ТЕХНИЧЕСКИЕ НАУКИ

УДК 004.04:519.6

Гранкин Виталий Владимирович, Мезенцева Оксана Станиславовна

ОБОСНОВАНИЕ КРИТЕРИЕВ ВЫБОРА ОСНОВАНИЙ СИСТЕМЫ ОСТАТОЧНЫХ КЛАССОВ ПРИ РЕАЛИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ УЗЛОВ НА ПЛИС

В данной работе рассмотрена проблематика выбора модулей СОК, предложен критерий потенциальных затрат для эффективной реализации на ПЛИС и на кристалле. Рассмотрены основные модулярные вычислители, а так же вычислители элементарных функций, проведена оценка логической сложности.

Ключевые слова: *модулярный код, система остаточных классов, кольцо вычетов, ПЛИС, элементарные функции.*

Vitaliy Grankin, Oksana Mezentseva

SOME SELECTION CRITERIAS FOR RADIXES IN RESIDUE NUMBER SYSTEM FOR EFFECTIVE IMPLEMENTATION ON FPGA

In this paper we consider the problems of selecting modules of RNS for the effective implementation in the FPGA and SoC. Proposed new criteria of potential waste of logical resources of die. The article describes basic modular calculators, as well as evaluators of elementary functions and their logical complexity.

Key words: *modular operations, residue number system, FPGA, elementary functions.*

На современном этапе развития информационных технологий особо значимой становиться проблема проведения параллельных вычислений. Данная проблема возникла в связи с физическими ограничениями роста быстродействия вычислительных устройств, упирающегося в невозможность эффективного повышения тактовой частоты вычислительных процессоров [1].

Часто во многих научных задачах требуется выполнить расчеты или произвести моделирование процесса с высокой точностью. Аппаратные средства ЭВМ позволяют непосредственно выполнять вычисления ограниченной точности. Для преодоления обозначенной трудности обычно используются длинная арифметика, точность при этом превышает размер разрядной сетки и ограничена только объемом доступной памяти и временем вычислений [2]. При этом время вычислений существенно зависит от размера чисел, над которыми производятся операции.

Приложение модулярной арифметики может быть одним из эффективных методов решения данной проблемы [3]. Система остаточных классов (Residue number system, RNS, СОК) является непозиционной системой представления чисел [4]. Числа в ней определяются кортежем разрядов, являющихся остатками от деления данного числа на модули системы – набор взаимно простых чисел, определяющих СОК, значение любого разряда не влияет на значения других. Благодаря отсутствию межразрядных переносов, модулярная арифметика позволяет произвести операции с числом в параллельных независимых каналах малой разрядности. Использование СОК может увеличить эффективность вычислений, в частности при реализации операций сложения, умножения, возведения в степень.

Операции сложения, вычитания, умножения называются модулярными и могут быть выполнены в кольце вычетов, при этом операции выполняются над каждым из остатков числа независимо и без учета переполнения. Модулярные операции имеют линейную асимптотику. Данные качества позволяют применять модулярную систему счисления на параллельных вычислительных устройствах и возможно ускорить традиционные последовательные операции.

Проблематика выбора оснований СОК. Во время проведения вычислений на ЭВМ как в модулярном коде, так и в традиционной системе счисления, необходимо, чтобы операнды, промежуточные значения и результат не выходили из возможного диапазона представления ЭВМ, то есть не превышали разрядную сетку вычислительного устройства (не происходило переполнение). Также могут быть губительны для результата процессы неучтённого исчезновения порядка (антипереполнение). Во избежание сказанного, нужно обеспечить вычислительный процесс с достаточным для задачи диапазоном изменения чисел и необходимой точностью. В двоичных ЭВМ это достигается увеличением разрядной сетки – количества бит представляющих переменную (мантиссу, порядок), в ЭВМ общего назначения, как было сказано, применением длинной арифметики.

