Инженерный вестник Дона, №3 (2018)
ivdon.ru/ru/magazine/archive/n3y2018/5042

Аппаратная реализация операции нахождения остатка от деления для входных данных большой разрядности на основе блоков редукции и коррекции

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

В.С. Рухлов, А.С. Михмель

Институт проблем проектирования в микроэлектронике РАН, Москва

Аннотация: Операция нахождения остатка от деления – это арифметическая операция, играющая большую роль в теории чисел. Важнейшую роль эта операция имеет при проектировании устройств, использующих, модулярную арифметику. Модулярная арифметика обладает высоким параллелизмом и часто используется с данными большой размерности, для ускорения вычислений. При работе с модулярной арифметикой, зачастую используются данные больших размерностей (порядка 128-512 бит). Для нахождения их модулярного представления требуется эффективный способ найти остаток от деления многоразрядных чисел на соответствующий набор модулярных базисов. В статье исследуются варианты построения устройств нахождения остатка от деления, для больших размерностей входных данных, в которых делитель p - является константой и известен на этапе проектирования устройства. Исследуются варианты реализации таких устройств, определяются их оптимальные параметры и проводится сравнение задержки и площади по сравнению с Verilog операцией «%» как при синтезе в СБИС, так и в ПЛИС.
Ключевые слова: модулярная арифметика, система остаточных классов, вычет, остаток от деления

Введение

Операция нахождения остатка от деления – это арифметическая операция, играющая большую роль в теории чисел. Обычно эта операция определяется для целых чисел следующим образом. Пусть и Y и p — целые числа, причём $p \neq 0$. Деление с остатком Y («делимого») на p («делитель» или «модуль») означает нахождение такого целого числа X, что выполняется равенство: $Y = p \cdot q + X$, где $0 \leq X < |p|$. Нахождения остатка от деления в литературе также записывают следующим образом $X = Y \bmod p$ или $X = |Y|_p$.

При проектировании микроэлектронных устройств эта операция встречается в языках описания аппаратуры Verilog и VHDL. Она обозначается символом % [1]. Важнейшую роль эта операция имеет при

проектировании устройств, использующих модулярную арифметику [2]-[4]. Модулярная арифметика обладает высоким параллелизмом и часто используется с данными большой размерности, для ускорения вычислений [5]. Существуют разные подходы для формирования модулярного базиса. Один заключается в использовании специальных модулей вида $p = 2^n \pm 1, 2^n \pm k$, дающих преимущества при проектировании операций модулярной арифметики [6]-[8]. Другой подход заключается в использовании большого числа модулей малой размерности (порядка 10-12 бит), особенно для больших размерностей входных данных (порядка 512 бит), где использовать специальные модули затруднительно [9]. Для произвольных модулей, требуется использовать некоторое универсальное решение, чтобы реализовать прямой преобразователь из позиционной системы счисления в модулярный код. Как вариант, можно использовать встроенную в САПР операцию «%». Однако это операция работает медленно и устройства, которые её используют, занимают много площади на кристалле. В данной статье предлагается реализация нахождения остатка от деления для произвольного модуля p, которая существенно превосходит по параметрам встроенную в САПРы операцию. Исследуются варианты построения устройств нахождения остатка от деления, для больших размерностей входных данных, в которых делитель p - является константой и известен на этапе проектирования устройства. Предложены варианты реализации таких устройств, определяются их оптимальные параметры и проводится сравнение задержки и площади по сравнению с Verilog операцией «%» как при синтезе в СБИС, так и в ПЛИС.

Аппаратная реализация нахождения остатка от деления

Пусть задано входное число X в позиционном виде разрядности N бит. Требуется реализовать микроэлектронное устройство выполняющее

преобразование из позиционного представления в модулярный код. Будем вести разработку устройства на HDL языке Verilog. Устройство должно находить остаток от деления входного числа на каждое число из набора, образующих модулярный базис $p_1, p_2, \ldots, p_m$. Так как алгоритм вычисления остатков будет одинаков для каждого из них, то рассмотрим произвольное число из набора *p* размерности *k* бит (рис. 1).

