# Отложенный перенос

## Операция умножения в поле чисел

ООО «Сайфер ЛТД», к.т.н. Влад Ковтун
Андрей Охрименко

# Содержание

- Актуальность
- Существующие алгоритмы
- Общее описание подхода
- Идея отложенного переноса
- Сравнение производительности
- Выводы

# Введение

**Цель:** повышение производительности операции умножения целых чисел посредством отложенного переноса

**Объект:** процесс умножения целых чисел

**Предмет:** механизм переноса переполнения из старшего разряда

# Актуальность

# Алгоритмы умножения

- «В столбик»
- Рекурсивный Карацубы-Офмана*
- Шёнхаге — Штрассена (Дискретное преобразование Фурье)
- Алгоритм Фюрера (развитие Шёнхаге — Штрассена )
- Comba*
- и другие

* Применимы для криптографических задач

# Подходы к повышению производительности

- Увеличение разрядности машинных слов
- Использование специализированных потоковых команд процессоров (MMX, SSE, SSE2, SSE3, SSE4, SSE5)
- Распараллеливание (многопоточность)
- Совершенствование алгоритмов
- Совершенствование структур данных

# Алгоритм Comba

**Алгоритм Comba.** Умножение целых

*Вход:* целое $a, b \in \mathbf{GF}(p)$, $w = 32$, $n = \log_{2^w} a$, $nk = 2n - 1$.

*Выход:* $c = a \cdot b$

1. $r_0^{(32)} \leftarrow 0$, $r_1^{(32)} \leftarrow 0$, $r_2^{(32)} \leftarrow 0$.

2. For $k \leftarrow 0$, $k < n$, $k{+}{+}$ do

2.1. For $i \leftarrow 0$, $j \leftarrow k$, $i \le k$, $i{+}{+}$, $j{-}{-}$ do

2.1.1. $(uv)^{(64)} \leftarrow a_i^{(32)} \cdot b_j^{(32)}$.

2.1.2. $r_0^{(32)} \leftarrow r_0^{(32)} + v^{(32)}$, $r_1^{(32)} \leftarrow r_1^{(32)} + u^{(32)} + carry$, $carry \leftarrow 0$.

2.1.3. $r_2^{(32)} \leftarrow r_2^{(32)} + carry$, $carry \leftarrow 0$.

2.2. $c_k^{(32)} \leftarrow r_0^{(32)}$, $r_0^{(32)} \leftarrow r_1^{(32)}$, $r_1^{(32)} \leftarrow r_2^{(32)}$, $r_2^{(32)} \leftarrow 0$.

3. For $k \leftarrow n$, $l \leftarrow 1, k < nk$, $k{+}{+}$, $l{+}{+}$ do

3.1. For $i \leftarrow l$, $j \leftarrow k - l$, $i < n$, $i{+}{+}$, $j{-}{-}$ do

3.1.1. $(uv)^{(64)} \leftarrow a_i^{(32)} \cdot b_j^{(32)}$.

3.1.2. $r_0^{(32)} \leftarrow r_0^{(32)} + v^{(32)}$, $r_1^{(32)} \leftarrow r_1^{(32)} + u^{(32)} + carry$, $carry \leftarrow 0$.

3.1.3. $r_2^{(32)} \leftarrow r_2^{(32)} + carry$, $carry \leftarrow 0$.

3.2. $c_k^{(32)} \leftarrow r_0^{(32)}$, $r_0^{(32)} \leftarrow r_1^{(32)}$, $r_1^{(32)} \leftarrow r_2^{(32)}$, $r_2^{(32)} \leftarrow 0$.

4. $c_{nk}^{(32)} \leftarrow r_0^{(32)}$.