В модулярной арифметике для увеличения диапазона и точности необходимо увеличить число $m_1 \cdot m_2 \cdot \ldots \cdot m_n = M$ – размер диапазона системы остаточных классов, где m_i – основания СОК. Увеличение диапазона M возможно путем выбора модулей, как большего размера, так и увеличением их количества. По требованию китайской теоремы об остатках, модули должны быть взаимно простыми [5]. Обычно в качестве оснований подбирают простые числа [6], но для выполнения модулярных операций, таких как умножение и сложение, данное условие не обязательно и для некоторых приложений, возможно, применять составные модули.

Для выбора модулей СОК необходимо исходить из критериев:

1) быстродействия;

2) аппаратных затрат;

3) требований применяемых алгоритмов модулярной арифметики;

4) требований точности и диапазона СОК.

Критерии быстродействия основываются на длине комбинационного пути, на фактической задержке сигналов при срабатывании логических элементов, числе независимых вычислительных каналов и степени их параллельности (коэффициента параллельности, отражающего модулярные и позиционные вычисления). Увеличение разрядности модулей увеличивает задержку срабатывания вычислительных узлов, так как растут задержки на межразрядные переносы, но по сравнению с ПСС данные задержки не являются критичными (при выборе сравнительно малых модулей) [3].

Критерий аппаратных затрат основывается на количестве элементарных блоков (логических элементов, транзисторов и др.), требуемых для построения вычислителей и таблиц, а также физического размера шины данных, с ростом количества модулей комбинационный путь не увеличивается, растут затраты на соединение модулей.

Требования применяемых алгоритмов модулярной арифметики накладывают ограничения на выбор оснований СОК. Так, используя составные множители, мы не сможем применить алгоритм индексного модулярного умножения [7], потому что для составных оснований не существует первообразного корня. При этом элементарный табличный метод умножения может быть применен для составных модулей.

Требования точности и диапазона СОК ставятся в зависимости от условий и ограничений задачи. Диапазон должен быть выбран таким образом, чтобы не происходило неучтенное переполнение на всех этапах вычислительного процесса, также должна быть учтена возможная потеря точности из-за недостаточной разрядности вычислительного процесса (антипереполнение).

Потенциальная потеря как критерий выбора модулей СОК. Для оценки возможных аппаратных затрат как по числу логических элементов, так и по сложности размещения шин данных можно ввести термин потенциальной потери.

Потенциальная потеря вычисляется по формуле:

$$\varepsilon_{m_i} = 2^{\lceil \log_2 m_i \rceil} - m_i \qquad (1)$$

где m_i – модуль СОК, $\lceil \ \rceil$ – округление вверх.

Потенциальная потеря (1) показывает, насколько неэффективен выбранный модуль СОК в сравнении с двоичной арифметикой. Другими словами ε_{m_i} – это разница между модулем СОК и числом бит требуемых для представления этого модуля. Так, выбрав модуль m = 17, мы получим потерю

значений $\varepsilon_m(17)=15$, то есть для модуля m = 17 необходимо использовать вычислительный канал шириной в 5 бит и диапазон $[0,16]$, а в позиционной системе счисления для такой ширины соответствует диапазон $[0,31]$.

