Jakub OLSZYNA, Wiesław WINIECKI
POLITECHNIKA WARSZAWSKA, INSTYTUT RADIOELEKTRONIKI,
ul. Nowowiejska 15/19, 00-665 Warszawa

Niskomocowy koprocesor kryptograficzny dla autonomicznych bezprzewodowych sieci czujnikowych

Mgr inż. Jakub OLSZYNA

Absolwent Wydziału Elektroniki i Technik Informacyjnych PW, specjalności Radiokomunikacja i Techniki Multimedialne. Obecnie doktorant w Instytucie Radioelektroniki, członek IEEE. Autor lub współautor ponad 15 publikacji naukowych. Aktualne obszary zainteresowań: rozproszone systemy pomiarowo-sterujące, sieci czujnikowe, arytmetyka modularna w kryptografii klucza publicznego.

e-mail: J.Olszyna@ire.pw.edu.pl

Prof. dr hab. inż. Wiesław WINIECKI

Prof. nzw. na Wydziale Elektroniki i Technik Informacyjnych PW. Kierownik zespołu Komputerowej Techniki Pomiarowej. Członek Komitetu Metrologii i Aparatury Naukowej PAN, wiceprezes POLSPAR, członek IEEE. Autor lub współautor 4 książek i ponad 170 publikacji naukowych. Obszary zainteresowań: systemy pomiarowe, przyrządy wirtualne, nowoczesne technologie komunikacyjne i programowe w skupionych i rozproszonych systemach pomiarowo-kontrolnych.

e-mail: W.Winiecki@ire.pw.edu.pl

Streszczenie

W artykule przedstawiono układ koprocesora kryptograficznego dostosowanego do specyfiki autonomicznych bezprzewodowych sieci czujnikowych. Układ taki z założenia ma wspomagać realizację różnych algorytmów kryptografii klucza publicznego bazujących na arytmetyce modularnej. Koprocesor opisany w postaci modeli GEZEL i VHDL może w prosty sposób zostać zrealizowany w postaci układu ASIC lub uruchomiony na niskomocowym układzie FPGA.

Słowa kluczowe: autonomiczne bezprzewodowe sieci czujnikowe, arytmetyka modularna, koprocesor kryptograficzny.

Low-power cryptographic coprocessor for autonomous wireless sensor networks

Abstract

The concept of autonomous wireless sensor networks involves energy harvesting, as well as effective management of system resources. Public-key cryptography (PKC) offers the advantage of elegant key agreement schemes with which a secret key can be securely established over unsecure channels. In addition to solving the key management problem, the other major application of PKC is digital signatures, with which non-repudiation of messages exchanges can be achieved. The motivation for studying a low-power and area efficient modular arithmetic algorithm comes from enabling public-key security for low-power devices that can perform under constrained environment like autonomous wireless sensor networks. This paper presents a cryptographic coprocessor tailored to the autonomous wireless sensor network constraints. Such a system is aimed at supporting the implementation of different public-key cryptosystems based on modular arithmetic in GF(p). The coprocessor key components are described as GEZEL models and can be easily transferred to VHDL and implemented in hardware.

Keywords: autonomous wireless sensor networks, modular arithmetic, cryptographic coprocessor.

1. Wprowadzenie

Koncepcja autonomicznych bezprzewodowych sieci czujnikowych zakłada pobieranie energii z otoczenia, jak również efektywne zarządzanie zasobami systemowymi. W zależności od charakterystyki źródła i konkretnej realizacji, obecnie możliwe jest pozyskanie z otoczenia od kilkudziesięciu do kilku tysięcy μW na każdy cm^2 układu pozyskującego energię [1]. Obszary zastosowań bezprzewodowych sieci czujnikowych czynią kluczowym zagadnienia związane z bezpieczeństwem wytwarzanych i przekazywanych za ich pośrednictwem informacji [2]. Bezpieczeństwo jest zapewniane przy użyciu specjalizowanych kryptograficznych modułów sprzętowych, które umożliwiają efektywną i oszczędną realizację algorytmów i protokołów kryptograficznych.

