Design of Double-Precision Floating-Point Division and Square Root Based on SRT Algorithm

Chen Ji-yang, Peng Yuan-xi, Lei Yuan-wu, and Deng Zi-ye

Department of Computer, National University of Defense Technology, Hunan, Changsha, China (824750961@qq.com)

***Abstract*—In the process of scientific computing, digital signals processing, communication and image processing, division and square root are the widely used basic operation. This article designs a unit of floating-point division and square root based on SRT4 algorithm. It reduces the iterative division and square root calculation delay by using parallel processing of quotient and residue addition and the on-the-fly conversion technology. At the same time, it reduces hardware resources and saves area by adopting reuse the lookup table and the adder. Experiments show that the cell area is 9812.3087um^2, the power is 10.7854mW, the cycles per division instruction is 31, the cycles per square root instruction is 28, comprehensive frequency up to 2.5GHz with 40nm technology library and the timing constraint 400ps.**

***Key words*—SRT4, floating-point division and square root, on-the-fly conversion**

基于 SRT 算法的双精度浮点除法和平方根运算的设计

陈际阳 彭元喜 雷元武 邓子椰

国防科技大学计算机学院，长沙，湖南，中国

摘 要 在科学计算、数字信号处理、通讯和图像处理等应用中，除法和平方根运算是常用的基本操作。本文基于SRT4算法，设计一个同时支持IEEE-754标准的双精度浮点除法和平方根的部件。首先提出商选择和余数加法的并行处理方法，通过商飞速转换技术，来降低除法/平方根的计算延时，同时复用查找表和加法器，减少硬件资源开销，节省面积。综合表明，在40nm工艺、时序约束400ps的条件下，本设计综合频率可达2.5GHZ，cell面积为9812.3087 um^2，功耗为10.7854mW，除法运算计算延时为31拍，平方根运算计算延时为28拍。

关键词 SRT4，浮点除法/平方根，商飞速转换

1．引言

浮点除法和平方根运算是科学计算、数字信号处理、通讯和图像处理等应用中的基本操作。高性能浮点除法/平方根部件对处理器性能的提高越来越重要。SRT 作为一种主要的浮点除法/平方根算法，已经广泛使用在现代微处理器中, 比如 Intel Pentium CPUs[1]、ARM 处理器[2]和 IBM FPUs[3]采用 SRT4 算法,Intel Core2 处理器[1]采用 SRT16 算法。同时，各种应用的处理器对计算速度、芯片面积以及功耗大小的要求也对 SRT 算法的实现提出了挑战。本文设计并实现高性能低开销的浮点除法/平方根部件以解决除法/平方根计算周期长、硬件开销大的问题。

目前对 SRT4 算法的研究已取得许多成果。文献[4]在协处理器 LSC87 中提出串行除法和平方根计算，最大限度利用通用数据路径完成 SRT 算法实现，但工作频率低，不适合高速处理器的实现；文献[5]分别用基 4 和基 16 算法实现了 SRT 除法/平方根，基 4 算法通过比较器预测下一商值，面积消耗较大，一种基 16 的算法通过单纯扩大查找表实现，但这样硬件复杂度会急剧增加，另一种基 16 方法由两个 SRT4 算法在一个时钟周期内迭代而成，但这样的方法容易引起时序紧张；文献[6]采用牛顿迭代来计算平方根，采用 Goldschmidt 来计算除法，虽然将两种运算集成在一个部件中，却无法实现硬件复用，面积较大。

本文基于 SRT4 算法，设计了一种高性能低开销的双精度浮点除法/平方根运算模块，在一个部件中支持 IEEE-754 标准的除法和平方根运算。在深入研究 SRT4 算法的基础上，提出商查找选择和余数加法的并行处理方法，

航天科学基金（项目号：2013ZC88003）
国防科技大学科研计划项目（项目号：JC120201）

降低关键路径延时，并通过比较商的冗余表示和商的飞速转换，选用周期较短的商飞速转换来降低迭代的计算延时，提高频率。另外本文将除法和平方根运算集成在同一个部件中，可复用查找表和加法器，减少硬件资源开销，节省面积。

