SQRT_inverse - algorytm szybkiego obliczania odwrotności pierwiastka

Leszek Błach
Damian Brzeziński

2021

0 Algorytm

0.1 Wstęp

Projekt opiera się o korzystanie istniejącego rozwiązania algorytmu do obliczania odwrotnego pierwiastka z zadanej liczby. Algorytm ten ma wiele praktycznych zastosowań (np. normalizacja długości wektora), ponieważ liczenie odwrotności pierwiastka w tradycyjny sposób zajmuje dużo czasu. Celem projektu jest zaimplementowanie tego algorytmu na układzie FPGA wchodzącego w skład Zynq Evaluation and Development Kit, z wykorzystaniem oprogramowania Xilinx Vivado 2018.3.

0.2 Działanie algorytmu

W normalnej sytuacji, aby policzyć odwrotny pierwiastek z danej liczby, trzeba go po prostu obliczyć używając następującego kodu:

```
float y = 1 / sqrt(x);
```

Kod ten jednak jest bardzo nieoptymalny w swojej szybkości działania. Twórcy gry Quake Arena stworzyli algorytm, który pozwolił zacznie przyśpieszyć tę operację. Jeżeli policzymy logarytm o podstawie 2 z odwrotności pierwiastka otrzymamy:

$$\log_2\left(\frac{1}{\sqrt{y}}\right) = \log_2\left(y^{-\frac{1}{2}}\right) = -\frac{1}{2}\log_2(y) \tag{0.1}$$

Wartość zmiennej typu floating point zapisana jest na 32 bitach w postaci:
[31] - bit znaku $[S]$
$[30:23]$ - eksponenta $[E]$
$[22:0]$ - mantysa $[M]$
Wartość liczby zapisanej w tym formacie obliczana jest jako:

$$(-1)^S \cdot \left(1 + \frac{M}{2^{23}}\right) \cdot 2^{E-127} \tag{0.2}$$

Wykorzystując przybliżenie $\log_2(1+x) \approx x + \mu$ dla $x < 1$, gdzie μ to współczynnik korygujący błąd przybliżenia, oraz zakładając, że $x > 0$ oraz obliczając logarytm o podstawie 2 z tej wartości otrzymujemy:

$$\log_2\left(\left(1 + \frac{M}{2^{23}}\right) \cdot 2^{E-127}\right) = \log_2\left(1 + \frac{M}{2^{23}}\right) + E - 127 \approx \frac{M}{2^{23}} + \mu + E - 127 \tag{0.3}$$

Przekształcając to równanie otrzymamy:

$$\frac{1}{2^{23}}\left(M + 2^{23} \cdot E\right) + \mu - 127 \tag{0.4}$$

Wartość $M + 2^{23} \cdot E$ jest reprezentacją bitową dodatniej liczby w formacie floating point. Zatem reprezentacja bitowa liczby zmiennoprzecinkowej jest jej przeskalowanym logarytmem.

```
1  float Q_rsqrt( float number )
2  {
3          long i;
4          float x2, y;
5          const float threehalfs = 1.5F;
6
7          x2 = number * 0.5F;
8          y  = number;
9          i  = * ( long * ) &y;                       // evil floating point bit level hacking
10         i  = 0x5f3759df - ( i >> 1 );               // what the duck?
11         y  = * ( float * ) &i;
12         y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
13 //      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
14
15         return y;
16 }
```

W kodzie przedstawionym wyżej przedstawiono oryginalny algorytm z gry Quake Arena. W 9 linii bitowa reprezentacja liczby zmiennoprzecinkowej zostaje przekonwertowana na zmienną typu long. W 10 linii następuje operacja przedstawiona w równaniu (0.1): `- ( i >> 1 )` to zanegowana reprezentacja bitowa danej liczy zmiennoprzecinkowej podzielona przez 2. Liczba heksadecymalna `0x5f3759df` to obliczona wartość o którą trzeba przeskalować logarytm. W 11 linii następuje konwersja odwrotna do liczby zmiennoprzecinkowej. W tym momencie jest już obliczona przybliżona wartość odwrotnego pierwiastka. W 12 i 13 linii wykonywane jest przybliżenie metodą Newtona. Mając równanie $f(y) = \frac{1}{y^2} - x$ oraz jego pochodną $f'(y) = \frac{-2}{y^3}$ można wyznaczyć kolejne przybliżenia wartości odwrotnego pierwiastka: $y_{n+1} = y_n \cdot (1.5 - \frac{x}{2}y^2)$. Wartość zmiennej `y` jest na tyle blisko dokładnej wartości, że wystarczy jedna iteracja metodą Newtona aby uzyskać zadowalający wynik.