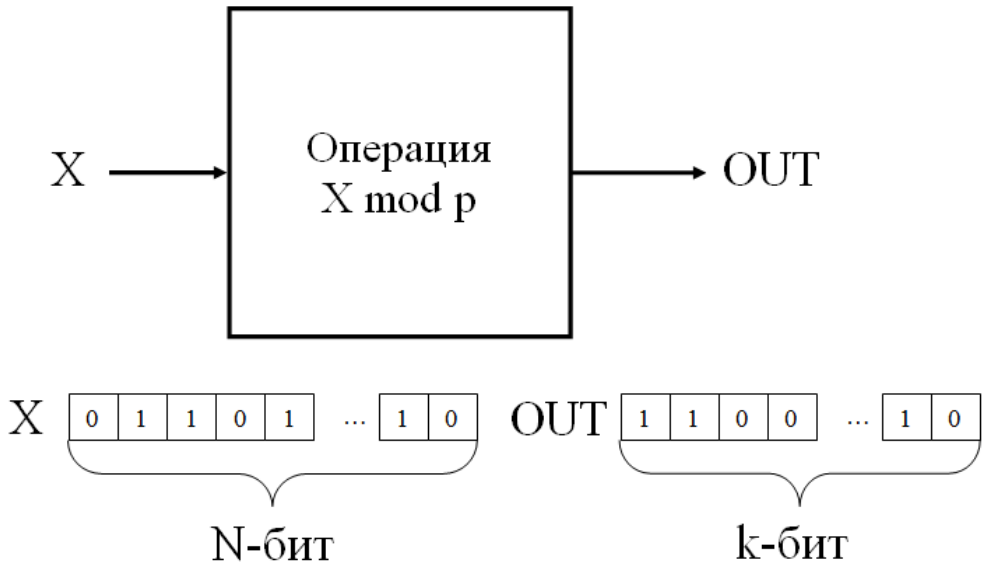

Рис. 1. – Входные и выходные данные прямого преобразователя

Параметры устройства характеризуются следующими условиями:

$0 \le X \le 2^N, 0 \le OUT \le p$

p - константа, заданная на этапе проектирования, то есть не меняющаяся во время функционирования устройства. При некоторых соотношениях между входными данными и заданным *p*, модуль нахождения остатка от деления можно реализовать простейшим образом. Рассмотрим такие специальные случаи.

Специальный случай 1. Модуль больше входных данных

Если $p > 2^N$, то в этом случае выходные данные совпадают с входными данными, то есть $OUT = X$.

```
module forward_conv_module_23 (in, out);
    input [3:0] in; // Max value: 15
    output [4:0] out; // Max value: 22
    assign out = {1'd0,in[3:0]};
endmodule
```

Специальный случай 2. Модуль равен степени два

Если $p = 2^t$, то в этом случае выходные данные равны младшим t-бит входных данных, то есть $OUT = X[t:0]$.

```
module forward_conv_module_16 (out0, in);
    output [3:0] out0; // Max value: 15
    input [7:0] in; // Max value: 255
    assign out0 = in[3:0];
endmodule
```

Специальный случай 3. N незначительно превосходит разрядность модуля

В случае если N незначительно превосходит разрядность модуля, достаточно проверить в какой из диапазонов вида $[0;p), [p;2 \cdot p), [2 \cdot p;3 \cdot p), \ldots$ попадает входное число X и вычесть из него поправочный коэффициент, соответствующий диапазону. 0 для $[0;p)$, p для $[p;2 \cdot p)$, $2 \cdot p$ для $[2 \cdot p;3 \cdot p)$ и т.д.

```
module mod_253_511 (in, out);
```

```
    input [8:0] in;
    output reg [7:0] out;
    always @ (in)
    begin
        if (in < 8'd253)
        begin
            out = in;
        end
        else if (in < 9'd506)
        begin
            out = in - 8'd253;
        end
        else
        begin
            out = in - 9'd506;
        end
    end
endmodule

module forward_conv_module_253 (in, out);
    input [8:0] in; // Max value: 511
    output [7:0] out; // Max value: 252
    mod_253_511 inst(in, out);
endmodule
```

Решение для общего случая

Представим входные данные X в следующем виде (здесь запись $X[i]$ - означает i бит в двоичной записи числа):

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

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:

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

Константы вида $A_i = |2^i|_p$ могут быть рассчитаны на этапе проектирования устройства и каждая константа не превосходит значение p. А общее значение выражения

$$SUM = (A_0 \cdot X[0] + \cdots + A_{N-1} X[N \quad 1]) \leq (p \quad 1) \cdot N \qquad (1)$$