本文将从以下结构对文章进行阐述：第 2 节， SRT 算法及分析；第 3 节，介绍除法和平方根算法的总体结构；第 4 节，比较商冗余表示和商飞速转换的结果，比较单独除法、单独平方根和除法/平方根算法的结果，比较其他处理器与本设计的除法/平方根部件的频率、延时；第 5 节，得出结论。

2．SRT 算法及分析

SRT[7]算法是一种数字循环算法，当它以 r=2k 为基时，每次迭代可产生非冗余二进制结果的 k 位，同时生成相应的部分余数。SRT 算法的关键是基数的大小，它决定了每次循环可以得到的商的位数，每次迭代得到的位数越多则需要迭代的次数就越少，运算所需的周期就越少，从而提高算法性能。但是增大基数会增加设计的硬件复杂度，同时也会增加部件的延迟和面积。

$$q=\sum_{i=1}^{n} q_i r^{-i} \tag{1}$$

迭代主要由移位加操作实现，计算结果采用冗余数表示，结果数值位 $q_i \in \{\bar{a},\bar{a}+1\cdots,-1,0,1,\cdots,a-1,a\}, a \le r-1$。采用冗余数的优点是可用截短的输入值来估计结果值中对应数位，从而简化设计，缩短操作时间；而且当本次迭代的结果位 q_i 有 1 位误差时，可通过冗余在下一次迭代中修正。本文 SRT 算法基值为 4，所以选择 a=2（如选择 a=3，对除数 d 生成 3d 不易实现）[4]。根据 IEEE754 标准[8]，单精度浮点数的尾数为 23 位，双精度浮点数的尾数为 52 位。假如采用基数为 4 的 SRT 算法，实现单精度浮点数的除法至少需要 $23/\log_2 4=12$ 个指令周期，而实现双精度浮点数的除法至少需要 $52/\log_2 4=26$ 个周期。

2.1 SRT 除法算法及分析

2.1.1 商数字集合和选择区间

为得到高速算法，我们使用一个对称的集合表示商数字的取值范围，即：

$$q_{j+1}=k \in \{\bar{a},\bar{a}+1\cdots,-1,0,1,\cdots,a-1,a\}$$

数字集合的冗余由冗余因子 ρ 的值决定，定义冗余因子 ρ：

$$\rho=\frac{a}{r-1}, \text{其中} \rho>\frac{1}{2} \tag{2}$$

在除法运算中，x、d 为被除数和除数的尾数，并定义部分余数 p[j]=r_j（x-dQ[j]），其中 j 为迭代序数，Q[j]为第 j 步迭代出的商值，p[j]为第 j 次迭代的部分余数，则迭代公式为：

$$p[j+1]=4*p[j]-d*q_{j+1} \tag{3}$$

余数边界为：

$$|p[j]|<d \tag{4}$$

设 U_k、L_k 为当 $q_{j+1}=k$ 且保证下次循环的部分余数在选择范围时，部分余数相应的最大值和最小值，q_{j+1} 的选择区间为：

$$U_k=(\rho+k)d, L_k=(-\rho+k)d \tag{5}$$

2.1.2 商选择函数

商选择函数部件位于除法算法设计的关键路径上，该部分实现会影响到整个设计的时间延迟。

在除法计算中，首先将除数 d 划分成长度为 $2^{-\delta}$ 的小区间 $[d_i,d_{i+1})$，且 $d_1=1/2$， $d_{i+1}=d_i+2^{-\delta}$。这样，区间就用除数的前 δ 位小数表示。在小区间 $[d_i,d_{i+1})$ 里，如果 $m_k(i) \le rP_j < m_{k+1}(i)$，则 $q_{j+1}=k$。设选择常数 m_k 的最小长度为 2^c，则 $m_k(i)=A_k(i)2^{-c}$，其中 $A_k(i)$ 为整数，m_k 定义如下：

$$\begin{cases} m_k(i) \ge \max\{L_k(d_i), L_k(d_{i+1})\} \\ m_k(i) \le \min\{U_{k-1}(d_i), U_{k-1}(d_{i+1})\} \end{cases} \tag{6}$$

