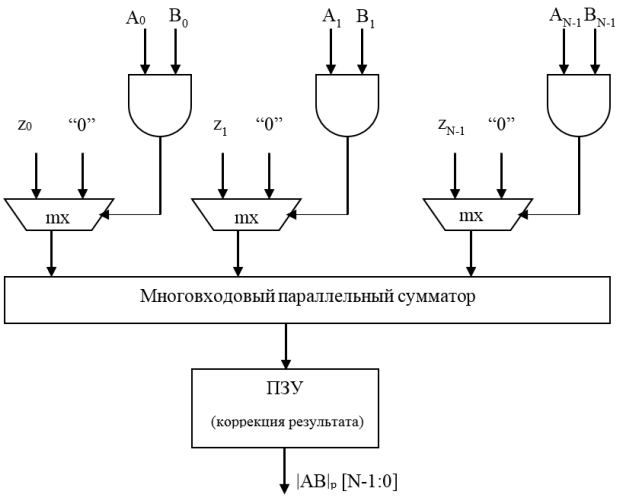

Рис. 5. Умножитель на базе частичных сумм

E. Умножитель на базе полностью табличной реализации

Данный умножитель реализуется посредством таблицы: каждому набору входных воздействий ставится в соответствие значение выходного сигнала. Такие умножители обладают отличными показателями времени задержки на критическом пути, но занимают много площади [5]. Площадь схемы имеет квадратичный рост $O(N^2)$ в зависимости от значения модуля. Время синтеза средствами САПР, а также расход ОЗУ, пропорциональны площади схемы. Как показали эксперименты, этот подход разумно использовать только для небольших значений модулей.

F. Умножитель на базе минимизированной таблицы истинности

Модулярный умножитель, как и любую другую логическую схему можно представить в виде таблицы истинности. Если размерность модуля p равна k бит, то таблица истинности будет иметь $2k$ входных столбцов и 2^{2k} строк. Для выходов будет соответственно k столбцов.

Таблица истинности минимизируется с помощью одного из доступных способов минимизации, например, с помощью эвристического алгоритма Espresso [6]. Для большинства умножителей (до 7 бит включительно) удалось построить точное решение, для остальных найти решение близкое к оптимальному.

Минимизированные таблицы истинности представляются в виде ДНФ в Verilog формате. Все сгенерированные умножители по этому методу доступны по ссылке [7].

III. Схема экспериментов

Для оценки эффективности рассматриваемых методов проектирования модулярных умножителей были написаны генераторы Verilog. С помощью этих генераторов, а также в автоматическом режиме

локально (для умножителей на базе минимизированной таблицы истинности), были созданы по 53 функциональных Verilog-описания каждого из рассматриваемых умножителей по простому модулю в диапазоне от 3 до 253.

Синтез схем, реализованных в СБИС, проводился в среде Synopsys Design Compiler в базисе созданной нами тестовой библиотеки. Библиотека состоит из трех логических элементов: INV, AND, OR. В данной серии экспериментов используются специальные вентильные условные единицы. Так, временная задержка для каждого вентиля устанавливается одна и та же и равна одной условной временной единице (1 у. е.). Также и с площадью вентиля. Площадь каждого вентиля равна одной условной единицы площади (1 у. е.^{2}) . Настройки для синтезатора для всех рассматриваемых схем были выставлены на минимизацию длины критического пути. Режим компиляции - *compile_ultra*.

Синтез схем, реализованных в ПЛИС, проводился в среде Altera Quartus II версии 10.0. Для исследования был выбран чип Altera Cyclone EP1C20F324C6. Ограничение данной ПЛИС – 20060 логических вентилей.

Для схем умножителей определялась задержка на критическом пути и площадь.

IV. Экспериментальные результаты.

На рис. 6 представлены результаты моделирования задержки критического пути (а) и площади (б) для 6 рассмотренных архитектур умножителей в базисе СБИС. Как видно из графиков, методы на базе Espresso и обычная табличная реализации всегда превосходят остальные методы по скорости, но значительно уступают им по площади. При этом площадь растет так сильно, что использовать табличные умножители становится нерационально уже на модулях порядка 31 (рис. 6в). Наименьшие аппаратные затраты демонстрирует умножитель на базе двоичного умножителя с прямым преобразователем. Если требуется минимизировать задержку на критическом пути и при этом получить приемлемую площадь устройства, то наилучшие показатели у индексного метода.