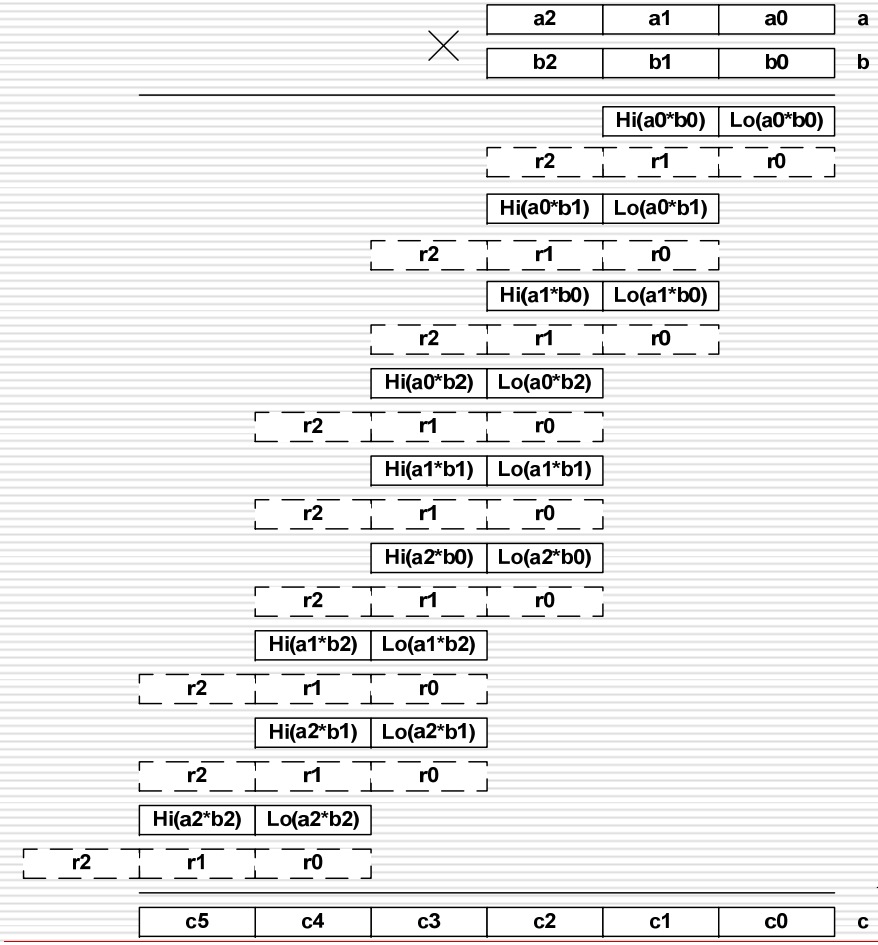
5. Return $(c)$.

# Алгоритм Comba: Сложность

$$I_{mul}^{Comba} = 4I_{assign}^{32} + n^2\left(1I_{mul}^{32} + 3I_{add}^{32} + 6I_{assign}^{32}\right) + 4\left(2n-1\right)I_{assign}^{32}$$

где $I_{assign}^{32}$ - операция присвоения 32-х разрядных чисел, $I_{add}^{32}$ - операция сложения 32-х разрядных чисел, $I_{mul}^{32}$ - операция умножения 32-х разрядных чисел.

# Алгоритм Comba: Структура




$$I_{mul}^{Comba} = 78I_{assign}^{32} + 9I_{mul}^{32} + 27I_{add}^{32}$$

# Алгоритм Comba: Недостатки

- Во вложенных циклах п. 2.1 и п. 3.1 происходит накопление суммы с переносом в 32-х разрядных временных переменных,  п. 2.1.2, 2.1.3 и п. 3.1.2, 3.1.3.
  Отметим, что в этом случае, происходит 3 операции сложения 32-х разрядных целых (две из них с учетом переноса), 3 присвоения 32-х разрядных переменных . Накопление суммы с учетом переноса производится на каждой итерации цикла 2.1.
- Во вложенных циклах п. 2.1 и п. 3.1 при накоплении суммы учитываются переносы, используя вставки на языке ассемблера, что в свою очередь не позволяет выполнять спаривание и распараллеливание таких операций, как следствие – неэффективное использование ресурсов процессора.
- Высокая внутренняя связность, в связи с учетом переносов, не дает возможности эффективно распараллелить циклы п. 2 и п. 3.
- Не учитывается возможность использования современными процессорами поддержки 64-х разрядных операций.

