Hardware reciprocal using SRT

Niels Möller

2017

1 Introduction

To compute the reciprocal using Newton iteration results in a very big circuit. To get a smaller circuit with reasonable performance for a given size, SRT division is a nice alternative. These notes cover the simplest radix-2 variant.

1.1 Notation

Our application is computing the reciprocal. We have a word size ℓ (we will use $\ell = 64$), and implied base $B = 2^\ell$. The divisor D is assumed normalized, $B/2 \leq D < B$. We want to compute the quotient

$$Q = \lfloor (B^2 - 1)/D \rfloor$$

Q is a 65-bit number, with most significant bit always one. We represent the partial remainder as a fractional number, with the binary point located so that D is added or subtracted to the integer part of P. Initially, we set

$$P_0 = B - 1/B = B - 1 + (B - 1)/D$$

The first quotient bit is then $q_0 = 1$, and we form the next partial remainder as

$$P_1 = 2(P_0 - D)$$

which lies in the range $0 < P_1 < B$.

2 SRT division

The SRT division algorithm (for radix 2) works by computing a partial remainder sequence P_k as

$$P_{k+1} = 2(P_k - q_k D)$$

where the quotients q_k are selected so that the sequence P_k stays within a bounded interval. For basic radix 2 SRT, quotients are selected from the set $\{-1, 0, 1\}$, and P_k is bounded by $|P_k| < 2D$.

Since we will be using two's complement arithmetic, it's more convenient to use a slightly asymmetric interval, $-M \leq P_k < M$. For the moment, drop the k subscript, and first examine the case $M = 2D$. The quotient is not uniquely determined; instead we get the following constraints:

$q = 0$: Possible when $-D \leq P < D$.

$q = 1$**:** Possible when $P \geq 0$.

$q = -1$**:** Possible when $P < 0$.

The overlapping intervals is what enables efficient implementation: We can select a working q based only on examining the top few bits of P.

3 Representation of P

We will only represent the integer part explicitly; since the fraction is initially ℓ ones, we can handle it by just shifting in a one bit in each iteration.

Since $|P_k| \leq 2D < 2B = 2^{\ell+1}$, we can represent P_k as an $\ell + 2$-bit two's complement integer. To select a working q_k, it is sufficient to examine the top three bits of P_k: If $P_k = 000\ldots$, then $0 \leq P < B/2 \leq D$, and we can choose $q_k = 0$. And in all other cases, $P \neq 0$ and with known sign, so we can choose q_k from the sign bit of P.

But to limit latency when adding or subtracting D, we will represent all but the top few bits of P_k using a redundant "carry save" representation. So we set

$$P_k = S_k + C_k$$

where S_k is a $\ell + 2$ bits, and C_k is a few bits smaller. The value of the bits of P_k are then the corresponding top bits of S_k plus any carry from adding in the smaller C_k. To accommodate the unknown carry when going from S_k to P_k, we need one more bit when selecting q_k. I.e., C_k can be $\ell - 2$ bits, 4 bits smaller than S_k.

This adds a complication: If $P_k = 0111... \approx 2B$, we must select $q_k = 1$, but if $P_k = 1000... \approx -2B$, we must select $q = -1$. And if the top bits of S_k are 0111, which of these cases we get depends on the carry, which we don't want to compute.

Since we have $|P_k| \leq 2D$, the ambiguity is possible only for D close to B. One solution is to use a smaller M in this case. If we can ensure that $P < 7B/4$, then $P = 0111\ldots$ is no longer possible.

4 Narrowing the range

So let us set $M = \min(2D, 7B/4)$. Then we rule out the border line values of the top four bits of P, since $P = 0111\ldots$ implies $P \geq M$ and $P = 1000\ldots$ implies $P < -M$.

To stay within this narrower range, the quotient selection constraints get a little stricter,

$q = 0$**:** Possible when $-M/2 \leq P < M/2$.

$q = 1$**:** Possible when $P \geq \max(0, D - 7B/8)$.

$q = -1$**:** Possible when $P < -\max(0, D - 7B/8)$.

If we tighten this a little bit more, we get the following constraints which are sufficient for all values of D:

$q = 0$**:** Possible when $-B/2 \leq P < B/2$.

$q = 1$: Possible when $P \geq B/8$.

$q = -1$: Possible when $P \leq -B/8$.

This lets us define quotient selection based on the top S_k bits only. Let h denote the value of the four most significant bits of S_k, interpreted as a two's complement number.

We have alerady ruled out the problematic case $h = 7$. So we can assume that $-8 \leq h \leq 6$, and each value corresponds the the following ranges for S_k and P_k:

$$hB/4 \leq S_k < (h+1)B/4$$
$$hB/4 \leq P_k < (h+2)B/4$$

We can therefore use the following rules:

$-8 \leq h \leq -3$: Then $-7B/4 \leq P_k < -B/4$, use $q_k = -1$. Note that $h = -8$ can happen only if we do get a carry from the addition of C_k.

$-2 \leq h \leq 0$: Then $-B/2 \leq P_k < B/2$. Use $q_k = 0$.

$1 \leq h \leq 6$: Then $B/4 \leq P_k < 7B/4$. Use $q_k = 1$.

$h = 7$: Can't happen.

5 Final processing

The iteration $P_{k+1} = 2(P_k - q_k D)$ can be turned around to

$$P_k = q_k D + P_{k+1}/2$$

After $\ell + 1$ iterations, we have

$$P_0 = \sum_{k=0}^{\ell} q_k 2^{-k} D + P_{\ell+1}/2^{\ell+1}$$

Recall that $P_0 = B - 1/B$ and multiply by B, to get

$$B^2 - 1 = \sum_{k=0}^{\ell} q_k 2^{\ell-k} D + P_{\ell+1}/2$$

Define

$$Q' = \sum_{k=0}^{\ell} q_k 2^{\ell-k} D \qquad\qquad R' = P_{\ell+1}/2$$

Then $B^2 - 1 = Q'D + R'$, and we have $-D \leq R < D$. Hence $Q = Q' + [R < 0]$.