Так как коэффициенты известны, то оценку сверху MAX_{SUM} для значения SUM можно сократить ещё больше:

$$MAX_{SUM} = A_0 + A_1 + \cdots + A_{N-1} \leq (p-1) \cdot N$$

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

1) Найти значение простого выражения SUM по формуле (1). Назовем аппаратный блок, который выполняет эту операцию «блок редукции» (Рис. 2)

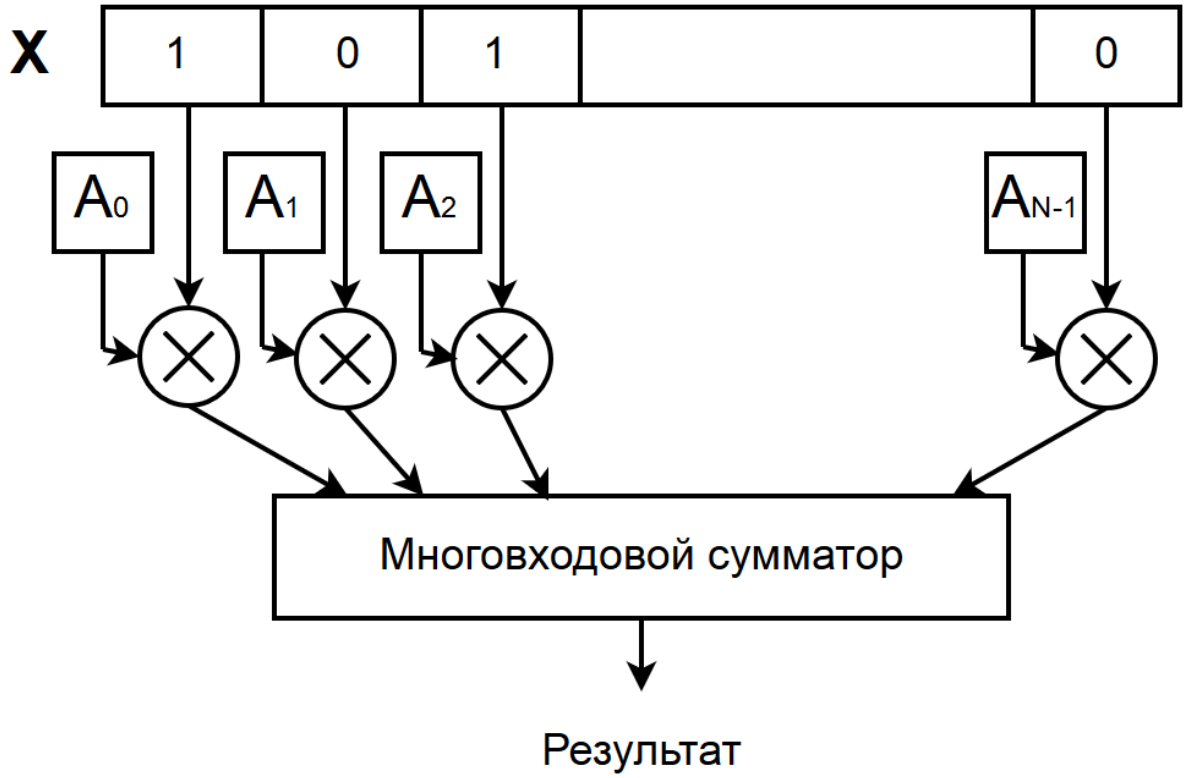

Рис. 2. – Входные и выходные данные прямого преобразователя

2) Определить в какой интервал вида $[0;p),[p;2\cdot p),[2\cdot p;3\cdot p),\dots,[(N-1)\cdot p;N\cdot p]$ попадает значение SUM и вычесть из него соответствующее значение 0 для $[0;p)$, p для $[p;2\cdot p)$, $2\cdot p$ для $[2\cdot p;3\cdot p)$ и.т.д. Назовем аппаратный блок, который выполняет эту операцию «блок коррекции» (Рис. 3)