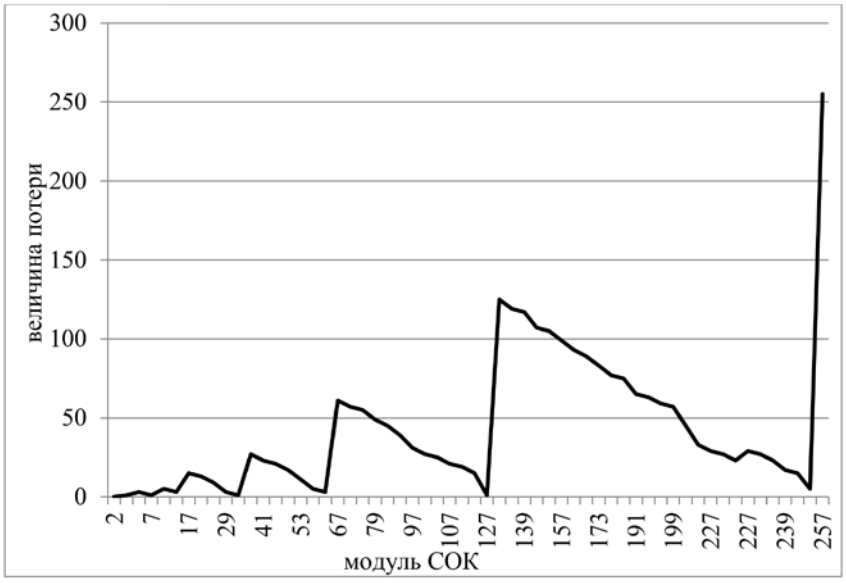

Рис. 1. Оценка зависимости потенциальной потери ε_m простыми модулями СОК при использовании двоичной элементарной базы

Как видно из графика (рис. 1), наиболее предпочтительны модули «близкие слева» к числам 2^n, например числа Мерсена 2^n-1, однако применение только их невозможно из-за их быстрого роста. Потенциальная потеря значений значительно увеличивается с ростом разрядности и для аппаратной реализации желательно выбирать меньшие модули.

Таблица 1

Потенциальная потеря значений для некоторых простых модулей (2, 3, 5, 7, 11, 13, 17, 31)

Модуль СОК	Бит на модуль	Бит необходимо	Число в ПСС	Потеря значений
2	1	1	2	0
3	1,5849625	2	4	1
5	2,3219281	3	8	3
7	2,8073549	3	8	1
11	3,4594316	4	16	5
13	3,7004397	4	16	3
17	4,0874628	5	32	15
31	4,9541963	5	32	1

При выборе небольших модулей можно также уменьшить накладные расходы на хранение таблиц преобразований, так как $m_1 \cdot m_2 \cdot \ldots \cdot m_n \geq m_1 + m_2 + \ldots + m_n, m > 1$, тем самым уменьшив число элементов вычислительного блока и занимаемой памяти.

При использовании составных модулей можно добиться меньших потенциальных потерь, например, взяв модуль $3 \cdot 3 \cdot 7 = 63$ получим $\varepsilon_m(63)=1$, причем сумма потерь $\varepsilon_m(3)+\varepsilon_m(3)+\varepsilon_m(7)=3$. Составные модули использовались в некоторых модулярных ЭВМ.

Таблица 2

Потенциальная потери значений модулей ЭВМ Т-340А и К 340А (главные конструкторы И. Я. Акушский, Д. И. Юдицкий) [8]

Модуль СОК	Бит на модуль	Бит в ПСС	Число в ПСС	Потеря значений
2	1	1	2	0
5	2,32	3	8	3
13	3,7	4	16	3
17	4,08	5	32	15
19	4,24	5	32	13
23	4,52	5	32	9
29	4,85	5	32	3
31	4,95	5	32	1
61	5,93	6	64	3
63	5,97	6	64	1

В таблице 2 указаны потенциальные потери модулей ЭВМ Т-340А и К 340А. Диапазон СОК данного набора модулей составляет 3336597244890, что соответствует двоичной ПСС с разрядностью 42 бит. Используется составной модуль $3 \cdot 3 \cdot 3 = 27$, уменьшающий аппаратную сложность вычислительной системы по сравнению с простыми модулями. Так, для представления данной СОК требуется 45 бит, для двоичной ПСС 42 бит. Для представления только лишь простых модулей 48 бит, для ПСС 43 бит (так как увеличился диапазон до 8222980095330). Причем увеличение затрат касается всех вычислительных узлов, начиная от сумматора и заканчивая таблицами преобразований.

Оценка логической сложности (количественная оценка логических элементов, составляющих модуль) вычислительных узлов и в целом микросхемы является достаточно сложной задачей [9]. Одним из препятствий оценки является широта выбора реализации вычислительного узла. Двоичный сумматор может быть реализован как по последовательной схеме, так и по параллельной (с последовательным переносом, с параллельным переносом). Сумматор может быть выполнен в виде комбинационной схемы полного сумматора или с применением таблиц поиска (LUT-таблицы на ПЛИС). Модулярный сумматор может быть реализован как таблица поиска (с применением LUT-таблиц), содержать несколько двоичных сумматоров. Некоторые блоки или их элементы могут не «раскладываться» в логические элементы и могут быть усечены (так как не имеют логической нагрузки и не требуются для физической реализации). Элементы шифраторов дешифраторов могут быть упрощены до проводников или усечены из реализации. Также в процессе оптимизации топологии модуля для согласования задержек могут быть добавлены дополнительные элементы (блоки ПЛИС используются как проводники, LUTs as wire).