将 $\rho=\frac{2}{3}, k \in \{-2,-1,0,1,2\}$ 代入 U_k 和 L_k 的表达式即可得除法的查找表。

q_{j+1} 的值可根据 rp[j]和除数 d 由查找表确定，查找表输入 rp[j]和 d 的截短值。

2.1.3 商的转换

由于在迭代中计算结果都是用冗余数表示，我们需要进行商的转换，将其转换成非冗余二进制形式，在传统方法中，我们将求出的商位按符号放在正商寄存器和负商寄存器中，最后的商是在迭代结束后，经过正商寄存器和负商寄存器的减运算得到。但这种方式需要通过加法器，速度较慢，为提高运算速度，我们借鉴文献[9]采用商的飞速转换。

我们设置寄存器 Q（商值）和 QM（商值减 1）。

$$Q[j]=\sum_{i=1}^{k} q_i r^{-i} \tag{7}$$

$$QM[j]=Q[j]-r^{-j} \tag{8}$$

随着每次迭代结果的产生，Q 和 QM 按以下形式进行更新（平方根的更新把 q_{j+1} 用 s_{j+1} 代替）：

$$Q[j+1]=\begin{cases}Q[j]+q_{j+1}r^{-(j+1)}, q_{j+1}\geq 0\\ QM[j]+(r-|q_{j+1}|)r^{-(j+1)}, q_{j+1}<0\end{cases} \tag{9}$$

$$QM[j+1]=\begin{cases}Q[j]+(q_{j+1}-1)r^{-(j+1)}, q_{j+1}\geq 0\\ QM[j]+((r-1)-|q_{j+1}|)r^{-(j+1)}, q_{j+1}<0\end{cases} \tag{10}$$

Q 和 QM 的变化如表 1 所示。

表 1 Q 和 QM 的变化

q_{j+1}	Q[j+1]	QM[j+1]
2	{Q[j],2}	{Q[j],1}
1	{Q[j],1}	{Q[j],0}
0	{Q[j],0}	{QM[j],3}
-1	{QM[j],3}	{QM[j],2}
-2	{QM[j],2}	{QM[j],1}

2.1.4 加法器输入 F 的产生

在除法运算中，加法器输入 F 的产生比较简单。先通过移位器产生 F=d 或 F=2d，然后再通过加法器产生新的部分余数 p[j+1]=4*p[j]±F[j]。

2.1.5 舍入和规格化

SRT 算法中除数需规格化至[1/2,1)，而被除数作为部分余数的初始值需满足连续性条件：

$$-\rho d\leq x\leq \rho d \tag{11}$$

将被除数通过移位操作进行调整：

$$x=s.001x_1x_2..x_{52}\leq \rho d \tag{12}$$

同时调整指数以保证等值。除数按类似方法进行调整：

$$d=s.1d_1d_2..d_{52}00 \tag{13}$$

所以本文操作数尾数扩展为 56 位，其中 1 位符号位，53 位尾数（包括隐藏位），2 位空余位供移位使用。符号位存放每次迭代产生的新余数的符号，用以判断下一次迭代时加法器是加操作还是减操作；53 位尾数可以保证除法结果精度的基本要求；2 位空余位用以被除数和除数的移位规格化。

为保证商值的精度，商值的尾数同样应为 56 位，因此，在若干次迭代求得商值以后需遵循一定的策略进行舍入和规格化操作以满足 IEEE-754 对浮点格式的要求。本文采用 IEEE-754 定义的就近舍入策略，舍入操作按表 2 执行[10]。

迭代完成后对商值进行规格化，即对尾数执行前导零检测并左移相应位数，同时调整指数保证等值。

因此，将尾数扩展为 56 位即可满足 IEEE-754 标准的舍入和规格化要求，同时保证结果的正确性，也节省了芯片面积、降低功耗和延时。

表 2 就近舍入

编号	数值	舍入	误差	编号	数值	舍入	误差
1	0.00	0.	0	5	1.00	1.	0
2	0.01	0.	-1/4	6	1.01	1.	-1/4
3	0.10	0.	-1/2	7	1.10	1.+1	+1/2
4	0.11	1.	+1/4	8	1.11	1.+1	+1/4