# Усовершенствования

- Избавиться от последовательного переноса – изменение порядка вычислений:
  - Использование 32-х и 64-х данных для накопления переносов
  - Применение переносов по необходимости
- Распараллелить не связанные вычисления:
  - Цикл накопления суммы произведений

# Алгоритм Modified Comba

**Алгоритм Modified Comba.** Умножение целых

*Вход:* целое $a, b \in \mathbf{GF}(p)$, $w = 32$, $n = \log_{2^w} a$, $nk = 2n - 1$.

*Выход:* $c = a \cdot b$

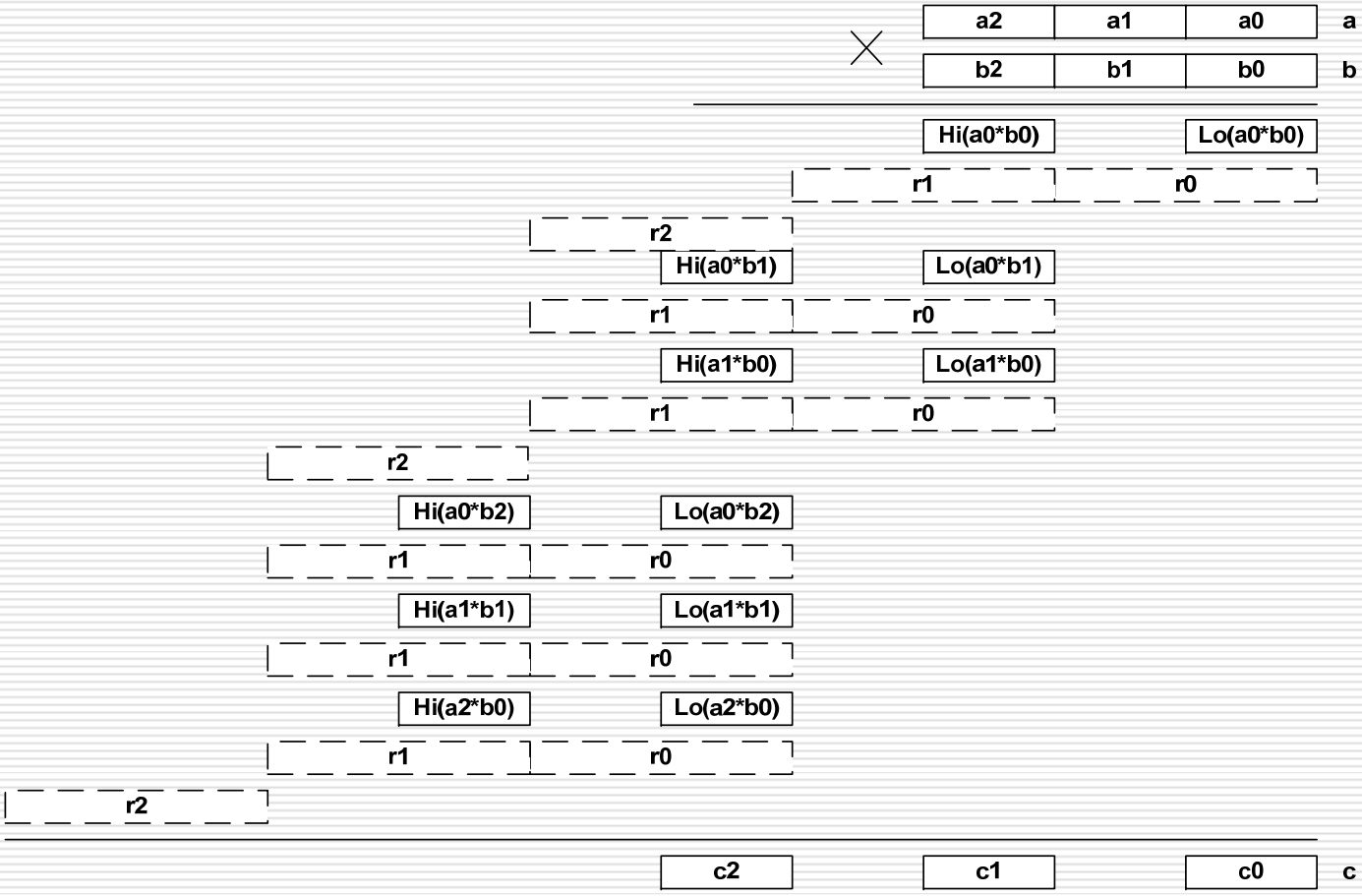
1. $r_0^{(64)} \leftarrow 0$, $r_1^{(64)} \leftarrow 0$, $r_2^{(64)} \leftarrow 0$.
2. For $k \leftarrow 0$, $k < n$, $k{+}{+}$ do
2.1. For $i \leftarrow 0$, $j \leftarrow k$, $i \le k$, $i{+}{+}$, $j{-}{-}$ do
2.1.1. $(uv)^{(64)} \leftarrow a_i^{(32)} \cdot b_j^{32}$.
2.1.2. $r_0^{(64)} \leftarrow r_0^{(64)} + v^{(32)}$, $r_1^{(64)} \leftarrow r_1^{(64)} + u^{(32)}$.
2.2. $r_1^{(64)} \leftarrow r_1^{(64)} + \mathrm{hi}_{(32)}\left(r_0^{(64)}\right)$, $r_2^{(64)} \leftarrow r_2^{(64)} + \mathrm{hi}_{(32)}\left(r_1^{(64)}\right)$.