Оценка затрат ресурсов ПЛИС осложнена архитектурными особенностями применяемого чипа, так блоки (slices, macrocells) FPGA могут содержать: таблицы поиска (LUT), память (ram), триггеры, сумматор, логические элементы, мультиплексор, умножитель и другие. Могут быть введены и дополнительные накладные расходы на временную оптимизацию проекта, с целью совмещения задержек различных модулей. Оптимизатор проекта ПЛИС должен учесть задержку переключения блока (slices, macrocells) и задержку распространения сигнала между блоками и выводами микрочипа.

Влияние размера и количества модулей на аппаратные затраты зависит от реализуемого устройства. Так, на регистры и каналы данных (шины) влияет только суммарная разрядность вычислительного тракта, оцениваемая как $\sum\lceil \log_2 m_i \rceil$ – минимальная разрядность шины для набора модулей m_i.

Оценка логической сложности модулярного сумматора зависит от типа его реализации. Полная тривиальная табличная реализация «на кристалле» (без применения запоминающих устройств) потребует в худшем случае k логических элементов:

$$k = \sum_i \left(2\log_2 \left\lceil m_i \right\rceil - 1\right) \cdot m_i^2, \tag{2}$$

где k – число логических элементов (логическая сложность), m_i – модуль СОК.

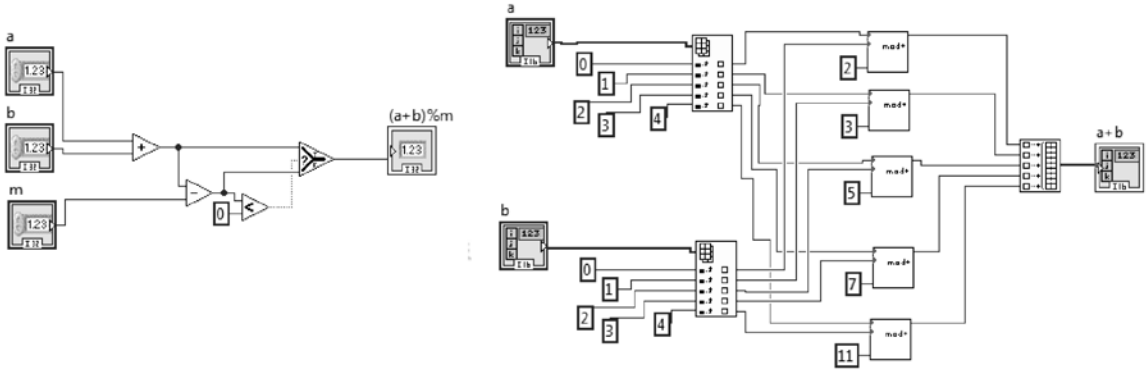

Рис. 2. Реализация сумматора в среде LabView, слева одномодульный сумматор, справа – сумматор для модулей (2, 3, 5, 7, 11)

На рис. 2 изображена схема аппаратной реализации сумматора с применением двоичных полных сумматоров.

Реализация с применением двоичных полных сумматоров потребует k логических элементов:

$$k = 17 \cdot \sum_i \log_2 \left\lceil m_i \right\rceil, \tag{3}$$

где k – число логических элементов, m_i – модуль СОК.

Асимптотики (2) и (3) показывают значительное превосходство реализации на сумматорах (рис. 2).

Данный сумматор (рис. 2) реализован для модулей (2, 3, 5, 7, 11) в среде National Instuments LabView для ПЛИС Spartan 3E500. Сумматор построен по схеме последовательных полных двоичных сумматоров с выбором результата по биту переполнения. Данный тип сумматора также был реализован на языке Verilog в среде Xilinx ISE Design Suite14.7.