2.2 SRT 平方根算法及分析

2.2.1 商数字集合和选择区间

在平方根运算中，定义部分余数 $w[j]=r^j(x-S[j]^2)$，其中，S[j]为第 j 步迭代出的平方根值，w[j]为第 j 次迭代的部分余数，则迭代公式为：

$$w[j+1]=4*w[j]-2S[j]s_{j+1}-s_{j+1}^2 4^{-(j+1)} \tag{14}$$

余数边界为：

$$-2\rho S[j]+\rho^2 r^{-j}<w[j]<2\rho S[j]+\rho^2 r^{-j} \tag{15}$$

sj+1 的选择区间为：

$$\begin{aligned}U_k&=2S[j](k+\rho)+(k+\rho)^2 r^{-(j+1)},\\ L_k&=2S[j](k-\rho)+(k-\rho)^2 r^{-(j+1)}\end{aligned} \tag{16}$$

2.2.2 商选择函数

在平方根运算中，如果 $m_k(i)\leq rw_j<m_{k+1}(i)$，则 $s_{j+1}=k$。选择常数 m_k 定义如下：

$$\begin{cases}m_k(i)\geq \max(L_k(I_i))\\ m_k(i)\leq \min(U_{k-1}(I_i))\end{cases} \tag{17}$$

所以一个可行结果的最小重叠区域为：

$$\min(U_{k-1}(I_i))-\max(L_k(I_i))\geq 0 \tag{18}$$

当 r=4 时，对所有的 j，没有一个唯一的选择函数，即 SRT 查询表中的选择常数要随 j 的变化而变化，可以找到

一个 J，把选择函数分为 j>J 和 j<J 来分析。经考虑选择 J=3，当 j=0,1,2 时，为简化设计，我们采用文献[11]中的方法，通过比较 j 在各取值时选择常数的区间，选择他们的交集，使对 j 的所有查询表中的选择常数为唯一值。

当 j≥3 时：当 k>0 时，

$$\begin{cases}\min(U_{k-1}(I_i)) = 2(2^{-1}+i*2^{-\delta})(k-1+\rho)\\ \max(L_k(I_i)) = 2(2^{-1}+(i+1)*2^{-\delta})(k-\rho)\end{cases} \tag{19}$$

当 k<0 时，

$$\begin{cases}\min(U_{k-1}(I_i)) = 2(2^{-1}+(i+1)*2^{-\delta})(k-1+\rho)\\ \max(L_k(I_i)) = 2(2^{-1}+i*2^{-\delta})(k-\rho)+(k-\rho)^2 r^{-(J+1)}\end{cases} \tag{20}$$

最坏情况是当 i=0 且 k=-a+1，在这种情况下根据 $\min(U_{k-1}(I_i))-\max(L_k(I_i))\geq 0$ 确定一个单独的选择函数：

$$(2\rho-1)-(\rho r-1)^2 r^{-(J+1)} \geq 2*2^{-\delta}(a-\rho) \tag{21}$$

当 r=4，ρ=2/3，J=3 时，得到

$$\frac{1}{3}-\frac{25}{36}4^{-3} \geq \frac{8}{3}2^{-\delta} \tag{22}$$

可取 δ =4。

将 $\rho=\frac{2}{3}, k\in\{-2,...,2\}, i\in\{0,1,...,7\}, J=3, \delta=4$ 代入 U_k 和 L_k 的表达式即可得平方根的查找表。

s_{j+1} 的值可根据 rw[j]和已产生的结果 S[j]由查找表确定，查找表输入 rw[j]和 S[j]的截短值。

遍历（19）（20）式可确定 c 最小值为 4，即 m_k 的最小长度为 1/16。

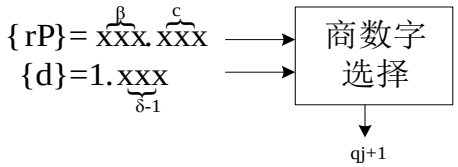

图 1 基于选择常数的商数字选择函数

图 1 中 β 是 rp[j]的整数位数，商数字选择模块中截短的 rp[j]（或 rw[j]）需要的输入位数为 $\log_2 4+c=6$ 位，对除数 d（或平方根 S[j]），由于小数第一位恒定为 1，所以只需 δ-1=3 位。所以 SRT 查找表输入中，rp[j]和 rw[j]输入位数为 6 位，d 和 S[j]输入位数为 3 位。

