David Baca

Fast 32-bit Division on the DSP56800E

Minimized nonrestoring division algorithm

Introduction

The processor core DSP56800E features several 32-bit registers. It may appear that performing a 32-bit integer division should be a simple task, however, if you're not an expert on binary division, you may need more support. DSP56800E uses an instruction called DIV to perform a division. The instruction represents a divide iteration calculating 1 bit of the division result. The instruction description along with 16-bit division algorithms can be found in the DSP56800E Reference Manual.

DIV accepts a 32-bit dividend, a 16-bit divisor and produces a 16-bit quotient and a 16-bit remainder after 16 iterations. The iteration step operates on two 32-bit operands where the 16-bit divisor occupies the upper 16 bits of the operand. The algorithm used for calculating the quotient and the reminder is Nonrestoring Division Algorithm (NrDA).

If we were to implement NrDA for 32-bit division, we would need 64-bit operands. Implementing NrDA with 64-bit operands on a 16-bit processor such as the DSP56800E is cumbersome because it needs support for 64-bit arithmetic. Another option is to apply the well known Newton-Rhapson iterative method that requires only 32-bit arithmetic. Nevertheless, this method employs division and therefore requires an abundance of tweaking to achieve the desired accuracy. Furthermore, we can't be sure that all pairs (dividend, divisor) have the correct result calculated unless the implementation is tested for almost all possible input pairs.

Minimized nonrestoring division algorithm (MNrDA) efficiently implements 32-bit non-restoring algorithm on 32-bit arithmetic. Since the DIV instruction on the DSP56800E also accepts the divisor as a 32-bit number, we can implement the NrDA algorithm effectively.

This article will:

- Introduce the fundamental restoring division algorithm
- Explain the nonrestoring division from the restoring algorithm
- Describe how to eliminate unnecessary iterations from NrDA to arrive at the minimized version of NrDA
- Provide an example implementation of MNrDA for the processor core DSP56800E

The Nonrestoring Division Algorithms

Integer division notation

Integer division can be defined as follows:

$$N = Q \bullet D + R$$

Where

- N is the numerator (dividend)
- D is the denumerator (divisor)
- Q is the quotient
- R is the remainder

For the sake of simplicity, we assume that N,D,Q is positive. R is positive by definition and $0 \le R < D$. Further, each operand can be dismantled into individual bits as follows $Q = \sum_{j=0}^{m-1} Q_j \cdot 2^j$.

Restoring Division Algorithm

Nonrestoring Division Algorithm (NrDA) comes from the restoring division. The restoring algorithm calculates the remainder by successively subtracting the shifted denominator from the numerator until the remainder is in the appropriate range. Now we can derive the algorithm for restoring division of m-bit integers.

The algebraic equation of the remainder calculation is:

$$R = N - (D \cdot Q)$$

We dismantle Q into bits.

$$R = N - D \cdot \sum_{j=0}^{m-1} Q_j \cdot 2^j$$
$$= N - D \cdot \sum_{j=0}^{m-1} Q_j \cdot D \cdot 2^j$$

This can be viewed as subtracting denominator shifted by j bits left, $D \cdot 2^j$, from N for each $Q_j = 1$. The subtraction can only occur if it still yields a positive reminder ($Q_j = 1$) given the previous subtractions. As a result, we can formulate the restoring algorithm as follows. We set:

$R(m) = N$ where m is the size of integers in bits, for example 16, 32

We then calculate $R(j)$ and Qj for $j = m - 1 \ldots 0$ as follows:

$$Z = R(j+1) - D \cdot 2^j$$
$$\text{if } Z \geq 0 \qquad Q_j = 1,\ R(j) = Z$$
$$\text{else} \qquad Q_j = 0,\ R(j) = R(j+1)$$

After calculating $R(0)$, $R(0)$ is the actual remainder and Q contains the resulting quotient.

In order to carry out the restoring algorithm, we need either the temporary variable Z or we have to restore the remainder at each step the subtraction yields a negative result. This is because we are determining Q_j and $R(j)$ calculation in one step. The nonrestoring algorithm eliminates this concurrency by determing $R(j)$ calculation from Q_{j+1}.