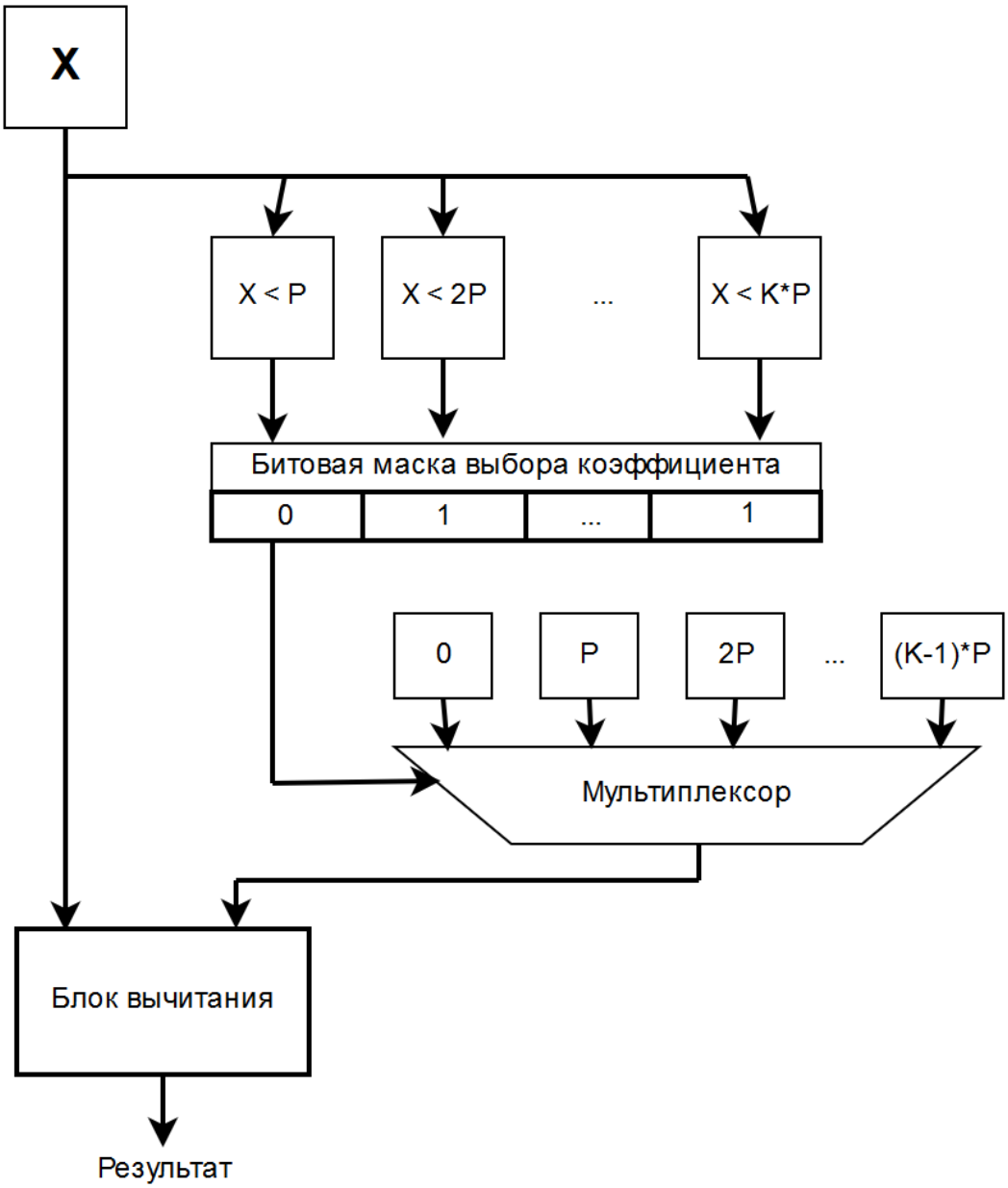

Рис. 3. – Блок коррекции

Модификации операции для нахождения остатка от деления для увеличения производительности устройства

Метод построения прямого преобразователя может быть модифицирован. В предыдущей главе разбивать исходное число предлагается на блоки по одному биту. Полученное после редукции число

сразу идёт на блок коррекции. Структуру операции для нахождения остатка от деления можно модифицировать по двум направлениям:

1) Выбирать блоки разбиения разного размера
2) Выполнять редукцию более одного раза