经过计算，发现除法和平方根的查找表可以复用，这可以简化逻辑并节约芯片面积。同时适用于除法和平方根计算的查找表如表 3 所示。

表 3 除法/平方根查找表

(1)真值=表中值/16，(2)真值=表中值/48

$[d_i,d_{i+1})$(1)	[8,9)	[9,10)	[10,11)	[11,12)
$L_2(d_{i+1}),U_1(d_i)$(2)	36,40	40,45	44,50	48,55
$m_2(i)$	12/16	14/16	15/16	16/16
$L_1(d_{i+1}),U_0(d_i)$(2)	9,16	10,18	11,20	12,22
$m_1(i)$	4/16	4/16	4/16	4/16
$L_0(d_{i+1}),U_{-1}(d_i)$(2)	-18,-8	-20,-9	-22,-10	-24,-11
$m_0(i)$	-4/16	-6/16	-6/16	-6/16
$L_{-1}(d_{i+1}),U_{-2}(d_i)$(2)	-45,-32	-50,-36	-55,-40	-60,-44
$m_{-1}(i)$	-13/16	-15/16	-16/16	-18/16
$[d_i,d_{i+1})$(1)	[12,13)	[13,14)	[14,15)	[15,16)
$L_2(d_{i+1}),U_1(d_i)$(2)	52,60	56,65	60,70	64,75
$m_2(i)$(1)	18/16	19/16	20/16	24/16
$L_1(d_{i+1}),U_0(d_i)$(2)	13,24	14,26	15,28	16,30
$m_1(i)$	6/16	6/16	8/16	8/16
$L_0(d_{i+1}),U_{-1}(d_i)$(2)	-26,-12	-28,-13	-30,-14	-32,-15
$m_0(i)$	-8/16	-8/16	-8/16	-8/16
$L_{-1}(d_{i+1}),U_{-2}(d_i)$(2)	-65,-48	-70,-52	-75,-56	-80,-60
$m_{-1}(i)$	-20/16	-20/16	-22/16	-24/16

2.2.3 商的转换

平方根商的转换和除法商的转换一样，可以采用商飞速转换，转换过程同样是设置寄存器 Q（商值）和 QM（商值减 1），其原理及 Q 和 QM 的更新与除法算法一样。

2.2.4 加法器输入 F 的产生

在平方根运算中，加法器输入 $F[j]=-(2S[j]s_{j+1}+s_{j+1}^2 r^{-(j+1)})$，从而新的部分余数 $w[j+1]=4*w[j]+F[j]$。在寄存器 Q 和 QM 的表示下，F[j]可转化为如下的表达式：

$$F[j]=\begin{cases}-2S[j]s_{j+1}-s_{j+1}^2 r^{-(j+1)}, s_{j+1}\geq 0\\ 2S[j]|s_{j+1}|-s_{j+1}^2 r^{-(j+1)}, s_{j+1}<0\end{cases}$$

$$=\begin{cases}-(2Q[j]+s_{j+1}r^{-(j+1)})s_{j+1}, s_{j+1}\geq 0\\ (2QM[j]+(2r-|s_{j+1}|)r^{-(j+1)})|s_{j+1}|, s_{j+1}<0\end{cases} \tag{23}$$

以字符串 aa…a 和 bb…b 分别表示 Q 和 QM，则 F 的更新如表 4 所示。

表 4 F 的更新

	F[j]		
s_{j+1}	以 S[j]形式	以 Q 和 QM 形式	字符串
0	0	0	0..00000
1	$-2S[j]-4^{-(j+1)}$	$-2Q[j]-4^{-(j+1)}$	a..aa001
2	$-4S[j]-4*4^{-(j+1)}$	$-4Q[j]-4*4^{-(j+1)}$	a..a0100
-1	$2S[j]-4^{-(j+1)}$	$2QM[j]+7*4^{-(j+1)}$	b..bb111
-2	$4S[j]-4*4^{-(j+1)}$	$4QM[j]+12*4^{-(j+1)}$	b..b1100