2.3. $c_k^{(32)} \leftarrow \mathrm{low}_{(32)}\left(r_0^{(64)}\right)$, $r_0^{(64)} \leftarrow \mathrm{low}_{(32)}\left(r_1^{(64)}\right)$, $r_1^{(64)} \leftarrow \mathrm{low}_{(32)}\left(r_2^{(64)}\right)$, $r_2^{(64)} \leftarrow 0$.
3. For $k \leftarrow n$, $l \leftarrow 1, k < nk$, $k{+}{+}$, $l{+}{+}$ do
3.1. For $i \leftarrow l$, $j \leftarrow k - l$, $i < n$, $i{+}{+}$, $j{-}{-}$ do
3.1.1. $(uv)^{(64)} \leftarrow a_i^{(32)} \cdot b_j^{32}$.
3.1.2. $r_0^{(64)} \leftarrow r_0^{(64)} + v^{(32)}$, $r_1^{(64)} \leftarrow r_1^{(64)} + u^{(32)}$.
3.2. $r_1^{(64)} \leftarrow r_1^{(64)} + \mathrm{hi}_{(32)}\left(r_0^{(64)}\right)$, $r_2^{(64)} \leftarrow r_2^{(64)} + \mathrm{hi}_{(32)}\left(r_1^{(64)}\right)$.
3.3. $c_k^{(32)} \leftarrow \mathrm{low}_{(32)}\left(r_0^{(64)}\right)$, $r_0^{(64)} \leftarrow \mathrm{low}_{(32)}\left(r_1^{(64)}\right)$, $r_1^{(64)} \leftarrow \mathrm{low}_{(32)}\left(r_2^{(64)}\right)$, $r_2^{(64)} \leftarrow 0$.
4. $c_{nk}^{(32)} \leftarrow \mathrm{low}_{(32)}\left(r_0^{(64)}\right)$.
5. Return $(c)$.

