A. Y. Rodionov

Efficient Hardware Implementation of the GOST R 34.10-2001, GOST R 34.10-2012 by FPGA

Keywords: digital signature, GOST R 34.10, efficient hardware implementation, Montgomery multiplication, DSP blocks, Jacobi representation, parallel computations

Algorithms of Montgomery multiplication, point's group operations in Jacobi representation, scalar multiplication are reviewed. The analysis and adaptation of those algorithms for efficient hardware implementation by FPGA consider several features: DSP blocks, possibility of parallel computations. Results of hardware implementation are presented.

А. Ю. Родионов

О СОЗДАНИИ ЭФФЕКТИВНОЙ АППАРАТНОЙ РЕАЛИЗАЦИИ ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012 НА ОСНОВЕ ПЛИС

В статье описывается подход к проектированию архитектуры высокопроизводительного функционального блока, который должен реализовывать ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012 для всех наборов параметров, принятых в отечественных стандартах. Данное условие и аппаратная реализация порождают ряд требований для выбираемых алгоритмов:

1) алгоритм редукции по простому модулю ρ должен поддерживать общий случай, то есть когда ρ не является простым числом псевдо-Мерсенна;

2) алгоритмы должны быть максимально просты и линейны из-за ограничения в ресурсах и сложности ПО конечного автомата;

3) алгоритмы должны быть итерационными для поддержки 256- и 512-битной схемы ЭЦП.

Минимальный набор алгоритмов, необходимых для реализации ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012, определяет структуру функционального блока. Алгоритмы генерации и проверки ЭЦП, скалярного произведения числа на точку целесообразно оформить в виде подпрограмм, исполняемых в конечном автомате. Наиболее трудоемкие операции могут быть реализованы аппаратно. Наилучшим кандидатом для выделения в АЛУ является алгоритм модулярного умножения. Для реализации на ПЛИС был выбран алгоритм умножения Монтгомери, позволяющий использовать битовые сдвиги вместо трудоемкой операции нахождения обратного значения по модулю при приведении по модулю результата умножения. В данной работе используется один из вариантов реализации, известный как FIOS [1]. Ниже приведен данный алгоритм.

Входные данные алгоритма: $A, B < M$; $2^{k(n-1)} \le M < 2^{kn}$; $M' = -M^{-1} \bmod 2^k$; $R = 2^{kn}$, *при НОД(M, R) = 1.*

Выходные данные алгоритма: $S \equiv ABR^{-1} (\bmod M), 0 \le S < 2M$.

1) $S_0 = 0$;

для $i = 0 \ldots n-1$ выполнить 3) – 4);

2) $q_i = (((S_i + a_i B) \bmod 2^k) M') \bmod 2^k$;

3) $S_{i+1} = (S_i + q_i M + a_i B) / 2^k$;

4) *если* $S \ge M$, *то* $S = S - M$;

5) конец.

Финальный шаг сравнения и вычитания длинных чисел труднореализуем в одном блоке с операциями внутри цикла. В то же время он необходим, так как полученный результат может

быть использован для следующей операции модулярного умножения в качестве входных данных, которые строго меньше M. Тем не менее его можно избежать, если ввести новые ограничения для входных данных A, $B < 2M$. При этом результат гарантированно будет $S < 2M$, если увеличить на одну итерацию цикл вычислений и принять верхнюю границу для $M < 2^{k(n+1)}$. Корректность данного выражения приведена в [2]. Поскольку теперь и входные, и выходные данные меньше $2M$, то все вычисления целесообразно производить по модулю $2M$, приводя к M лишь конечный результат.

Полученная структура алгоритма переносится на DSP-блоки ПЛИС, являющиеся высокопроизводительными конфигурируемыми устройствами. Наличие в них умножителя, сумматора, регистров задержек позволили применить конвейеризованный алгоритм умножения Монтгомери на двух DSP-блоках, объединенных в каскад.