Nonrestoring Division Algorithm

In the nonrestoring algorithm, we use the negative result for the next iteration. To formulate the algorithm we start from the restoring algorithm. Suppose that we obtain a negative remainder at the k-th step.

$$R(k) = R(k+1) - D \cdot 2^k$$

Obviously $Q_k = 0$ and therefore at the very next step we should compute:

$$R(k-1) = R(k+1) - D \cdot 2^{k-1}$$

By using the result $R(k)$ and for $Q_k = 0$ we use addition.

$$\begin{aligned} R(k-1) &= R(k) + D \cdot 2^{k-1} \\ &= R(k+1) - D \cdot (2^k - 2^{k-1}) \\ &= R(k+1) - D \cdot 2^{k-1} \end{aligned}$$

Hence, we formulate the iteration in the non-restoring algorithm as follows:

$$\begin{aligned} &R(j) = R(j+1) - D \cdot 2^j && \text{if } Q_{j+1} = 1 \\ &R(j) = R(j+1) - D \cdot 2^j && \text{if } Q_{j+1} = 0 \\ &Q(j) = 1 && \text{if } R(j) \text{ is positive} \\ &Q(j) = 0 && \text{if } R(j) \text{ is negative} \end{aligned}$$

We combine the two equations above to express the iteration step algebraically as:

$$R(j) = R(j+1) + (1 - 2 \cdot Q_{j+1}) \cdot D \cdot 2^j$$

We start the algorithm by initializing $R(m) = N$ and $Q^m = 1$. After performing $m - 1$ iterations we obtain $R(0)$.

$$R(0) = N - (2^{m-1} - \sum_{j=0}^{m-1} (1 - 2 \cdot Q_{j+1}) \cdot 2^j) \cdot D$$
$$R(0) = N - (2^0 - \sum_{j=0}^{m-1} Qj \cdot 2^j) \cdot D$$

Clearly, $R(0)$ is not the correct actual remainder. Therefore, at the end of the algorithm we need to perform one additional calculation to obtain the remainder.

$$R = R(0) + (1 - Q_0) \cdot D$$

Finally, we can summarize the non-restoring algorithm as follows:

$$\begin{aligned} &R(m) = N,\ Q_m = 1 \text{ (initial state)} \\ &\text{for } j = m-1 \text{ to } 0 \\ &\quad R(j) = R(j+1) - D \cdot 2^j \text{ if } Q_{j+1} = 1 \\ &\quad R(j) = R(j+1) + D \cdot 2^j \text{ if } Q_{j+1} = 0 \\ &\quad Q_j = 1 \text{ if } R(j) \geq 0 \\ &\quad Q_j = 0 \text{ if } R(j) < 0 \\ &\text{end for} \\ &\quad R = R(0) + \neg Q_0 \cdot D \end{aligned}$$

The nonrestoring algorithm requires shifting D $m - 1$ times. In particular for 32-bit division, D has to be shifted 31 bits left. So, 64-bit arithmetic is needed to perform NrDA. In general, one needs $(2*m)$-bit arithmetic to execute NrDA for m-bit division.

Minimized Nonrestoring Division Algorithm

Minimized Nonrestoring Division Algorithm (MNrDA) is designed to execute NrDA for m-bit numbers on m-bit arithmetic.

Proposition 1 If n, d and q are the indices of the first significant bits of N, D and Q respectively, then $n - d \geq q$,

Proof of proposition 1: For m-bit integers we have:

$$N = Q \cdot D + R$$
$$\sum_{j=0}^{m-1} N_j \cdot 2^j = \sum_{j=0}^{m-1} D_j \cdot 2^j \cdot \sum_{j=0}^{m-1} Q_j \cdot 2^j + R$$
$$2^n + \sum_{j=0}^{n-1} N_j \cdot 2^j = (2^d + \sum_{j=0}^{d-1} D_j \cdot 2^j) \cdot (2^q + \sum_{j=0}^{q-1} Q_j \cdot 2^j) + R$$

The condition $n - d \geq q$ immediately follows.

Corollary 1 Let N be the numerator and D be the denominator of an integer division. Let n and d be the indices of the first significant bits of N and D, respectively. Then $Q_j = 0$ for all $j > n - d$.