Формула для блоков разбиения размера 4 может быть расписана следующим образом:

$$|X|_p = \left|2^0 \cdot X[0:3] + 2^4 \cdot X[4:7] + \cdots + 2^{4k}[4k:4(k+1)-1]\right|_p$$

В общем случае для блока размерности T:

$$|X|_p = \left|2^0 \cdot X[0:T-1] + 2^T \cdot X[T:2T-1] + \cdots + 2^{Tk}[Tk:T(k+1)-1]\right|_p \quad (2)$$

Выбирать T следует таким образом, что бы размерность T не превышала размерность p. То есть необходимо гарантировать, что переменные $X[0:T-1]$ меньше p. В ином случае потребуется находить значение $|X[0:T-1]|_p$ либо увеличивать размер блока редукции, что негативно сказывается на производительности устройства. При выполнении этого условия можно переписать формулу (2) как:

$$|X|_p = \left||2^0|_p \cdot X[0:T-1] + |2^T|_p \cdot X[T:2T-1] + \cdots + \left|2^{Tk}\right|_p [Tk:T(k+1)-1]\right|_p = |M_1|_p \quad (3)$$

Значение M_1 внутри скобок не превосходит $p * 2^T * k$. Видно, что значение растет экспоненциально с ростом размерности блока и линейно относительно значения модуля и размерности входных данных. Здесь T, k и N связаны следующим соотношением: $T * k \geq N > T * (k-1)$.

Поскольку значение M_1 всё ещё может быть большим, особенно для большой размерности входных данных, то можно выполнить редукцию

диапазона ещё один раз по той же схеме что описана в формуле (3), но переменная k и число слагаемых в формуле будет меньше. Т.е. $|M_1|_p = \cdots = |M_2|_p$, где $M_1 \gg M_2$.

В данной работе проводится попытка выяснить, какие параметры оптимальны для проектирования аппаратных блоков для операции нахождения остатка от деления на практике и как их показатели отличаются от встроенных в современные САПР реализаций этой операций.

Экспериментальные результаты

Были проведены эксперименты с быстрыми блоками для операции нахождения остатка от деления (далее FFC) в системах Synopsys Design Compiler для заказных СБИС схем и Altera Quartus для ПЛИС. Сравнение проводилось с встроенной функцией синтеза операции остатка от деления (далее Default). Типовой модуль такого вида на языке Verilog выглядит следующим образом (приведен пример для модуля 23):

```
module mod_23 (in, out);
    input [16:0] in;
    output [4:0] out;
    assign out = in % 23
endmodule
```

Для генерации аппаратных описаний FFC было подготовлено программное обеспечение, которое свободно доступно онлайн на сайте ИППМ РАН [10].

Эксперимент 1. Исследование параметров FFC для различных входных размерностей и неспециального модуля малой размерности

В первом эксперименте проверялось, как работает FFC при разных значениях параметров «размерность блока разбиения» от 1 до 8 и «количество этапов редукции» 1 или 2. Модуль был выбран равным 43. Разрядность входных данных выбиралась из списка 64, 128, 256 и 512. Приведены данные расчетов в Synopsys на настройках compile_ultra и в одной из последних версий от 2014 года. Этот режим используется для очень сильной оптимизации схем. На рисунке 4 приведен сводный график задержек для всех размерностей входных данных. В таблице 1 приводится сравнение FFC с Default.