Kryptografia klucza publicznego obejmuje metody szyfrowania z wykorzystaniem publicznie dostępnego klucza - dwa węzły sieci są dzięki temu w stanie wymieniać poufne informacje bez konieczności uzgadniania wspólnego klucza. W przypadku autonomicznych bezprzewodowych sieci czujnikowych takie podejście jest szczególnie korzystne, pozwala na łatwe dołączanie nowych węzłów oraz ograniczenie ilości przesyłanych danych kontrolnych. Najpopularniejsze algorytmy z kluczem publicznym bazują na operacjach arytmetyki modularnej w strukturach algebraicznych, w szczególności ciałach skończonych. Zysk z optymalizacji implementacji podstawowych operacji w ciałach skończonych w znaczący sposób przekłada się więc na efektywność realizacji kryptosystemów klucza publicznego. W przypadku pozostającego standardem algorytmu RSA najbardziej kosztowną operacją jest potęgowanie modularne. Znaczne ograniczenie zasobów przy zachowaniu ustalonego poziomu bezpieczeństwa można osiągnąć przy zastosowaniu algorytmu Rabina, gdzie zamiast potęgowania występuje jedynie podnoszenie do kwadratu. Algorytmy kryptografii krzywych eliptycznych wykorzystują dodatkowo operację inwersji modularnej [3].

Wszystkie wymienione operacje mogą zostać zrealizowane za pomocą układów arytmetycznych dla ciała skończonego $GF(p)$. Ze względu na stosowane rozmiary operandów (rzędu kilku tysięcy bitów w przypadku RSA) algorytmy opracowane i zoptymalizowane na potrzeby DSP nie znajdują prostego zastosowania. Rozwiązaniem tego problemu jest dedykowany sprzętowy układ koprocesora kryptograficznego wspomagającego realizację kluczowych operacji różnych algorytmów kryptograficznych (w szczególności algorytmów kryptografii klucza publicznego). Ze względu na specyfikę autonomicznych bezprzewodowych sieci czujnikowych (asymetria mocy po stronie sensorowej), moduły składowe koprocesora powinny być optymalizowane według kryterium zapotrzebowania na energię (zwiększenie czasu życia modułu przy zasilaniu bateryjnym) lub moc (umożliwienie zasilania modułu za pomocą układu pozyskującego energię z otoczenia). W szczególności dotyczy to doboru samych algorytmów jak i sposobu implementacji.

2. Mnożenie modularne

Mnożenie modularne dwóch elementów ciała $GF(p)$ może być przedstawione jako

$$u = ab \bmod p = ab - \lfloor ab / p \rfloor p \,. \tag{1}$$

Najprostszym algorytmem szeregowym realizującym mnożenie modularne w $GF(p)$ jest algorytm mnożenia przeplatanego z redukcją. W ciągu k cykli zegara algorytm oblicza wynik cząstkowy za pomocą przesunięć bitowych i sumowań. W każdej iteracji wynik jest mniejszy od $2p$, dlatego konieczna jest końcowa redukcja, w co najwyżej dwóch krokach. W algorytmie wymiar rezultatu mnożenia w zapisie binarnym jest tylko o 1 lub 2 bity większy od wymiaru operandów.

Wejście: $a, b, p, 0 \le a, b < p, k = \lceil \log_2 a \rceil$

Wyjście: $u = a \cdot b \bmod p$

Krok 1. $u = 0$

Krok 2. dla $i = k - 1$ do 0

a. $u = 2 \cdot u + a_i b$

b. dopóki $u \ge p : u = u - p$

Alternatywą dla opisanego wyżej rozwiązania jest algorytm Montgomery'ego, pozwala na realizację mnożenia modularnego liczb całkowitych bez konieczności wykonywania dzielenia, stosując jedynie serię sumowań i przesunięć bitowych operandów [4]. Dla danych elementów *a*, *b*, *p* algorytm wyznacza

$$\overline{u} = M_{p,r}(a,b) = a \cdot b \cdot r^{-1} \bmod p . \qquad (2)$$

gdzie *r* jest pewną ustaloną liczbą naturalną. Aby uzyskać wynik mnożenia modularnego należy jeszcze wyeliminować czynnik r^{-1}

$$u = M_{p,r}(\overline{u}, r^2 \bmod p) = a \cdot b \bmod p . \qquad (1)$$

Algorytm Montgomery'ego przetwarza operandy od lewej do prawej strony (inaczej, niż w przypadku algorytmu mnożenia przeplatanego z redukcją).

