



Politechnika Wrocławska

**Opracowanie publikacji Danila
Gorodocky'ego i Tiziano Villa'y -
sumatory modularne**

**Łukasz Wdowiak 264026
Damian Jabłoński 264025**

Prowadzący: dr inż. Piotr Patronik
Grupa: K03-47a, Pon 15:15-16:55 TN

Wydział Informatyki i Telekomunikacji
Informatyka Techniczna
IV semestr

Spis treści

1	Cele projektu	2
2	Założenia projektu	2
3	Algorytmy	2
3.1	Modulo dla dowolnych X i P - bit po bicie	2
3.1.1	Opis algorytmu	2
3.1.2	Implementacja algorytmu	3
3.2	Algorytm mnożenia modułowego	4
3.2.1	Opis algorytmu	4
3.2.2	Implementacja algorytmu	5
4	Wnioski	5

1 Cele projektu

Celem projektu jest analiza oraz implementacja poszczególnych algorytmów związanych z arytmetyką modularną zaprezentowanych w artykule naukowym *Efficient Hardware Operations for the Residue Number System by Boolean Minimization* autorstwa Dana Gorodecky i Tomera Villi. W ramach projektu zostaną zaimplementowane oraz wytłumaczone algorytmy:

- Modulo dla dowolnych X i P - bit po bicie
- Algorytm mnożenia modułowego

2 Założenia projektu

Nasz projekt powinien skupiać się na implementacji oraz analizie algorytmów związanych z arytmetyką modularną. W ramach projektu powinny zostać zaimplementowane wcześniej wymienione algorytmy. Algorytmy implementowane będą w języku Python.

3 Algorytmy

Autorzy artykułu zaproponowali kilka algorytmów związanych z arytmetyką modularną. W ramach projektu skupiliśmy się na kilku z nich.

3.1 Modulo dla dowolnych X i P - bit po bicie

Algorytm ten pozwala na $X(\text{mod}P)$ z dowolnych liczb. Jego główną cechą jest to, że redukujemy liczbę X bit po bicie, aż dojdziemy do reszty z dzielenia.

3.1.1 Opis algorytmu

W artykule autorzy zaproponowali sposób obliczania oparty na następującej reprezentacji:

$$X = P \cdot Q + R = \tag{1}$$

$$= P \cdot 2^\delta \cdot q_\delta + P \cdot 2^{\delta-1} \cdot q_{\delta-1} + \dots + P \cdot 2^0 \cdot q_0 + R \tag{2}$$

$X(\text{mod}P) = R$, gdzie $X = (x_\psi, x_{\psi-1}, \dots, x_1)$ i δ jest określona nierówność $P \cdot 2^\delta < 2^\psi - 1 \leq P \cdot 2^{\delta+1}$.

Na przykład, $X = (x_{10}, x_9, \dots, x_1)$ i $P = 21$, przy $\delta = 5$. Wynosi:

$$\begin{aligned} X &= 21 \cdot Q + R = \\ &= 21 \cdot 2^5 \cdot q_5 + 21 \cdot 2^4 \cdot q_4 + 21 \cdot 2^3 \cdot q_3 + \\ &+ 21 \cdot 2^2 \cdot q_2 + 21 \cdot 2^1 \cdot q_1 + 21 \cdot 2^0 \cdot q_0 + R. \end{aligned}$$

Każdy kolejny iloczyn częściowy jest wejściem kolejnego bloku obliczeniowego. R jest wynikiem szóstego bloku oraz resztą z dzielenia przez 21. Jeśli chcemy policzyć $X(\text{mod } P) = R$, gdzie $X = 888$, a $P = 21$, to:

$$\begin{aligned} X_5 &\geq 21 \cdot 2^5, 888 \geq 672, X_4 = 888 - 21 \cdot 2^5 = 216; \\ X_4 &< 21 \cdot 2^4, 216 < 336, X_3 = 216; \\ X_3 &\geq 21 \cdot 2^3, 216 \geq 168, X_2 = 216 - 21 \cdot 2^3 = 48; \\ X_2 &< 21 \cdot 2^2, 48 < 84, X_1 = 48; \\ X_1 &\geq 21 \cdot 2^1, 48 \geq 42, X_0 = 48 - 21 \cdot 2^1 = 6; \\ X_0 &< 21 \cdot 2^0, 6 < 21, R = 6. \end{aligned}$$

W pierwszym kroku porównujemy X z $21 \cdot 2^5$. Następnie odejmujemy $21 \cdot 2^5$ od X i otrzymujemy $X_4 = 216$. W kolejnym kroku porównujemy X_4 z $21 \cdot 2^4$ i otrzymujemy $X_3 = 216$. Następnie odejmujemy $21 \cdot 2^3$ od X_3 i otrzymujemy $X_3 = 48$. W kolejnym kroku porównujemy X_2 z $21 \cdot 2^2$ i otrzymujemy $X_1 = 48$. Następnie odejmujemy $21 \cdot 2^1$ od X_1 i otrzymujemy $X_0 = 6$. W kolejnym kroku porównujemy X_0 z $21 \cdot 2^0$ i otrzymujemy $R = 6$.