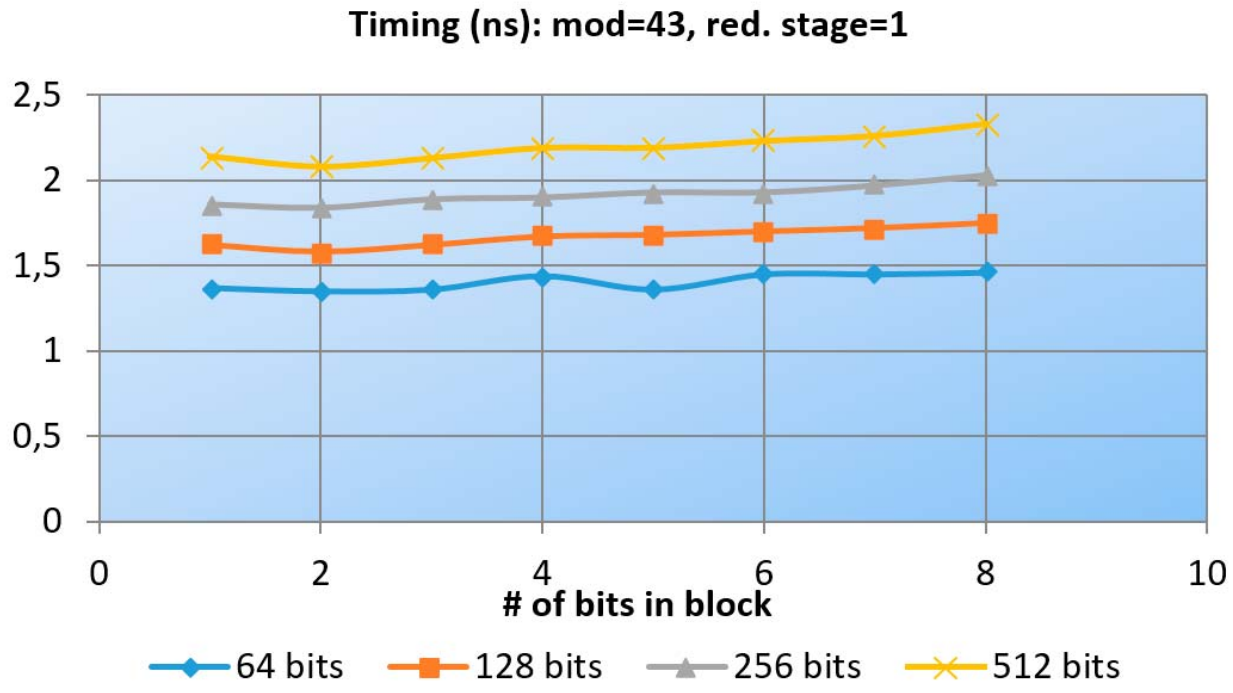

Рис. 4. – Сводный график задержки для разных размерностей входных данных (количество этапов редукции равно 1)

Таблица № 1

Сравнение Deafult и FFC прямых преобразователей для модуля 43

Размерность входных	**Задержка (нс)**			**Площадь (мкм2)**		
	FFC	Default	Ускорение	FFC	Default	Уменьшение

данных	(min delay)		(%)	(min area)		(раз)
64	1.35	1.25	**-8 %**	1345	4880	**~ 4 раза**
128	1.58	2.08	**24 %**	1997	22901	**~ 12 раз**
256	1.84	2.36	**22 %**	3086	77267	**~ 25 раз**
512	2.08	2.76	**25 %**	4752	269991	**~ 59 раз**

Сравнение со стандартной реализацией нахождения остатка от деления показывает, что использование FFC является более предпочтительным, как минимум из-за значительного сокращения площади. На больших размерностях входных данных так же удается сократить и задержку устройства примерно на 20%.

Эксперимент 2. Сравнение скорости работы и площади FFC и стандартного прямого преобразователя на ПЛИС

Во втором эксперименте проводился синтез FFC и Default в среде Altera Quartus (версия Prime Version 16.0.0 Build 211 04/27/2016 SJ Lite Edition). Размерность входных данных выбиралась из списка: 8, 16, 32. Выбор размерности определен тем фактом, что Altera может синтезировать операцию нахождения остатка от деления для размерности входных данных не более 64. Модуль был выбран равным 7. Результаты измерений приведены в таблицах 2 и 3.

Таблица № 2

Сравнение FFC и Default для разных размерностей входных данных

Размерность входных данных	Значение модуля	**Задержка**			**Площадь**		
		Default (нс)	FFC (нс)	Улучшение (в X раз)	Default (ячейки)	FFC (ячейки)	Улучшение (%)
8	7	19.9	10.4	~1.9 раз	71	23	~3.1 раз

16	7	37.5	15.9	~2.4 раз	167	48	~3.5 раз
32	7	75.3	15.3	~4.9 раз	359	87	~4.1 раз

Таблица № 3

Сравнение задержки и площади FFC при разных значениях параметров для модуля p=7 и размерности входных данных 8

Размер блока разбиения	**Задержка (нс)**	**Площадь (ячейки)**	**Задержка (нс)**	**Площадь (ячейки)**
	Число этапов редукции = 1		Число этапов редукции = 2	
1	10.8	28	11.1	30
2	10.5	26	10.9	25
3	10.9	25	11.7	25
4	11.0	25	12.3	25
5	11.0	25	11.4	28
6	11.8	37	11.7	35
7	10.5	59	13.5	48
8	**10.4**	**23**	**10.4**	**23**

Как видно из эксперимента на ПЛИС. Быстрый FFC оказался в несколько раз лучше, того который используется для синтеза операции нахождения остатка от деления в среде Altera Quartus. Вывод справедлив, как для задержки, так и для площади.