2.2.5 规格化

平方根的规格化和除法的规格化一样，初始时被开方数应规格化为[1/2,1)，因此尾数扩展为 56 位，即可满足结果的精度要求和正确性。

2.3 SRT 除法/平方根算法总结

由上述除法和平方根算法的分析可以知道，两者的计算过程有很多相似之处：商选择查找表可复用；由于尾数扩展精度一样，加法器可复用；商的转换都可采用商的飞速转换。因此可以将除法和平方根计算放在一个部件中，节省硬件开销。

3．总体结构设计

3.1 除法总体结构

两个浮点数据相除，被除数与除数的符号位进行异或运算得到结果的符号位；被除数的指数减去除数的指数得到中间指数结果；被除数与除数的尾数部分进行定点除运算，定点除法使用基 SRT4 算法。本文涉及的算法主要是针对尾数计算。

传统的 SRT 浮点除法总体结构如图 2 所示。

在传统 SRT4 除法迭代计算实现中每一次迭代涉及运算：

1、部分余数 P_j 经过移位器左移 2 位生成 $4*P_j$；

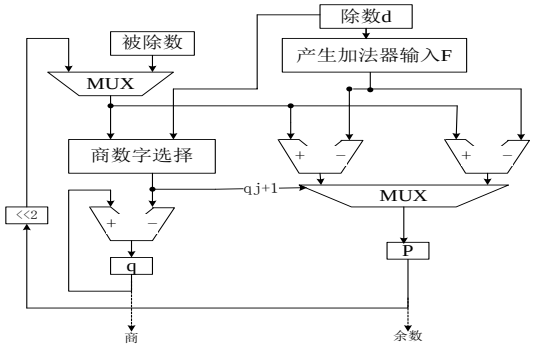

图 2 传统浮点除法总体结构

2、然后以 $4*P_j$ 和除数 d 为输入，经商选择逻辑查表得到该次循环的商 q_{j+1}；

3、通过 q_{j+1} 选择乘积值 $F=-q_{j+1}*d$；

4、最后通过加法器运算得到下一次循环的部分余数 P_{j+1}。

而本文设计的 SRT4 除法算法利用商的飞速转换将冗余结果转换为非冗余二进制形式，较少了加法器的使用，不仅减少了面积，还大大提高了速度，除法结构如图 3 所示。

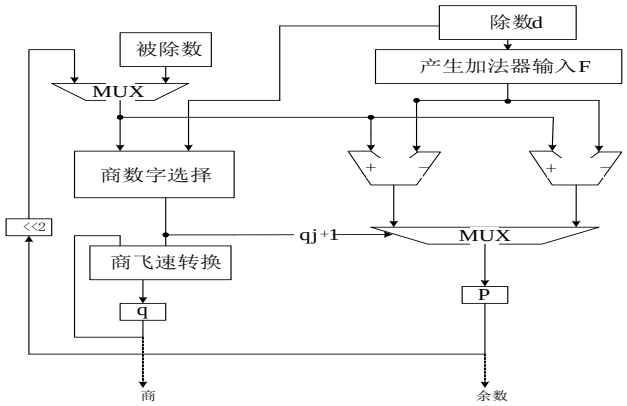

图 3 本文浮点除法总体结构

3.2 平方根总体结构

一个浮点数据求平方根，符号位不变；被开方数的指数除以 2 得到指数结果；被开方数的尾数部分使用 SRT4 算法进行运算。

在传统 SRT4 平方根迭代计算实现中每一次迭代涉及运算：

1、部分余数 W_j 经过移位器左移 2 位生成 $4*W_j$；

2、然后以 $4*W_j$ 作为输入，经商选择逻辑查表得该次循环的商 s_{j+1}；

3、通过 s_{j+1} 产生加法器输入 F；

4、最后通过加法器运算得到下一次循环的部分余数 W_{j+1}。

而本文设计的 SRT4 平方根算法利用商的飞速转换将冗余结果转换为非冗余二进制形式，同除法优化的原理一样，减少了加法器的使用，减少了面积，大大提高了速度。本文平方根结构如图 4 所示。