# Алгоритм Modified Comba: Сложность

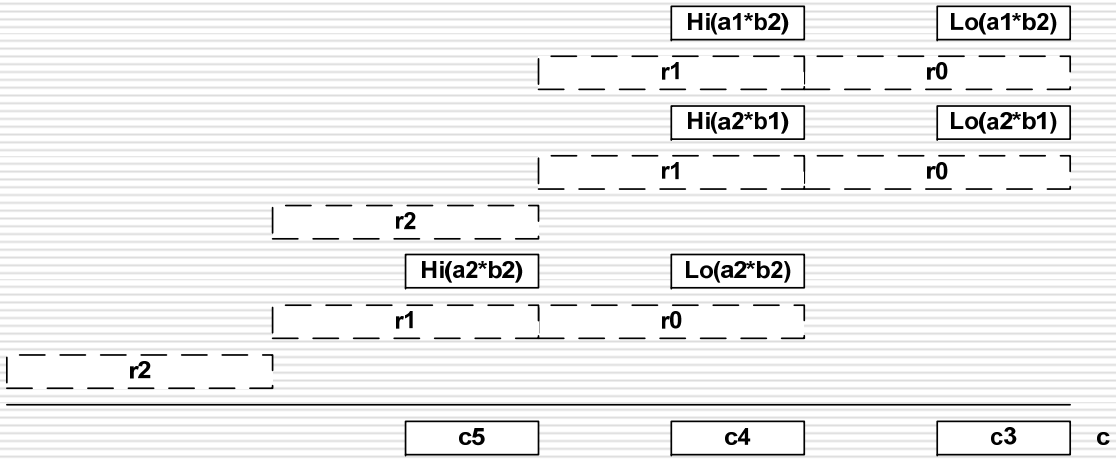
$$I_{mul}^{Mod.Comba} = 4I_{assign}^{64} + n^2\left(1I_{mul}^{32} + 2I_{add}^{64|32} + 2I_{assign}^{64}\right) + \left(2n-1\right)\left(2I_{add}^{64|32} + 1I_{assign}^{64} + 1I_{assign}^{32}\right)$$

где $I_{assign}^{32}$ - операция присвоения 32-х разрядных чисел, $I_{assign}^{64}$ - операция присвоения 64-х разрядных чисел, $I_{add}^{32}$ - операция сложения 32-х разрядных чисел, $I_{add}^{64|32}$ - операция сложения 32-х и 64-х разрядных чисел, $I_{mul}^{32}$ - операция умножения 32-х разрядных чисел.

# Алгоритм Modified Comba



# Алгоритм Modified Comba




$$I_{mul}^{Mod.Comba} = 27I_{assign}^{64} + 9I_{mul}^{32} + 28I_{add}^{64|32} + 5I_{assign}^{32}$$

# Алгоритм Modified Comba: Преимущества

- Современные 32-х разрядные процессоры эффективно реализуют операции сложения 32-х и 64-х разрядных целых чисел, используя 64-х либо 32-х разрядные команды и переменные. Это позволяет реализовать накопление переноса в результате сложения 32-х разрядных значений в 64-х разрядной переменной, что избавит от необходимости после каждого сложения, выполнять учет и корректировку переноса. Накопленный перенос будет учитываться на финальных итерациях цикла п. 2 и п. 3.
- Современные процессоры обладают многоядерной архитектурой, что позволяет им параллельно выполнять несколько потоков команд. Это позволяет выполнить итерации циклов п. 2 и п. 3 параллельно используя реализацию OpenMP .

# Библиотеки

- Целочисленной арифметики:

- CLN
- CryptoPP
- GNU MP
- LiDIA
- MIRACL
- NTL
- OpenSSL
- PIOLOGIE
- MUNTL
- TinyECC
- MPFQ
- GFA
- BBNUM
- FLINT
- LibTom: TomsFastMath

# Библиотеки

- Наилучшие показатели у GMP*
- В GMP 4.1.2 используется алгоритм умножения Карацубы
- Дальнейшее сравнение будет с GMP 4.1.2, скомпилированной с помощью Microsoft Visual Studio .NET (Win32, Max Speed, SSE2)
- Тестовое приложение на Microsoft Visual C++ 2010 (Win32, Max Speed, SSE2)
- Среднее для 1 млн. операций