Jak widać powyżej jest to prosta operacja odejmowania i porównywania, która jest wykonywana w pętli.

3.1.2 Implementacja algorytmu

Algorytm został zaimplementowany za pomocą trzech funkcji.

- **calc_length** - oblicza długość binarną liczby 'number' poprzez przesuwanie jej bitów w prawo i zliczanie ilości przejść, zwracając ostateczną długość.
- **get_delta** - funkcja obliczająca δ na podstawie parametrów l i P , sprawdzając warunek związanym z potęgami dwójki.
- **mod_bit_by_bit** - funkcja obliczająca resztę z dzielenia - w pętli obliczane są kolejne wartości X . Jeżeli $X_i \geq P \cdot 2^i$, to $X_{i-1} = X_i - P \cdot 2^i$, w przeciwnym wypadku $X_{i-1} = X_i$. Pętla kończy się, gdy wartość δ wynosi 0, a funkcja zwraca R jako resztę z dzielenia.

3.2 Algorytm mnożenia modułowego

Algorytm ten pozwala na mnożenie liczb w systemie resztowym. Jego główną cechą jest to, że dzielimy liczby na subwektory, a następnie mnożymy je w sposób opisany poniżej.

3.2.1 Opis algorytmu

Autorzy artykułu zaproponowali algorytm mnożenia modułowego, który pozwala policzyć $A \cdot B = R \pmod{P}$, gdzie $A = (A_\delta, A_{\delta-1}, \dots, A_1)$, $B = (B_\delta, B_{\delta-1}, \dots, B_1)$ A_δ oraz B_δ oznaczają najbardziej znaczące bity liczb A i B, a δ jest długością słów binarnych z których się składają. Np. $A = 13$ i $B = 14$ to $A = (1, 1, 0, 1)$ i $B = (1, 1, 1, 0)$, a $\delta = 2$ to $A_2 = (1, 1)$ oraz $B_2 = (1, 1)$. Staramy się dzielić liczby wejściowe na dwu, trzy lub czterobitowe subwektory. Odpowiednie pary subwektorów mnożymy używając poniższego wzoru:

$$R = \sum_{i=1}^{\delta} \sum_{j=1}^{\delta} A_i \cdot B_j \cdot \left(2^{m-(i+j-2)-3} \pmod{P} \right) = S_{temp}$$

S_{temp} nie może przekraczać $2^{3 \cdot \delta + 2}$

Przykład wykorzystania algorytmu:

Wybieramy dwie 6-bitowe liczby $A = 45$ and $B = 15$ oraz $P = 47$. Dzielimy je na 3-bitowe subwektory. Oznacz to, że $\delta = 2$
 $A_1 = (1, 0, 1) = 5$ $B_1 = (1, 1, 1) = 7$ $A_2 = (1, 0, 1) = 7$ $B_2 = (1) = 1$
 $A \cdot B = S(\text{mod}47)$

$$S_{temp} = A_1 \cdot B_1(\text{mod}47) + A_1 \cdot B_2 \cdot 2^3(\text{mod}47) + A_2 \cdot B_1 \cdot 2^3(\text{mod}47) + A_2 \cdot B_2 \cdot 2^6(\text{mod}47) = 5 \cdot 7(\text{mod}47) + 5 \cdot 1 \cdot 2^3(\text{mod}47) + 7 \cdot 7 \cdot 2^3(\text{mod}47) + 7 \cdot 1 \cdot 2^6(\text{mod}47) = 35(\text{mod}47) + 40(\text{mod}47) + 45(\text{mod}47) + 38(\text{mod}47) = 158$$

$158 > 2^{3 \cdot 2 + 2} = 128$, co oznacza, że musimy wykonać kolejną iterację aby zmniejszyć S_{temp} .

$$S_{temp} = 158 = (1, 0, 0, 1, 1, 1, 1, 0)$$

$$S_{temp1} = (1, 1, 0) \quad S_{temp2} = (0, 1, 1) \quad S_{temp3} = (1, 0)$$

$$S_{temp} = 6 + 3 \cdot 2^3(\text{mod}47) + 2 \cdot 2^6(\text{mod}47) = 6 + 24 + 34 = 64.$$

$64 \leq 128$, więc

$$S(\text{mod}47) = 64(\text{mod}47) = 17$$

3.2.2 Implementacja algorytmu

4 Wnioski

bla bla bla

Literatura

- [1] D. Gorodecky and T. Villa, "Efficient Hardware Operations for the Residue Number System by Boolean Minimization", Advanced Boolean Techniques, Minsk, Belarus, January, 2020, p. 237-258