Как видно из графиков «табличный метод» и «метод на базе Espresso» имеют схожую задержку и похожие показатели по площади. Очевидно, синтезатор Synopsys использует похожее на Espresso средство минимизации логических функций. Поэтому при прочих равных условиях нет смысла в отдельной минимизации таблицы истинности перед синтезом.

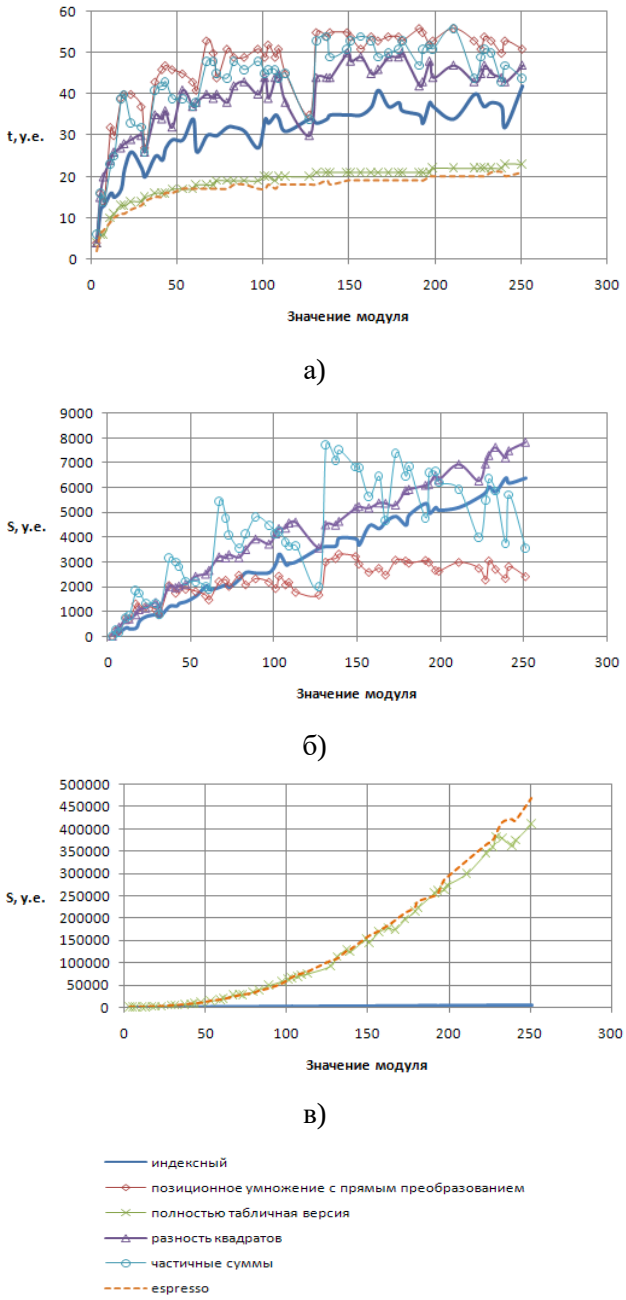

Рис. 6. Результаты моделирования умножителей по модулю в базисе СБИС

На рис. 7 представлены результаты моделирования задержки (а) и количества задействованных логических вентилей (б) для 6 рассмотренных архитектур умножителей в базисе ПЛИС.