Благодаря компактности реализации блока умножения Монтгомери стало возможным использование нескольких блоков для распараллеливания вычислений. С этой целью был произведен анализ групповых операций над точками в Якобиевом представлении. В результате были определены зависимости данных во время исполнения и оптимальное количество блоков модулярного умножения. При этом была введена дополнительная координата Z^4, позволившая сократить наибольший путь до конечного результата в операции удвоения точки. Ниже приведены алгоритмы удвоения и сложения точки на эллиптической кривой при параллельном использовании четырех модулярных умножителей и конечного автомата, реализующего модулярные операции сложения, вычитания и деления на 2.

Таблица 1. Алгоритм удвоения точки

$P_2(X_2, Y_2, Z_2, Z_2^4) = 2P_1(X_1, Y_1, Z_1, Z_1^4)$

Конечный автомат	**Умножитель Монтгомери № 1**	**Умножитель Монтгомери № 2**	**Умножитель Монтгомери № 3**	**Умножитель Монтгомери № 4**
$Y_2 = 2Y_1$			$T_3 = aZ_1^4$	$T_0 = X_1^2$
	$Y_2 = Y_2^2$	$Z_2 = Y_2 Z_1$	…	…
	…	…	…	…
$T_2 = 2T_0$	…	…		
$T_0 = T_0 + T_2$	$T_1 = Y_2^2$	$T_3 = Z_2^2$	$Y_2 = X_1 Y_2$	
$T_0 = T_0 + T_3$	…	…	…	
	…	…	…	$X_2 = T_0^2$
$T_2 = 2Y_2$				…
$Y_2 = T_2 + Y_2$				…
$Y_2 = Y_2 - X_2$				
$X_2 = X_2 - T_2$		$Z_2^4 = T_3^2$	$Y_2 = T_0 Y_2$	
$T_1 = T_1 / 2$		…	…	

		…	…	
$Y_2 = Y_2 - T_1$				

Таблица 2. Алгоритм сложения точек

$$P_2\,(X_2, Y_2, Z_2, Z_2^4) = P_0\,(X_0, Y_0, Z_0, Z_0^4) + P_1\,(X_1, Y_1, Z_1, Z_1^4)$$

Конечный автомат	**Умножитель Монтгомери № 1**	**Умножитель Монтгомери № 2**	**Умножитель Монтгомери № 3**	**Умножитель Монтгомери № 4**
	$T_4 = Z_0^2$	$T_0 = Z_1^2$	$T_2 = Z_0 Z_1$	
	$T_1 = T_4 Z_0$	$T_3 = T_0 Z_1$	$X_2 = X_0 T_0$	$T_4 = T_4 X_1$
$T_4 = T_4 - X_2$	$T_1 = T_1 Y_1$	$Y_2 = T_3 Y_0$		
	…	…	$T_0 = T_4^2$	$Z_2 = T_2 T_4$
	…	…	…	…
$T_1 = T_1 - Y_2$			…	…
$Y_2 = T_1 - Y_2$	$X_2 = T_1 T_1$	$T_0 = T_0 X_2$	$T_2 = T_4 T_0$	$T_4 = Z_2 Z_2$
	…	…	…	…
	…	…	…	…
$T_3 = 2T_0$				
$T_0 = T_0 + T_3$				
$T_0 = T_0 - X_2$				
$X_2 = X_2 - T_3$		$T_0 = T_0 T_1$	$Y_2 = Y_2 T_2$	$Z_2^4 = T_4^2$
$X_2 = X_2 - T_2$		…	…	…
$Y_2 = Y_2 + T_0$				

Ниже, в таблице 3, приведено время исполнения сложения и удвоения точки, оцененное в количестве необходимых модулярных умножений. Время выполнения модулярных сложений и вычитаний не учитывается, поскольку оно значительно меньше времени выполнения модулярных умножений.