# Библиотеки

- Modified Comba скомпилирован с помощью Microsoft Visual C++ 2010 (Max Speed, SSE2)
- Тестовое приложение на Microsoft Visual C++ 2010 (Max Speed, SSE2)
- Среднее для 1 млн. операций

# Тестовые поля

| Поле | Простое число |
|---|---|
| ***GF(p82)*** | 500000000000000008503491 |
| ***GF(p164)*** | 249999999999941304386009994022094639661975160756 99 |
| ***GF(p192)*** | 627710173538668076383578942 3 1760590137671947731828422840 81 |
| ***GF(p224)*** | 269599466671506397946670150870196306735579162600263081435100 66298881 |
| ***GF(p256)*** | 115792089210356248762697446949407573530086143415290314195533 631308867097853951 |
| ***GF(p320)*** | 427197407184182016479004215920066905783641406233172413793356 5193825968 6865762670800870819848380 97 |
| ***GF(p384)*** | 394020061963944792122790401001436138050797392704654466679469 0527962765 939911326356939895 630815229491355443653942643 |
| ***GF(p521)*** | 686479766013060971498190079908139321726943530014330540939446 345918554318339765539424505774633321719753296399637136332111 386476861244038034 03 728088927070054 49 |

* Поля рекомендованные NIST для ECC

# Замеры производительности

| Процессор | Операционная система | Ядер | Потоков выполнения команд | Примечание |
|---|---|---|---|---|
| Intel Pentium Dual Core E5400 | Microsoft Windows XP (x86) | 2 | 2 | |
| Intel Core i3 M350 | Microsoft Windows 7 (x64) | 2 | 4 | With Hyper Threading |

# Замеры производительности

| Поле | Время, мкс | | | | | |
|---|---|---|---|---|---|---|
| | Core i3 | | | Pentium Dual Core | | |
| | Comba* | Comba | GMP4.1 | Comba* | Comba | GMP4.1 |
| ***GF(p82)*** | 0,075 | 0,120 | 0,121 | 0,0687 | 0,119 | 0,125 |
| ***GF(p164)*** | 0,21 | 0,393 | 0,4 | 0,209 | 0,363 | 0,407 |
| ***GF(p192)*** | 0,276 | 0,393 | 0,41 | 0,289 | 0,363 | 0,414 |
| ***GF(p224)*** | 0,343 | 0,69 | 0,549 | 0,364 | 0,59 | 0,522 |
| ***GF(p256)*** | 0,422 | 0,875 | 0,638 | 0,456 | 0,744 | 0,648 |
| ***GF(p320)*** | 0,6973 | 1,278 | 0,97 | 0,686 | 1,053 | 0,969 |
| ***GF(p384)*** | 0,961 | 1,75 | 1,38 | 0,94 | 1,45 | 1,36 |
| ***GF(p521)*** | 1,63 | 2,8 | 2,663 | 1,486 | 2,41 | 2,643 |

# Выводы

- 1. Предложенный в работе подход отложенного переноса, позволяет добиться увеличения производительности программной реализации алгоритма умножения целых чисел Comba, в 1,5-2 раза, а также превзойти производительность популярной математической библиотеки GMP 4.1.2, в среднем в 1,5 раз.
- 2. Алгоритм умножения Modified Comba, является предпочтительнее алгоритма Карацубы, используемого в GMP, т.к. программная реализация алгоритма умножения Modified Comba оказалась быстрее, реализации алгоритма Карацубы в GMP, для современных аппаратных платформ (32-х и 64-х бит).
- 3. Механизм отложенного переноса позволяет применить различные техники распараллеливания к алгоритму Modified Comba, например OpenMP.

# Вопросы?

Спасибо за внимание!

# ООО «САЙФЕР ЛТД»

Владислав Ковтун

email: vlad.kovtun@cipher.kiev.ua
www: http://www.cipher.kiev.ua