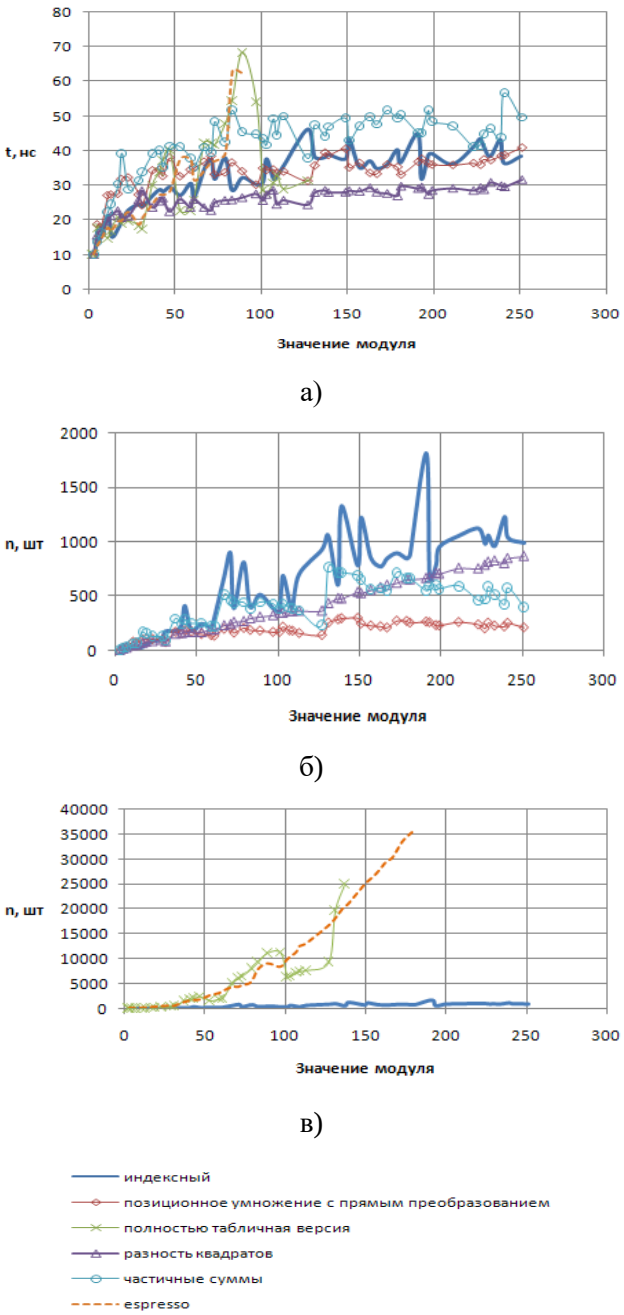

Рис. 7. Результаты моделирования умножителей по модулю в базисе ПЛИС

Как видно из графиков, табличный метод и метод на базе минимизации здесь уже не являются такими привлекательными в плане задержки, как это было в СБИС. Так же и с площадью схемы (ее рост еще больше, чем у СБИС) (рис. 7в). Для табличного метода в данной ПЛИС Quartus не смог синтезировать умножители для модулей больше чем 127, а для минимизированных таблиц истинности для модулей больше 97. Впрочем, это является ограничением данной конкретной ПЛИС, и верхняя граница модуля может быть выше для других ПЛИС, где число логических ячеек больше. Наименьшие аппаратные затраты демонстрирует, как и в случае СБИС, умножитель на базе позиционного умножителя с прямым преобразователем. Если рассматривать комплексно задержку на критическом пути и площадь схемы, то наилучшие показатели у метода разности квадратов.

V. Заключение

В статье исследованы методы реализации умножителей по модулю, а также проведены исследования их характеристик быстродействия и аппаратных затрат. На основании данных экспериментов, можно выделить табличную реализацию умножителя как самую аппаратно затратную. И если для схем СБИС применение этого метода можно хоть как-то оправдать превосходными показателями значения задержки, то для схем ПЛИС применение данного метода не оправдывает себя ни по временной задержке, ни, тем более, по используемой площади. Нужно также отметить, что в схеме ПЛИС серьезным ограничителем является количество логических вентилей, которые возможно задействовать. В целом, результаты демонстрируют схожие тенденции, как для СБИС, так и для ПЛИС, что подтверждает универсальность представленных методов. Эксперименты показали, что лучшим методом в общем случае для построения умножителей по модулю для СБИС является индексный метод, а для ПЛИС – метод разности квадратов.

Поддержка

Работа выполнена при поддержке гранта РФФИ-14-07-00004.

Литература

[1] DesignWareLibrary [Электронныйресурс]: Synopsys. Silicon to Software URL: https://www.synopsys.com/IP/SOCINFRASTRUCTUREIP/DESIGNWARE/Pages/default.aspx (датаобращения 25.05.2016)

[2] Виноградов И.М. «Основы теории чисел», издание шестое. М.: Наука, 1981.