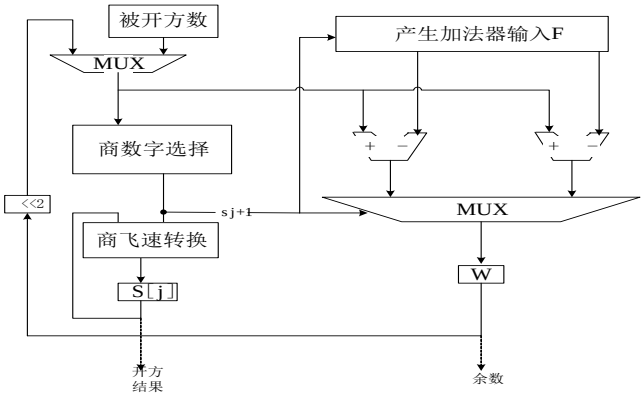

图 4 本文浮点平方根总体结构

3.3 除法/平方根总体结构

除法/平方根部件通过复用加法器和查找表来减少硬件开销，并通过 choose 模块来选择除法计算还是平方根计算。

本设计的 SRT 浮点除法/平方根总体结构如图 5 所示。

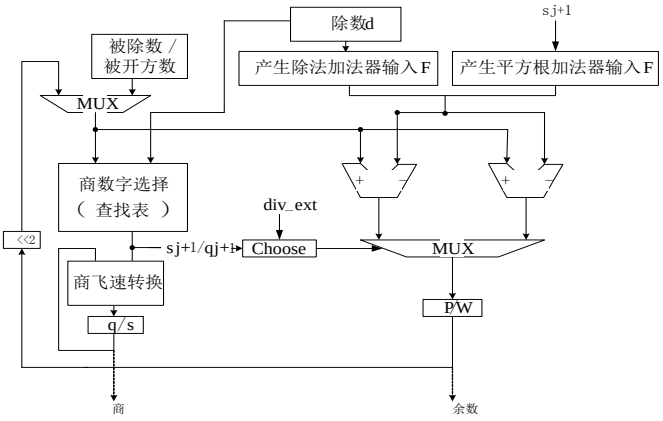

图 5 本文 SRT4 浮点除法/平方根总体结构

4. 实验结果

本文使用 Verilog 硬件描述语言进行代码设计，使用 Synopsys 公司的 DC（Design Complier）作为综合工具，采用 40nm 标准单元库，在“Typical”典型常温常压（1V，25℃）条件下对除法/平方根部件进行综合。

在时序约束时钟周期 400ps 的条件下，综合运行频率可以达到 2.5GHz，比较商冗余表示的除法和商飞速转换的除法，在相同实验环境下综合结果如表 5 所示。商飞速转换的除法部件面积是传统除法的 66.36%，相对减少 33.64%；功耗是传统除法的 72.25%，相对减少 27.75%；商飞速转换的时钟周期数是 31 拍，比传统方法的 32 拍减少了 1 拍。面积节省和性能提升是由于采用商飞速转换技术，有效降低存储商延迟，面积减少了一个 56 位全加器。

表 5 除法器综合结果比较

	面积(um^2)	功耗（mW）	周期数
商冗余除法	6451.0655	8.3313	32
商飞速转换	4280.6399	6.0195	31
性能提升	减少 33.64%	减少 27.75%	少 1 拍

由于商飞速转换除法的性能较优，因此本文同样采用商飞速转换的方法设计平方根算法，得到的单独除法部件、单独平方根部件以及除法/平方根部件的综合结果如表 6 所示。在时序约束时钟周期 400ps 的条件下，这三个部件的综合运行频率均可达 2.5GHZ，单独除法的面积为 4280.6399um^2，单独平方根的面积为 7087.5168um^2，除法/平方根部件的面积为 9812.3087um^2，是两个单独部件的面积和的 86.31%，相对减少 23.69%；单独除法的功耗为 6.0195mW，单独平方根的功耗为 6.0015mW，除法/平方根部件的功耗为 10.7854mW，是两个单独部件的功率和的 89.72%，相对减少 10.28%。