Эксперимент 3. Сравнение скорости работы и площади FFC для модуля большой размерности.

В третьем эксперименте проводился синтез FFC и Default в среде Altera Quartus (версия Prime Version 16.0.0 Build 211 04/27/2016 SJ Lite Edition) для ПЛИС Altera Cyclone IV E: EP4CE22F17C6 в режиме «speed grade 6» (core voltage 1.2V, and temperature 85 Celsius). В этом эксперименте исследовалась большая размерность входных данных и 12-битный модуль. Размерность входных данных выбиралась из списка: 64, 128 (для входных данных большей размерности в большинстве случаев устройство не удавалось синтезировать). Модуль был выбран равным 2999, чтобы быть на достаточном расстоянии от степеней двойки: 2048 и 4096.

Для стандартного преобразователя были посчитаны задержка и площадь для размерности 64 бита и модуля 2999.

Задержка: 196.2 нс

Площадь: 2066 ЛЕ

Результаты для FFC приведены в таблице 4. Там, где в таблице значение «-», параметр не удалось посчитать из-за проблем с синтезом.

Таблица № 4

Результаты расчетов FFC в ПЛИС

	Задержка, нс				**Площадь, ЛЭ**			
Разрядность	64 бита		128 бит		64 бита		128 бит	
Этапов редукции	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**
1	37.59	37.56	44.97	**39.26**	1319	711	2798	1373
2	36.53	**32.01**	49.73	39.90	1570	**679**	3510	1371
3	37.83	34.56	45.88	35.27	1989	768	4500	**1335**
4	44.35	36.42	59.74	40.35	3380	786	6746	1489
5	49.48	40.55	81.81	45.87	5407	1529	12240	2145

6	55.53	39.70	95.08	39.90	6488	1181	15195	1780
7	76.80	45.72	-	-	11488	2243	-	-
8	-	-	-	-	25472	2452	-	-

Из таблицы 4 видно, что оптимальный 64-битный FFC имеет 2 этапа редукции и размерность блока разбиения 2. Задержка меньше, чем у стандартного (Default) более чем в 6 раз, а площадь (число логических ячеек) меньше в 3 раза.

Заключение

Можно сделать следующие выводы из проведённых экспериментов:

- В случае синтезирования операции *X%p*, где *p* - константа современные САПР не применяют специальных методов, которые позволяют синтезировать оптимальные устройства такого вида. Что выливается в чрезмерные затраты по площади устройства, а в случае ПЛИС ещё и в очень большой задержке. Поэтому вместо операции “%” лучше использовать предложенные в статье устройства. Выигрыш для СБИС по скорости около 20% и по площади в 4-60 раз. Для ПЛИС выигрыш по скорости в 2-5 раз, по площади в 3-4 раза.
- Операцию нахождения остатка от деления можно реализовывать разными способами используя различные блоки разбиения и разное число блоков редукции. В СБИС и ПЛИС оптимальные параметры различаются.
- Для СБИС оптимальный размер блока разбиения варьируется, но обычно наиболее стабильный размер блока = 1, который дает результат близкий к оптимальному. Количества блоков редукции следует выбирать в зависимости от целей – минимизировать

площадь или минимизировать задержку. Для задержки оптимально уменьшать число блоков редукции, для площади наоборот увеличивать.

- Для ПЛИС на больших размерностях входных данных лучше использовать два блока редукции и небольшой размер блока разбиения (1-2).

Работа выполнялась при поддержке Российского фонда фундаментальных исследований, грант № 17-07-00404 А.

Литература

1. IEEE Standards Association et al. IEEE Standard for Verilog Hardware Description Language // Design Automation Standards Committee, IEEE Std 1364TM-2005. – 2005. – V. 2. pp. 45-47.