Wejście: $a, b, p, 0 \le a, b < p, k = \lceil \log_2 p \rceil, r = 2^k, n' = -p^{-1} \bmod 2$

Wyjście: $u = a \cdot b \cdot r^{-1} \bmod p$

Krok 1. $u = 0$

Krok 2. dla $i = 0$ do $k - 1$

a. $u = u + a_i b + n'_0 \, a_i b_0 p$

b. $u = \lfloor u / 2 \rfloor$

Krok 3. jeżeli $u \ge p : u = u - p$

Dzięki odpowiedniemu doborowi *r* oraz faktowi, że *p* jest liczbą nieparzystą wiemy, że najmniej znaczący bit *p'* jest zawsze równy 1, co upraszcza krok 2a.

Algorytm Barretta pozwala na efektywną realizację redukcji modularnej [5]. Idea działania opiera się na oszacowaniu wyniku redukcji *a* modulo *p*, przy wykorzystaniu obliczonej wcześniej wartości parametru μ. Do znalezienia wyniku wystarczą 3 mnożenia i maksymalnie 3 dodawania/odejmowania. Parametr *l* określa podstawę systemu liczbowego zapisu operandów, zgodnie z oryginalnym algorytmem należy przyjąć wartość większą od 3. W przypadku, gdy *l* wynosi 4 można korzystać z zapisu binarnego, a operacje dzielenia z kroków 2 i 3 mogą być zrealizowane w postaci przesunięć bitowych.

Wejście: $a, b, p, 0 \le a, b < p, k = \lceil \log_2 p \rceil, \mu = \lfloor l^{2k} / p \rfloor, l = 4$

Wyjście: $u = a \cdot b \bmod p$

Krok 1. $u = a \cdot b$

Krok 2. $q = \lfloor u / l^{k-1} \rfloor \cdot \mu$

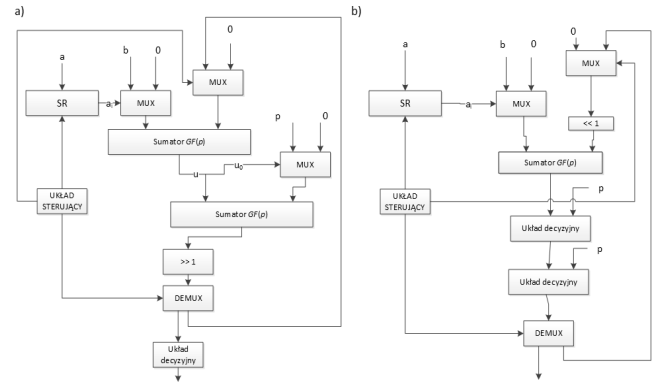
Krok 3. $q = \lfloor q / l^{k+1} \rfloor$

Krok 4. $r_1 = u \bmod l^{k+1}, r_2 = q \cdot p \bmod l^{k+1}, \; u = r_1 - r_2$

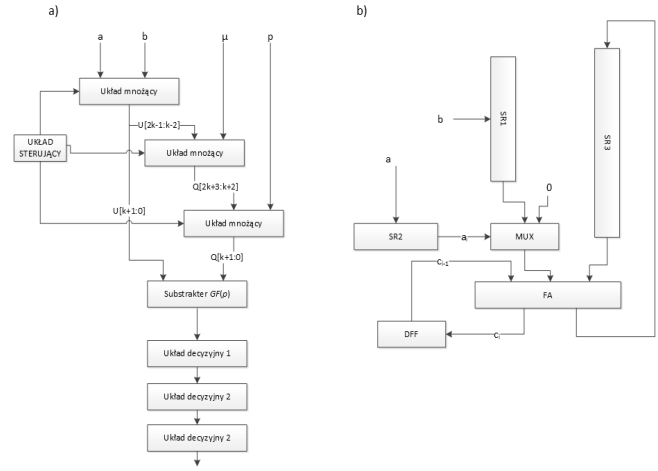
Krok 5. jeżeli $u < 0 : u = u + l^{k+1}$

Krok 6. dopóki $u \ge p : u = u - p$

Schematy blokowe realizacji algorytmu mnożenia przeplatanego z redukcją oraz mnożenia Montgomery'ego zostały przedstawione na rysunku 1. Schemat realizacji algorytmu Barretta został przedstawiony na rysunku 2a. Szeregowy układ mnożący pozwalający na realizację operacji mnożenia przedstawiono na rysunku 2b.