表 6 综合结果对比

	面积(um^2)	功耗（mW）	周期数
单独除法	4280.6399	6.0195	31
单独平方根	7087.5168	6.0015	28
除法/平方根	9812.3087	10.7854	
性能提升	减少 23.69%	减少 10.28%	

本文设计的除法/平方根部件每完成一次除法运算的时钟周期数为 31 拍，每完成一次平方根运算的时钟周期数为 28 拍，综合频率可达 2.5GHz。而 LSC87[3] 的芯片系统工作频率仅 8MHz，完成除法和平方根运算都需要 38 拍；r4comb[6]完成除法和平方根运算需 28 拍，但它的运行频率仅 100MHz。可以发现，本设计的综合运行频率远高于 LSC87 和 r4comb，时钟周期数相较 LSC87 也有所减少。

5. 结论

本文设计实现了一个浮点除法/平方根部件，首先在传统 SRT4 结构的基础上，提出商选择和余数加法并行处理，减少关键路径延时，同时运用商飞速转换技术，加快除法/平方根部件迭代运算速度。本文将 SRT4 浮点除法和平方根运算集成在一个部件中，复用查找表和加法器，减少了资源开销，节省了整体的面积。本文将传统除法与商飞速转换除法进行比较，将单独除法、单独平方根和除法/平方根合并运算进行比较，将 LSC87、r4comb 的除法/平方根设

计与本设计进行比较，综合结果表明，40nm 工艺库下，本设计的除法/平方根部件在面积、功耗、延时、频率等方面都有较大改进。

本文除法运算的计算延时 31 拍和平方根运算的计算延时 28 拍仍较长，如今一些高基 SRT 算法研究已取得较大成果，如 IntelPenryn 处理器采用 SRT16 算法将除法和平方根算法延时减少到 13 拍，但相应的硬件复杂度、面积、功耗等较大。下一步工作是用我们的方法设计一种低开销、低延时、高性能的高基 SRT 除法/平方根运算部件，以便在延时和性能之间寻求更好的折衷。

参考文献(References)

[1] H. Baliga, N. Cooray, E. Gamsaragan, P. Smith, K. Yoon, J. Abel, and A. Valles, “Improvements in the Intel Core2 Penryn Processor Family Architecture and Microarchitecture,” Intel Technology J., vol. 12, no. 3, pp. 179-192, http://www.intel.com/technology/

[2] N. Burgess and C.N. Hinds, “Design of the ARM Vfp11 Divide and Square Root Synthesisable Macrocell,” Proc. 18th IEEE Symp. Computer Arith- metic, pp. 87-96, July 2007.

[3] G. Gerwig, H. Wetter, E.M. Schwarz, and J. Haess, “High Performance Floating-Point Unit with 116 Bit Wide Divider,” Proc. 16th Symp. Computer Arithmetic, pp. 87-94, 2003.

[4] Liang Zhen and Shen Xu-bang, The Combinative Implemen-tation of Division and Square Root in An Embedded Coprocessor, Journal of Computer Research and Development, pp. 1016-1020,2001.

[5] Wei Liu and Alberto Nannarelli. Power Efficient Division Square Root Unit, IEEE Trans. Vol. 61, no. 8, pp. 1059-1070, August 2012.

[6] Yan Jiang-yu, Research and Circuit Design of Floating-point Division and Square Root, Master’s thesis of North China Electric Power University, 2004.

[7] M D Ercegovac, T Lang. Division and square root: Digit recurrence algorithms and implementations. Norwell, Mass: Kluwer Academic Publishers, 1994

[8] David Stevenson et *al.* An Atnerican national standard IEEE standard for binary floating-point arithmetic. ACM SIGPLAN Notices, 1987, 22(2): 9~25.

[9] Li Da-peng, Design and Implementation of High-performance Floating-point Division and Square Root, Master’s thesis of Northwestern Polytechnical University, 2006.

[10] Wang Wen-guang, Design and Implementation of Double Precision 64bits Floating-point Division Unit, Master’s thesis of Central South University, 2007.

[11] M D Ercegovac, T Lang. Radi-4 square root without initial PLA. IEEE Trans in Computers, 1990 39(8):1016-1024.