Proof of corollary 1: The corollary is a direct result of proposition 1

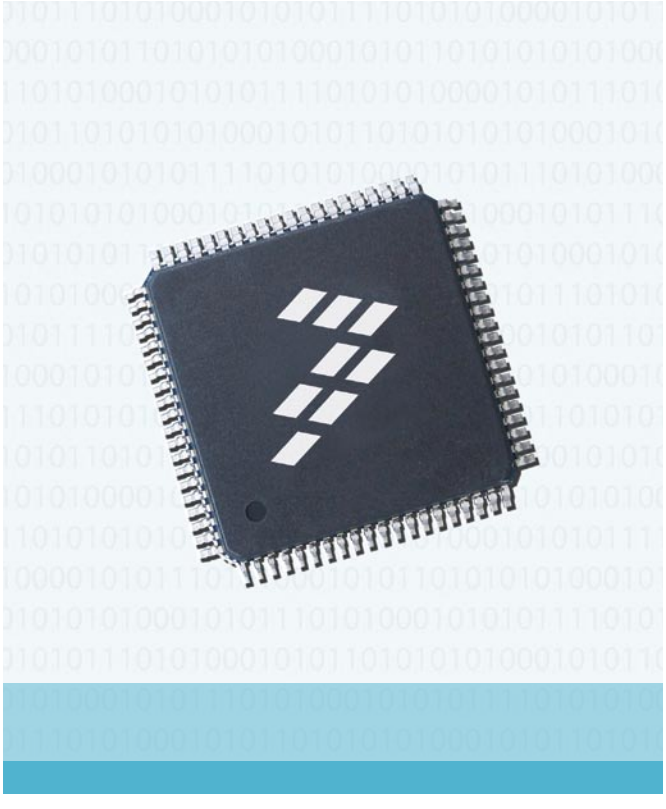
If we apply corollary 1 to NrDA, we can jump in the algorithm to $R(n-d)$ because previous iterations resulted in the quotient bits of the value 0.

$$\begin{aligned} R(n-d) &= N - D \cdot 2^{m-1} + \sum_{j=n-d}^{m-2} D \cdot 2^j \\ &= N - D \cdot 2^{n-d} \\ &= N - 2^{n-d} \cdot (2^d + \sum_{j=0} D_j \cdot 2^j) \\ &= N - (2^n + \sum_{j=0}^{d-1} D_j \cdot 2^{j+n-d}) \end{aligned}$$

Since the first calculated partial remainder is $R(n-d)$, we only need to shift D $n-d$ times effectively moving the first significant bit of D to position n. Hence, since $m - 1 \geq n$, we need only m bits to store shifted D.

MNrDA doesn't decrease the number of iterations for the worst case scenario ($n = m - 1$, $d = 0$). Since timing analyses are typically based on the worst case execution times, we can simply assume $n = m - 1$. Thus, MNrDA is performed as follows:

1. Calculate the leading zeroes l_0 of D.
2. Shift D left by l_0 bits.
3. Execute Nonrestoring Division Algorithm starting from $R(l_0)$ and setting $Q_j = 0$, $j = l_0 + 1 \ldots m - 1$



MNrDA Implementation for Signed Division on the DSP56800E

The DIV instruction on the DSP56800E represents the iteration of 16-bit restoring division. The instruction accepts 32-bit dividend and 16-bit divisor. The divisor is applied as shifted 16 bits left and instead of shifting the divisor right in each iteration, the dividend is shifted left. The resulting bits Q_j are shifted in during each iteration. As a result, after performing 16 iterations, the remainder $R(0)$ is contained in the upper 16 bits of the dividend. This can be algebraically illustrated as follows. One iteration step is:

$$\begin{aligned} R'(j+1) &= 2 \cdot R'(j+1) \pm D \cdot 2^n \\ &= 2^{n-j} \cdot R(j+1) \pm D \cdot 2^n \\ &= 2^{n-j} \cdot (R(j+1) \pm D \cdot 2^n) \\ &= 2^{n-j} \cdot R(j) \end{aligned}$$

This ultimately leads to:

$$R'(0) = 2^n \cdot R(0)$$

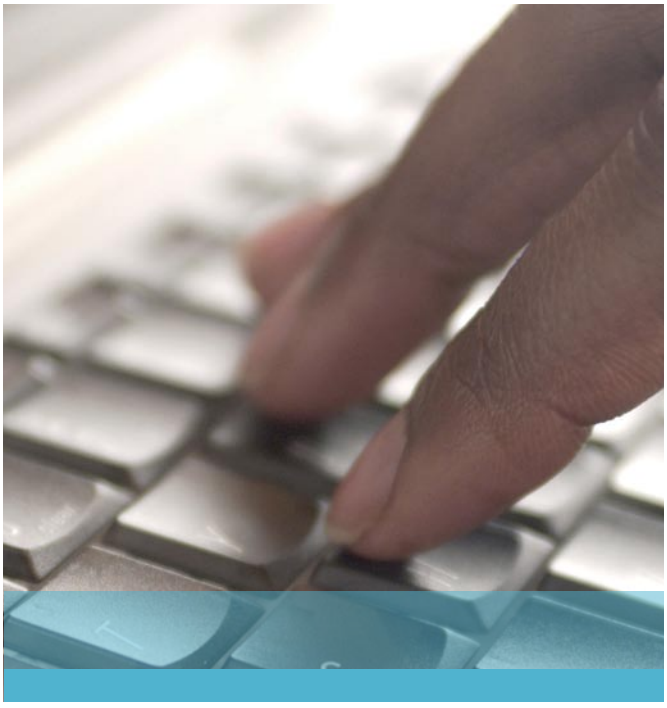
In our case $n = 16$ and therefore $R(0)$ is contained in the upper 16 bits. The quotient occupies the lower 16 bits.