Rys. 1. a) Schemat układu realizującego algorytm Montgomery'ego b) Schemat układu realizującego algorytm mnożenia przeplatanego z redukcją
Fig. 1. Block diagram of: the Montgomery multiplication circuit (a), the interleaved multiplication circuit (b)



Rys. 2. a) Schemat blokowy układu realizującego algorytm Barretta b) Schemat blokowy szeregowego układu mnożącego
Fig. 2. Block diagram of: the Barrett multiplication circuit (a),the serial multiplier circuit (b)

3. Potęgowanie modularne

Potęgowanie modularne elementu ciała *GF*(*p*) z wykładnikiem *e* sprowadza się do wykonania serii mnożeń modularnych

$$u = a^e \bmod p . \qquad (4)$$

Najmniej złożoną realizację potęgowania modularnego umożliwiają algorytmy binarne, które przetwarzają wykładnik bit po bicie i wyliczają wynik końcowy za pomocą serii podnoszeń do kwadratu i mnożeń [6]. Algorytm RtL przetwarza wykładnik od strony najmniej znaczących bitów w kierunku najbardziej znaczących bitów.

Wejście: $a, e, p, 0 \le a < p, e = \sum_{i=0}^{l-1} e_i 2^i, e_{l-1} = 1$

Wyjście: $u = a^e \bmod p$

Krok 1. $u = 1$

Krok 2. dla $i = 0$ do $l - 1$

a. jeżeli $e_i = 1 : u = u \cdot a \bmod p$

b. $a = aa \bmod p$

Algorytm LtR przetwarza wykładnik od strony najbardziej znaczących bitów w kierunku najmniej znaczących bitów.

Wejście: $a, e, p, 0 \le a < p, e = \sum_{i=0}^{l-1} e_i 2^i, e_{l-1} = 1$

Wyjście: $u = a^e \bmod p$

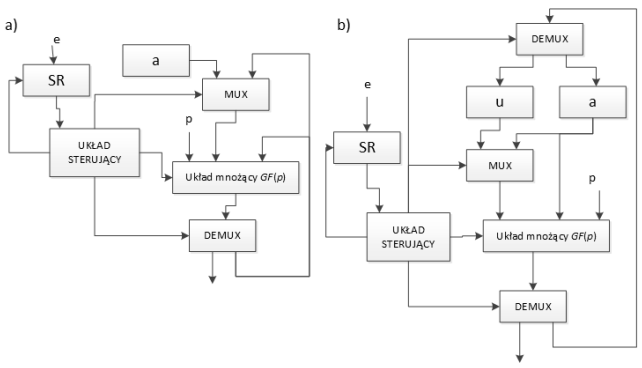
Krok 1. $u = 1$

Krok 2. dla $i = l-1$ do 0

a. $u = uu \bmod p$

b. jeżeli $e_i = 1 : u = u \cdot a \bmod p$

Schematy układów realizujących algorytmy LtR i RtL zostały przedstawione na rysunku 3. W każdym może zostać użyty dowolny układ mnożący spośród tych opisanych w rozdziale 2.



Rys. 3. a) Schemat blokowy układu realizującego algorytm LtR b) Schemat blokowy układu realizującego algorytm RtL

Fig. 3. Block diagram of: the LtR circuit (a), the RtL circuit (b)

4. Inwersja

Inwersja elementu ciała $GF(p)$ określa element odwrotny względem operacji mnożenia, tj.

$$a^{-1}a \equiv 1 \bmod p\,. \tag{5}$$

W najprostszy sposób inwersja w ciele $GF(p)$ może być zrealizowana w oparciu o Małe Twierdzenie Fermata, tj.

$$a^{p-1} \equiv 1 \bmod p \leftrightarrow a^{-1} \equiv a^{p-2} \bmod p\,. \tag{6}$$

W takim przypadku inwersja sprowadza się do operacji potęgowania z wykładnikiem p-2. Dużo efektywniejszym rozwiązaniem są algorytmy oparte na binarnej wersji rozszerzonego algorytmu Euklidesa, jak algorytm EEI [7].