Задержка для сумматора такого типа при применении модулей большой разрядности несколько повышается из-за межразрядных переносов, причем рост комбинационного пути сигнала составляет 4 логических элемента на один дополнительный разряд (на два элемента в двоичном сумматоре). Таким образом, для достижения большой тактовой частоты необходимо применять модули малой и средней разрядности.

Таблица 3

Затраты ресурсов ПЛИС Spartan 3 500e при реализации модулярных сумматоров с наборами модулей большой и малой разрядностью

Набор модулей СОК	Таблиц поиска, LUT	Таблиц поиска как проводников, LUT as a route-thru	Всего таблиц поиска LUT на сумматор	Ячеек ПЛИС, occupied SLIECES	Всего задействовано LUT
в проекте					
31, 53, 59, 61, 63, 127, 251	105	47	152	75	1696
2147483647, 8191	76	60	136	65	1714

В таблице 3 указаны значения использования ресурсов ПЛИС Spartan 3 500e при реализации модулярного сумматора с модулями малой разрядности (31, 53, 59, 61, 63, 127, 251) и большой разрядности (2147483647, 8191), всего 44 бит на каждый набор модулей. Как видно из таблицы 3, сумматор с модулями малой разрядности требует несколько больше ресурсов ПЛИС для своей реализации, по сравнению с сумматором большой разрядностью. Данная закономерность обусловлена тем, что в ПЛИС Spartan 3 сумматоры представлены таблицами поиска LUTs с 4 входами и происходят потери кратности модулей СОК.

Оценка логической сложности модулярного умножителя аналогична оценке для модулярного сумматора. Логическая сложность полной табличной реализации «на кристалле» (без применения запоминающих устройств) оценивается по формуле (3). При реализации умножителей применение модулей большой разрядности в большинстве случаев не оправдано. В случае применения табличных умножителей увеличиваются затраты на хранение-представление таблиц.

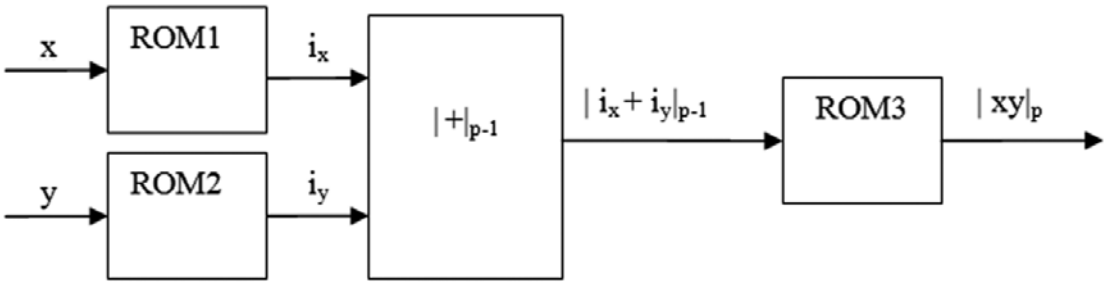

Рис. 3. Схема индексного модулярного умножителя (fast RNS Galois field multiplier)

На рис. 3 изображена схема индексного модулярного умножителя [7], для реализации умножителя такого типа с набором модулей (2147483647, 8191) потребуется хранить таблицы размером 32 Гб (без учета оптимизации). В свою очередь набор модулей (31, 53, 59, 61, 63, 127, 251) требует только 2212 байт. Реализация тривиального умножителя (умножение и последующее вычисление остатка) не рассматривается как эффективный метод для ПЛИС и «на кристалле» в связи с их аппаратными затратами и временем исполнения.

Оценка логической сложности модулярного вычислителя элементарных функций. В статье [10] описаны и реализованы табличный и полиномиальный методы вычисления элементарных функций в модулярном коде.