2. Эрдниева, Н. С. Использование системы остаточных классов для маломощных приложений цифровой обработки сигналов. // Инженерный вестник Дона, 2013, № 2. URL: ivdon.ru/ru/magazine/archive/n2y2013/1621.

3. Соловьев Р. А., Тельпухов Д. В. Аппаратная реализация операции нахождения остатка целочисленного деления для входных данных большой разрядности в модулярной арифметике // Известия высших учебных заведений. Электроника. – 2013. – №. 4. – С. 75-83.

4. Тельпухов Д. В., Михмель А. С., Соловьев Р. А. Проектирование энергоэффективных модулярных КИХ фильтров на базе редуцированных мультиконстантных умножителей. // Инженерный вестник Дона, 2017, № 4. URL: ivdon.ru/ru/magazine/archive/n4y2017/4535

5. Инютин С. А. Вычислительные задачи большой алгоритмической сложности и модулярная арифметика // Вестник Тюменского государственного университета. – 2002. – №. 3. – С. 3-10.

6. Gallaher D., Petry F. E., Srinivasan P. The digit parallel method for fast RNS to weighted number system conversion for specific moduli (2^k-1, 2^k, 2^k+

1) // Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on. – 1997. – V. 44. – №. 1. – pp. 53-57.

7. Е.С. Балака, И.С. Рухлов, А.Н. Щелоков, Р.А. Соловьев. Разработка и синтез сумматоров по модулям 2^n±1 с параллельным переносом // Труды конгресса по интеллектуальным системам и информационным технологиям. 2016. Т. 1, С. 159-164

8. Hiasat A. A Residue-to-Binary Converter for the Extended Four-Moduli Set {2^{n}-1, 2^{n}+ 1, 2^{2n}+ 1, 2^{2n+ p}} // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2017. – V. 25. – №. 7. – pp. 2188-2192.

9. Molahosseini, A. S., Sorouri, S., & Zarandi, A. A. E. (2012, July). Research challenges in next-generation residue number system architectures. In Computer Science & Education (ICCSE), 2012 7th International Conference on (pp. 1658-1661). IEEE.

10. Генератор Verilog универсального прямого преобразователя для произвольных модулей. URL: icdm.ippm.ru/2015/forward-converter-universal.php

References

1. IEEE Standards Association et al. IEEE Standard for Verilog Hardware Description Language. Design Automation Standards Committee, IEEE Std 1364TM-2005. 2005. V. 2. pp. 45-47.

2. Erdnieva, N. S. Inženernyj vestnik Dona (Rus), 2013, № 2. URL: ivdon.ru/ru/magazine/archive/n2y2013/1621.

3. Solov'ev R. A., Tel'pukhov D. V. Izvestiya vysshikh uchebnykh zavedeniy. Elektronika. 2013. №4. pp. 75-83.

4. Tel'pukhov D. V., Mikhmel' A. S., Solov'ev R. A. Inženernyj vestnik Dona (Rus), 2017, №4. URL: ivdon.ru/ru/magazine/archive/n4y2017/4535

5. Inyutin S. A. Vestnik Tyumenskogo gosudarstvennogo universiteta. 2002. №3 pp. 3-10.

6. Gallaher D., Petry F. E., Srinivasan P. The digit parallel method for fast RNS to weighted number system conversion for specific moduli (2^k-1, 2^k, 2^k+ 1). Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on. 1997. V. 44. №1. pp. 53-57.

7. E.S. Balaka, I.S. Rukhlov, A.N. Shchelokov, R.A. Solov'ev. Trudy kongressa po intellektual'nym sistemam i informatsionnym tekhnologiyam. 2016. V. 1, pp. 159-164

8. Hiasat A. A Residue-to-Binary Converter for the Extended Four-Moduli Set {2^{n}-1, 2^{n}+ 1, 2^{2n}+ 1, 2^{2n+ p}}. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2017. V. 25. №7 pp. 2188-2192.

9. Molahosseini, A. S., Sorouri, S., & Zarandi, A. A. E. (2012, July). Research challenges in next-generation residue number system architectures. In Computer Science & Education (ICCSE), 2012 7th International Conference on (pp. 1658-1661). IEEE.

10. Generator Verilog universal'nogo pryamogo preobrazovatelya dlya proizvol'nykh moduley. URL: icdm.ippm.ru/2015/forward-converter-universal.php