Wejście: a, p

Wyjście: $r = a^{-1} \bmod p$

Krok 1. $u = p, v = a, r = 0, s = 1$

Krok 2. dopóki $v > 0$

a. jeżeli $u_0 = 0 : u = u/2, r = (r + r_0 p)/2$

b. inaczej jeżeli $v_0 = 0 : v = v/2, s = (s + s_0 p)/2$

c. inaczej jeżeli $u > v : u = (u - v), r = r - s$

jeżeli $r < 0 : r = r + p$

d. wpp. : $v = (v - u), s = s - r$

jeżeli $s < 0 : s = s + p$

Krok 3. jeżeli $r > p : r = r - p$

Krok 4. jeżeli $r < 0 : r = r + p$

Alternatywą dla powyższego jest algorytm Kaliskiego [7], który pozwala także na obliczenie inwersji w dziedzinie Montgomery'ego (patrz: rozdział 2).

Wejście: $a, p, n = \lceil \log_2 p \rceil$

Wyjście: $r = a^{-1} \bmod p$

Krok 1. $u = p, v = a, r = 0, s = 1, k = 0$

Krok 2. dopóki $v > 0$

a. jeżeli $u_0 = 0 : u = u/2, s = 2s$

b. inaczej jeżeli $v_0 = 0 : v = v/2, r = 2r$

c. inaczej jeżeli $u > v : u = (u - v)/2, r = r + s, s = 2s$

d. wpp.: $v = (v - u)/2, s = r + s, r = 2r$

e. $k = k + 1$

Krok 3. jeżeli $r \ge p : r = 2p - r$

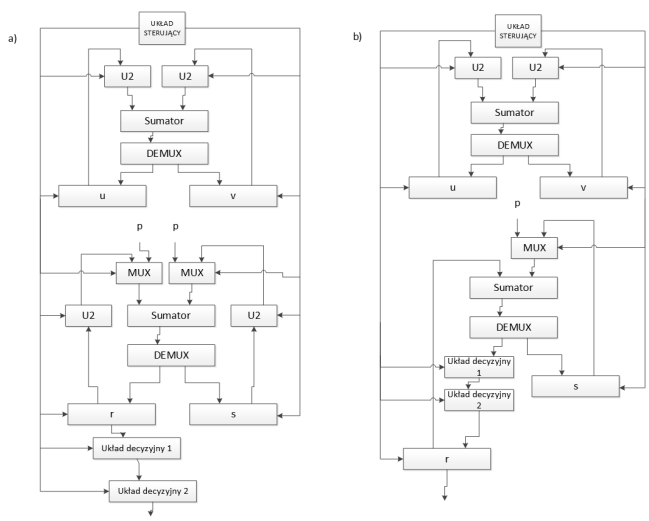
wpp. : $r = p - r$

Krok 4. dla $i = 0$ do $k - 1$

a. jeżeli $r_0 = 0 : r = r/2$

b. wpp.: $r = (r + p)/2$

Jeżeli obliczenia są wykonywane w dziedzinie Montgomery'ego do obliczenia inwersji wystarczy *k-n* iteracji pętli w kroku 4. Schematy blokowe układów realizujących algorytm EEI i algorytm Kaliskiego zostały przedstawione na rysunku 4.



Rys. 4. a) Schemat blokowy układu realizującego algorytm EEI b) Schemat blokowy układu realizującego algorytm Kaliskiego

Fig. 4. Block diagram of: EEI circuit (a), Kaliski circuit (b)

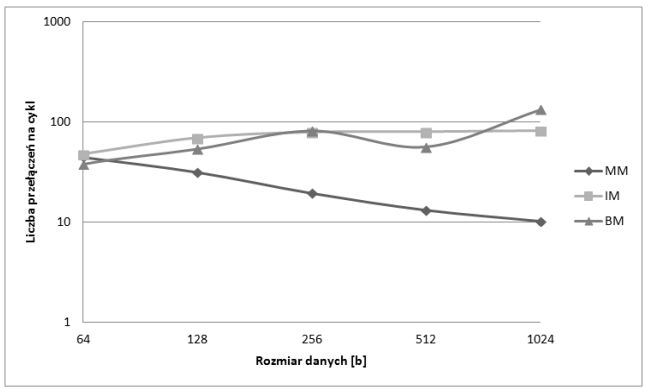
5. Symulacje i weryfikacja rezultatów

Moc rozpraszana w układach CMOS może być wyrażona jako:

$$P = (\frac{1}{2} C_{wy} V_{DD}^2 + Q V_{DD}) N f + (I_{upływ} + I_{stat}) V_{DD}\,. \tag{7}$$