Вычислитель полинома может быть построен как многотактный узел с одним модулярным умножителем и одним модулярным сумматором. Случай с конвейерной реализацией (последовательным размещением сумматоров и умножителей) не рассматривается ввиду значительного потребления площади «кристалла».

Применение модулей большой разрядности для данного метода (рис. 4) не оправдано. При реализации полиномиального вычислителя, значительные ресурсы ПЛИС затрачиваются на размещение умножителя.

Логическая сложность полиномиального вычислителя является суммой от сложности сумматора (2), умножителя (3) и логической сложности вспомогательных регистров:

$$k = \sum_i \left(2\log_2\left\lceil m_i\right\rceil - 1\right)\cdot m_i^2 + \sum_i \left(5\cdot 2 + 5 + 3\right)\cdot \log_2\left\lceil m_i\right\rceil + s\cdot \log_2\left\lceil m_i\right\rceil, \quad (4)$$

где k – число логических элементов (логическая сложность) полиномиального вычислителя, m_i – модуль СОК, s – накладные расходы на реализацию вспомогательных регистров (на регистр «защелку» входа, временный регистр и т. д.).

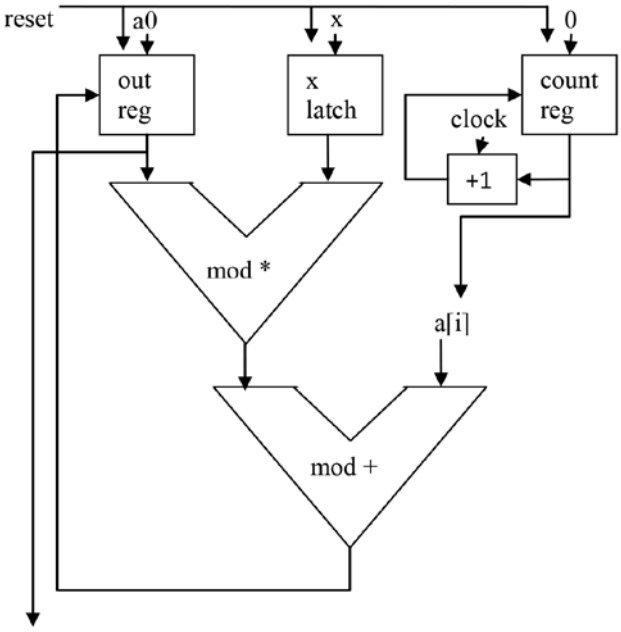

Рис. 4. Схема реализации модулярного вычислителя значения полинома

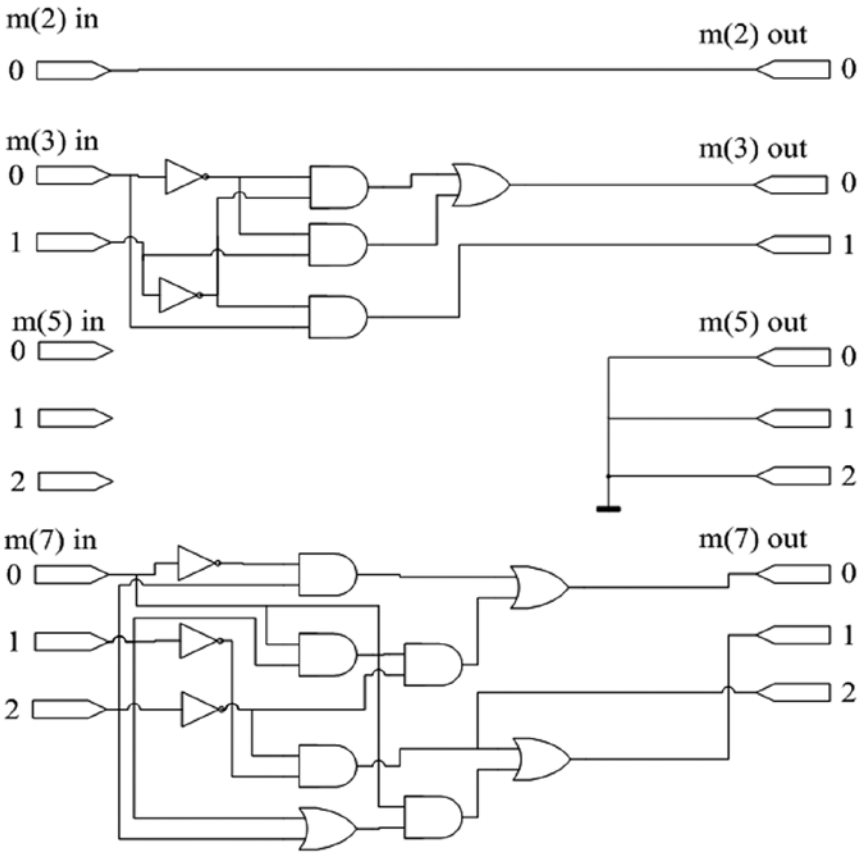

Рис. 5. комбинационная схема модулярного табличного вычислителя экспоненты

При реализации табличного вычислителя элементарных функций, с ростом величины модуля значительно растут размеры таблиц. Причем асимптотика роста составит

$$k = \sum_i \left(\log_2 \left\lceil m_i \right\rceil + 1\right) \cdot \left(m_i + 1\right) \quad (5)$$

где k – оценка сверху общей логической сложности табличного модулярного вычислителя элементарных функций, m_i – модуль СОК. Оценка (5) является теоретической, при применении методов оптимизации можно добиться значительного уменьшения потребления ресурсов.

На рис. 5 изображена комбинационная схема табличного вычисления значения экспоненты в СОК с набором модулей (2, 3, 5, 7, 11), схема для модуля 11 не приведена для экономии места. Данная схема была оптимизирована и построена без применения средств проектирования электронных устройств (при применении схема будет иметь другой вид). Можно видеть, что для модуля 5 отсутствуют логические элементы, вычислительный тракт для этого модуля может быть полностью усечен до последующих вычислительных узлов; для модуля 7 удалены повторяющиеся элементы.