[3] В.М. Амербаев, Д.Б. Малашевич Анализ эффективности реализации модульных операций индексной модулярной арифметики // ЭЛЕКТРОНИКА. Известия вузов, 2009, №6 (80). С. 54-57.

[4] Baham Rashidi, Reza Rezaeian Farashahi, Sayed Masoud Sayedi, “Efficient Implementation of Low Time Complexity and Pipelined Bit-Parallel Polynomial Basis Multiplier over Binary Finite Fields” The ISC Int’l Journal of Information Security, July 2015, Volume 7, Number 2, pp. 101-114.

[5] Ирхин В. П. Табличная реализация операций модулярной арифметики // Сб.науч.тр. Юбилейной Международной научно-технической конференции «50 лет модулярной арифметики», 2005. СС. 268-273.

[6] Espresso. Минимизация булевых функций. [Электронный ресурс]. – Режим доступа: http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/ (дата обращения 25.05.2016)

[7] Сайт Отдела методологии проектирования интегральных схем ИППМ РАН / [Электронный ресурс]. – Режим доступа: http://icdm.ippm.ru/ (дата обращения 25.05.2016).

Design features of the RNS-based multipliers using advanced CAD

D.V. Telpukhov, R.A. Sovovyev, E.S. Balaka, V.S. Rukhlov, A.S. Mikhmel

Institute for Design Problems in Microelectronics RAS, nofrost@inbox.ru

***Keywords* — Residue Number System, RNS-based multiplier, FPGA, VLSI.**

ABSTRACT

This work is devoted to the analysis of different approaches to modular multiplier design using modern CAD systems. Internal CAD libraries contain optimized architectural solutions for functional blocks, which can be selected depending on the device input constraints. As there are no such decisions for modular devices in the libraries, we have to estimate which of the many developed functional block implementations are the most effective.

In this paper methods of implementing modulo multipliers are investigated and their performance characteristics and hardware costs are studied. In course of work, six most popular architectures of modulo multipliers have been analyzed. To evaluate their effectiveness for two technological bases (CAD Quartus II FPGA Altera and Synopsys Design Compiler for ASIC), two-input multipliers were synthesized for a wide range of basic modules.

Based on these experiments, we can highlight look-up table multiplier implementation as the most area-consuming. Moreover, if for VLSI circuits the use of this method is justified by their excellent performance, for FPGA circuits this method is not justified either with regard to delay, or, even more so, with regard to area. It should also be noted that for FPGA circuit the number of logic gates we can use is a serious limitation. Generally, the results show similar trends for both VLSI and FPGA, which confirms the versatility of the presented methods. Experiments have shown that normally the best method to build modulo multipliers for VLSI is the index method, and for FPGA - difference of squares method.

SUPPORT

This work was supported by RFBR-14-07-00004 grant.

REFERENCES

[1] DesignWare Library: Synopsys. Silicon to Software URL: https://www.synopsys.com/IP/SOCINFRASTRUCTUREIP/DESIGNWARE/Pages/default.aspx (accessed 25.05.2016)

[2] Vinogradov I.M. «Osnovy teorii chisel», izdanie shestoe. Moscow, Nauka, 1981 (in Russian).

[3] V.M. Amerbaev, D.B. Malashevich Analysis of the effectiveness of the implementation of modular operations modular arithmetic index // EHLEKTRONIKA. Izvestiya vuzov, 2009, No. 6 (80). pp. 54-57 (in Russian).

[4] Baham Rashidi, Reza Rezaeian Farashahi, Sayed Masoud Sayedi, “Efficient Implementation of Low Time Complexity and Pipelined Bit-Parallel Polynomial Basis Multiplier over Binary Finite Fields” The ISC Int’l Journal of Information Security, July 2015, Volume 7, Number 2, pp. 101-114.

[5] Irhin V. P. Tabular implementation of modular arithmetic operations // Sb.nauch.tr. YUbilejnoj Mezhdunarodnoj nauchno-tekhnicheskoj konferencii «50 let modulyarnoj arifmetiki», 2005. pp. 268-273 (in Russian).

[6] Espresso. URL: http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/ (accessed 25.05.2016)

[7] Website of the Department of methodology of designing integrated circuits IPPM RAS. Available at: http://icdm.ippm.ru/ (accessed 25.05.2016) (in Russian).