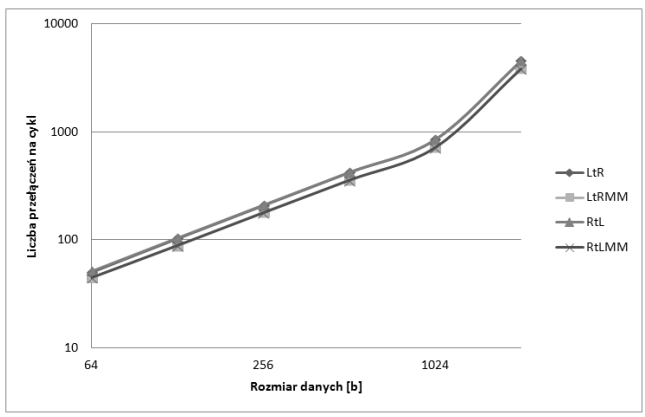
Pierwszą składową w powyższym wzorze jest moc dynamiczna, która zależy od zastępczej pojemności wyjściowej bramki C_{wy}, napięcia zasilania V_{DD}, ładunku Q przenoszonego przez prąd zwarcia, a także częstotliwości taktowania zegara f oraz parametru N określającego średnią liczbę przełączeń stanu bramek pomiędzy wartościami '0' a '1' w ciągu jednego cyklu zegara. Drugą składową jest moc statyczna, zależna od konkretnej technologii implementacji oraz pośrednio od wymiarów układu.

Zaprezentowane w rozdziałach 2, 3 i 4 algorytmy zostały zaimplementowane w Scilab, w postaci modelu referencyjnego, co umożliwiło późniejszą weryfikację poprawności działania układów po wprowadzaniu modyfikacji (testy regresyjne). Algorytmy zostały następnie opisane w postaci modeli w wysokopoziomowym języku GEZEL [8]. Przyjęcie takiego sposobu opisu umożliwiło szacowanie mocy dynamicznej na wysokim poziomie abstrakcji, a także przeprowadzenie pełnych testów funkcjonalnych w oparciu o model referencyjny. Proces projektowania przebiegał zgodnie z regułami redukcji mocy rozpraszanej na poziomie algorytmu i architektury, gdzie istnieje największa możliwość redukcji mocy dynamicznej [9]. Symulacje wykonano dla stałego zestawu wektorów testowych, dla różnych rozmiarów danych. Na rysunku 5 przedstawiono porównanie liczby przełączeń na cykl (szacowanie mocy) w funkcji rozmiaru danych dla układów mnożenia przeplatanego z redukcją (IM), mnożenia Montgomery'ego (MM) i mnożenia Barretta (BM).



Rys. 5. Szacowanie mocy dynamicznej dla algorytmów mnożenia modularnego
Fig. 5. Dynamic power estimation for modular multiplication algorithms

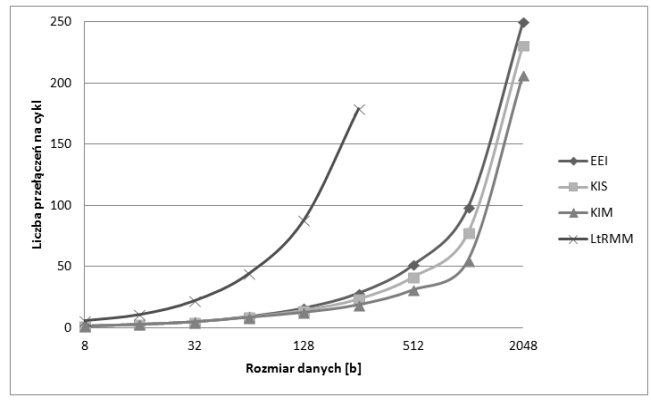
Na rysunku 6 przedstawiono porównanie liczby przełączeń na cykl w funkcji rozmiaru danych dla układów potęgowania modularnego. Układy RtL i LtR wykorzystują mnożenie przeplatane z redukcją, układy RtLMM i LtRMM – mnożenie Montgomery'ego.