При проведении мер по логической оптимизации можно получить, более эффективную реализацию, для схемы на рис. 5 практическая асимптотика составляет:

$$k = \sum_i \left(2\log_2 \left\lceil m_i \right\rceil + 1\right) \quad (6)$$

где k – оценка сверху общей логической сложности табличного модулярного вычислителя элементарных функций, m_i – модуль СОК. Асимптотика (6) не может быть применена для других таблиц (может быть различной для других элементарных функций).

Заключение. Предложен критерий выбора модулей СОК на основании потенциальных аппаратных затрат (потенциальная потеря значений). На основании данного критерия можно делать выводы о целесообразности применения конкретного модуля и набора модулей при реализации элементов модулярной арифметики на ПЛИС и на «кристалле».

Разработаны схемы вычислительных устройств модулярной арифметики, таких как сумматор, умножитель, вычислитель элементарных функций.

Проведена оценка логической сложности некоторых вычислительных узлов модулярной арифметики, в том числе: сумматора (вычитателя), умножителя, вычислителя элементарных функций (полиномиального и табличного).

Показано, что при аппаратной реализации основных вычислительных устройств СОК в особенности табличных вычислителей, предпочтительно использовать модули малой разрядности. При этом необходимо выбирать модули близкие к своему двоичному представлению слева (по критерию потенциальных потерь).

Показано, что при применении логической оптимизации табличного вычислителя можно добиться значительного уменьшения потребления аппаратных ресурсов.

Литература

1. Zhirnov Victor V., Cavin III Ralph K., Hutchby James A., Bourianoff George I. Limits to Binary Logic Switch Scaling – A Gedanken Model // Proceedings Of The IEEE, VOL. 91, № 11. 2003. С. 1.

2. Оцоков Ш. А. Алгоритмическая и структурная организация высокопроизводительных ЭВМ с использованием модели безошибочных вычислений: дис. … канд. техн. наук: 05.13.15 / Ш. А. Оцоков. М., 2003. С. 13, 264.

3. Червяков Н. И. Реализация высокоэффективной модулярной цифровой обработки сигналов на основе программируемых логических интегральных схем // Нейрокомпьютеры: разработка и применение. 2006. № 10. С. 24–36.