Таблица 3. Вычислительная сложность групповых операций над точками эллиптической кривой в Якобиевом представлении

	Удвоение точки	**Удвоение точки на 4 умножителях**	**Сложение точки**	**Сложение точки на 4 умножителях**
Количество модулярных умножений	8	3	16	5

Для скалярного произведения точки на число было рассмотрено два метода: wNAF и алгоритм одновременного вычисления степеней. Ниже в таблице приведена оценка их вычислительной сложности для случаев с произвольной и фиксированной точкой при длине обрабатываемых данных $m = 256$ и $m = 512$ бит, где k — параметр алгоритма, соответствующий количеству бит скаляра, действия с которыми производятся в один момент времени. Дополнительным ограничением является размер таблицы с предвычисленными значениями, которая не может превышать размер буфера обмена функционального блока в 8 килобайт. Результаты приведены в количестве модулярных умножений при использовании рассмотренных выше групповых операций над точками на эллиптической кривой.

Таблица 4. Вычислительная сложность скалярного произведения для произвольных точек эллиптической кривой

Алгоритм	Длина чисел в битах	k	Количество модулярных умножений	Размер таблицы в килобайтах
SimMul	256	2	763	2,1 (генерация ЭЦП для 256 бит)
SimMul	256	2	1467	2,1 (проверка ЭЦП для 256 бит)
SimMul	512	2	1467	4,1 (генерация ЭЦП для 512 бит)
SimMul	512	3	2875	4,1 (проверка ЭЦП для 512 бит)
wNAF	256	5	1020	1
wNAF	512	6	1980	4

Таблица 5. Вычислительная сложность скалярного произведения для фиксированных точек эллиптической кривой

Алгоритм	Длина чисел в битах	k	Количество модулярных умножений	Размер таблицы в килобайтах
SimMul	256	2	704	2,1 (генерация ЭЦП для 256 бит)
SimMul	256	2	1408	2,1 (проверка ЭЦП для 256 бит)
SimMul	512	2	1408	4,1 (генерация ЭЦП для 512 бит)
SimMul	512	3	2816	4,1 (проверка ЭЦП для 512 бит)
wNAF	256	5	928	4,25
wNAF	512	6	1902	4,1

При заданных наборах параметров случаю с фиксированной точкой соответствуют алгоритмы генерации ЭЦП, проверки ЭЦП, генерации ключевой пары; с произвольной точкой — генерация ключа ДХ. Поскольку алгоритм одновременного возведения в степень требует меньшего времени исполнения для наибольшего числа случаев использования, был выбран именно он.

Рассматриваемый функциональный блок был реализован на основе ПЛИС xc5vfx100t. Его характеристики приведены в таблице 6.

Таблица 6. Характеристики функционального блока на основе ПЛИС xc5vfx100t

	Flip Flop	LUT	DSP	BRAM	Время генерации ЭЦП, мс	Время проверки ЭЦП, мс
Значение	2872	3256	8	5	3,33 (256 бит) 16,66 (512 бит)	6,25 (256 бит) 31,25 (512 бит)

СПИСОК ЛИТЕРАТУРЫ:

1. *Grobschadl J., Posch K. C., Tillich S.* Architectural Enhancements to Support Digital Signal Processing and Public-Key Cryptography // Proceedings of the 2nd Workshop on Intelligent Solutions in Embedded Systems. 2004. P. 129–143.
2. *Hachez G., Quisquater J.-J.* Montgomery Exponentiation with no Final Subtractions: Improved Results // Cryptographic Hardware and Embedded Systems — CHES 2000. P. 293–301.

REFERENCES:

1. *Grobschadl J., Posch K. C., Tillich S.* Architectural Enhancements to Support Digital Signal Processing and Public-Key Cryptography // Proceedings of the 2nd Workshop on Intelligent Solutions in Embedded Systems. 2004. P. 129–143.
2. *Hachez G., Quisquater J.-J.* Montgomery Exponentiation with no Final Subtractions: Improved Results // Cryptographic Hardware and Embedded Systems — CHES 2000. P. 293–301.