Rys. 6. Szacowanie mocy dynamicznej dla algorytmów potęgowania modularnego
Fig. 6. Dynamic power estimation of modular exponentiation algorithms

Na rysunku 7 przedstawiono porównanie liczby przełączeń na cykl w funkcji rozmiaru danych dla układów inwersji. Układy KIS i KIM reprezentują realizacje algorytmu Kaliskiego odpowiednio dla liczb w reprezentacji standardowej oraz w reprezentacji Montgomery'ego. Dla porównania wyniki zestawiono z danymi dla układu realizującego inwersję przez potęgowanie (LtRMM).

Przedstawione rezultaty symulacji modeli szeregowych realizacji różnych algorytmów arytmetyki modularnej w $GF(p)$ pokazują, że najmniejsza moc dynamiczna jest osiągana dla układów operujących w dziedzinie Montgomery'ego. Szczególnie widoczne jest to dla większych rozmiarów danych (klucza), stosowanych praktycznie w kryptografii. Dlatego też do realizacji koprocesora kryptograficznego wybrano układ mnożenia modularnego Montgomery'ego (MM), układ potęgowania LtR z mnożeniem Montgomery'ego (LtRMM) oraz układ inwersji modularnej Kaliskiego (KIM).



Rys. 7. Szacowanie mocy dynamicznej dla algorytmów inwersji modularnej
Fig. 7. Dynamic power estimation for modular inverse algorithms

6. Podsumowanie

Układy realizujące operacje arytmetyki modularnej zaprezentowane w artykule mogą służyć do implementacji podstawowych usług kryptografii klucza publicznego (poufność, uwierzytelnianie strony, uwierzytelnianie danych i integralność danych) w autonomicznych bezprzewodowych sieciach czujnikowych w postaci komponentów układu koprocesora kryptograficznego. Ze względu na specyfikę autonomicznych bezprzewodowych sieci czujnikowych komponenty koprocesora zostały zoptymalizowane według kryterium zapotrzebowania na moc (umożliwienie zasilania modułu za pomocą układu pozyskującego energię z otoczenia). W szczególności dotyczyło to doboru samych algorytmów jak i sposobu ich implementacji. Koprocesor został opisany w postaci modeli GEZEL, za pomocą których można automatycznie wygenerować modele VHDL. Dalsze prace będą skoncentrowane na implementacji zaproponowanych modeli w konkretnej technologii (ASIC lub FPGA) oraz oszacowaniu całkowitej mocy rozpraszanej oraz podatności na ataki związane z analizą mocy oraz czasu przetwarzania.

7. Literatura

[1] Olszyna J., Winiecki W.: Metody pozyskiwania energii dla autonomicznych bezprzewodowych sieci czujnikowych, PAK, t. 58 (10), s. 837–839, 2012.
[2] Olszyna J., Winiecki W.: Bezpieczeństwo w sieciach czujnikowych, Przegląd Telekomunikacyjny, vol. 83 (12), s. 1738-1740, 2010.
[3] Olszyna J.: Modular multiplication in GF (p) for public-key cryptography, Proc. SPIE, s. 84542E–84542E–8, 2012.
[4] Montgomery P. L.: Modular multiplication without trial division, Mathematics of Computation, t. 44(170), s. 519–521, 1985.
[5] Barrett P.: Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor, Proc. Advances in Cryptology, Santa Barbara, California, United States, s. 311–323, 1987.
[6] Karpiński M.: Bezpieczeństwo informacji, PAK, 2012.
[7] Lórencz R.: New Algorithm for Classical Modular Inverse, CHES 2002, t. 2523, s. 57–70, 2003.
[8] Verbauwhede I. M. R. (Ed.): Secure Integrated Circuits and Systems, Springer, 2010.
[9] Piguet C.: Low-Power CMOS Circuits: Technology, Logic Design and CAD Tools, CRC Press, 2006.

otrzymano / received: 20.03.2013
przyjęto do druku / accepted: 01.05.2013 *artykuł recenzowany / revised paper*