4. Omondi A., Premkumar B. Residue number systems. Theory and Implementation // London: Imperial College Press, 2007. С. 258.

5. Ноден П., Ките К. Алгоритмическая алгоритмика. М.: Мир, 1999. С. 720.

6. Акушский И. Я., Юдицкий Д. М. Машинная арифметика в остаточных классах. М.: Советское радио, 1968. С. 440.

7. Damu Radhakrishnan. A fast RNS galois field multiplier // Yong Yuan Department. Department of Electronic English, Idaho University Moscow, ID DOI: 10.1109/ISCAS.1990.112619. Circuits and Systems, IEEE International Symposium on Source: IEEE Xplore. 1990.

8. Малашевич Б. М., Малашевич Д. Б. Отечественные модулярные и троичные ЭВМ // 50 лет модулярной арифметике. Зеленоград, 2005. С. 107.

9. Угрюмов Е. П. Цифровая схемотехника: учебное пособие для вузов. 2 е издание. СПб.: БХВ-Петербург, 2005. С. 634.

10. Гранкин В. В., Мезенцева О. С. К вопросу о реализации модулярных вычислителей элементарных функций в среде LabVIEW и их реализации на ПЛИС // Материалы VI Международной научно-технической конференции «Инфокоммуникационные технологии в науке, производстве и образовании». Ставрополь: СевКавГТИ, 2014. С. 248–254.

УДК 621.382:621.315.592

Девицкий Олег Васильевич, Сысоев Игорь Александрович

МОДЕЛИРОВАНИЕ СОЛНЕЧНОГО ЭЛЕМЕНТА НА ОСНОВЕ ГЕТЕРОСТРУКТУР ALXGA1-XAS – INXGA1-XAS – GAAS[1]

В программе Silvaco TCAD выполнено моделирование электрических параметров солнечных элементов на основе AlxGa1-xAs – InxGa1-xAs – GaAs в условиях AM 1.5 при различных значениях параметра x. Показано, что солнечные элементы на основе AlxGa1-xAs – InxGa1-xAs – GaAs потенциально имеют большую эффективность по сравнению с эффективностью обычных солнечных элементов.

Ключевые слова: *наногетероструктуры, фотовольтаика, солнечная энергетика, фотоэлектрический преобразователь, солнечный элемент, Silvaco TCAD, моделирование солнечных элементов.*

Oleg Devitsky, Igor Sysoev

SIMULATION OF SOLAR CELLS BASED ON HETEROSTRUCTURES AlxGa1-xAs – InxGa1-xAs – GaAs

The program Silvaco TCAD simulated electrical parameters of solar cells based on AlxGa1-xAs – InxGa1-xAs – GaAs under AM 1.5 for different values of x. It is shown that the solar cells on the basis of AlxGa1-xAs – InxGa1-xAs – GaAs may be quite higher efficiency compared to the efficiency of conventional solar cells.

Key words: *heterostructure, photovoltaics, solar energy, photoelectric converter, solar cell, Silvaco TCAD, modeling of solar cells.*

В настоящее время исследованию каскадных солнечных элементов на основе полупроводниковых гетероструктур A3B5 посвящено большое количество работ, на многих отечественных и зарубежных конференциях по фотовольтаике таким исследованиям уделяется достаточно большое внимание. В большинстве этих работ красной нитью проходит идея расширения области поглощения света за счет использования узкозонных вставок в виде квантовых ям или многопереходных гетероструктур для увеличения фототока. Распространено мнение, что наиболее сильное увеличение фототока и соответственно эффективности элемента возможно при создании в его базовой области сильного электрического поля, в соответствии с этим моделируются солнечные элементы со структурой $p - i - n$. Нами же будет рассмотрен вопрос, затрагивающий оптимально возможные условия, при каких возможно достичь максимальной эффективности многопереходного солнечного элемента на основе

1 Работа выполнена при финансовой поддержке Минобрнауки России в рамках государственного задания по проекту №2014/216, код проекта: 2516.