In order to perform full 32-bit division, we supply the DIV instruction with a 32-bit divisor shifted $n - d$ bits left where n and d are the most significant bits of the dividend and the divisor respectively.

$$\begin{aligned} R'(n-d) &= R(n-d) \\ &= N - D \cdot 2^{n-d} \\ R'(n-d-1) &= 2 \cdot R(n-d) - D \cdot 2^{n-d} \\ &= 2 \cdot (R(n-d) - D \cdot 2^{n-d-1}) \\ &= 2 \cdot R(n-d-1) \\ &\vdots \\ R'(n-d-j) &= 2^j \cdot R(n-d-j) \\ &\vdots \\ R'(0) &= 2^{n-d} \cdot R(0) \end{aligned}$$

For this reason, $R(0)$ is contained in the upper $m - 1 - (n - d)$ bits and the quotient in the lower $n - d$ bits. If we put $n = m - 1$, then $R(0)$ resides in the d upper bits with its LSB at the (n-d)th bit. We apply mask to the DIV destination operand in order to retrieve the quotient. In our case, $m = 31$ as we assume positive dividend and divisor for the actual division. We present four-quadrant division employing MNrDA that produces the quotient. Since the complexity of retrieving the remainder from MNrDA is more or less equivalent to multiplying the quotient and subtracting it from the dividend, we don't produce the remainder. The worst case time execution of this algorithm is 60 cycles.

```
;  A = signed dividend (numerator)
;  B = signed devisor (denominator)
;  A1:A0 / B1:B0
;  The routine uses the following registers : A, B, D, X0, Y

   tfr     B,Y       ;  Y=den

   ;  the quotient sign
   eor.l   A,B       ;  MSB of B holds the sign bit

   ;  prepare R(q) and den
   abs     A         ;  abs (num)
   abs     Y         ;  abs (den) , so we can calculate R(q)
                     ;  DIV accepts negative S but for
                     ;  R(q) calculation we hardcode subtraction

   ;  shifted denominator and the first bit
   clb     Y,X0      ;  calculate leading zeroes
   asll.l  X0,Y      ;  Y=den*2^q
   sub     Y,A       ;  A=R(q), Carry=1 if Y>A

   ;  calculate the bit q; 1 if R(q)>=0
   bftstl  #$8,A2    ;  set Carry if positive

   ;  divide loop
   rep     X0        ;  calculate the bits q–1. . . 0
   div     Y,A       ;  Carry has the least significant bit

   rol.l   A         ;  last Carry has to be rolled in
                     ;  DIV would destroy the highest bit

   ;  mask off bits 0. . . q–1
   eor.l   Y,Y       ;  clear Y
   add.l   #2,Y      ;  for X0=0, we calculate the bit 0
   asll.l  X0,Y
   dec.l   Y         ;  Y contains the mask
   and.l   Y,A       ;  apply the mask

   ;  apply the sign bit
   brclr   #$8000, B1, Qpositive
   nop
   neg     A
Qpositive:
```



Conclusion

In this article, we showed that Freescale processors with the DSP56800E core are capable of performing a full precision 32-bit division. Since the DSP56800E DIV instruction can accept 32-bit number as the divisor, we can implement the minimized version of the nonrestoring division algorithm (MNrDA) to carry out 32-bit division. The DIV instruction was originally designed for 16-bit non-restoring division where the divisor initial shift is automated to full 16 bits.

Unlike with 32-bit division, applying MNrDA to 16-bit division would rather increase the worst case execution time as the reduction of the number of iteration is more or less consumed by calculating and performing the initial divisor shift. On the other hand, for 32-bit division we are running out of the register sizes on a 16-bit processor; therefore MNrDA is the best option. Another benefit of this option is that the effect of the reduction of iterations is leveraged by the fact that 32 iterations are needed for the pure NrDA.

David Baca is an application engineer at Freescale. He has been with the company for almost two years. He holds masters degrees in electrical engineering and business administration and is a PhD candidate in artificial intelligence. Before joining Freescale, Baca worked on R&D projects sponsored by agencies such as NASA and the Air Force.