文章编号：2095-4980(2018)02-0342-06

一种基于约化因子上三角矩阵求逆的 FPGA 实现方法

周 杨[1,2]，王佳薇[1]，黄志洪[1]，杨海钢[*1,2]

(1.中国科学院 电子学研究所 可编程芯片与系统研究室，北京 100190；2.中国科学院大学 微电子学院，北京 100049)

摘 要：矩阵运算广泛应用于实时性要求的各类电路中，其中矩阵求逆运算最难以实现。基于现场可编程门阵列(FPGA)实现矩阵求逆能够充分发挥硬件的速度与并行性优势，加速求逆运算过程。基于改进的脉动阵列的计算架构，采用一种约化因子求逆的优化算法，将任意一个 $n\times n$ 阶上三角矩阵转换成对角线为 1 的上三角矩阵，使得除法运算与乘加运算分离开来，大大简化矩阵求逆运算过程。以一个 4×4 阶上三角矩阵求逆为例，在 Xilinx ISE 平台下，采用 Virtex5 FPGA 完成算法实现与功能验证，在 14 个周期内，使用了 2 个除法器，3 个乘法器与 4 个加法器实现整个矩阵求逆运算。相比于经典的脉动阵列架构，仅占用近一半资源的同时，性能提升了 26.43%；相比于集成更多处理单元(PE)的脉动阵列实现方式，在性能近乎不变的情况下，耗费的资源缩减到 1/4，大幅度提升了资源利用率。

关键词：矩阵求逆；现场可编程门阵列；约化因子

中图分类号：TN402 **文献标志码：**A **doi：**10.11805/TKYDA201802.0342

A method of implementing upper-matrix inversion based on the simplification factor in FPGA

ZHOU Yang[1,2]，WANG Jiawei[1]，HUANG Zhihong[1]，YANG Haigang[*1,2]

(1.Institute of Electronics，Chinese Academy of Sciences，Beijing 100190，China；
2.School of Microelectronics，University of Chinese Academy of Sciences，Beijing 100049，China)

Abstract：It's practically complicate to implement the matrix inversion which is widely used in all kinds of real-time circuit calculation. Taking Field Programmable Gate Arrays(FPGA) to implement the operation is able to take advantages of the hardware's speed and parallelism to accelerate the matrix inversion. Based on improved systolic architecture, a simplification factor algorithm is proposed, in which any $n\times n$ upper-matrix is transferred into an upper-matrix whose diagonal value is 1 to split dividing processor with multiplying and adding processor to greatly simplify the matrix inversion. Taking a 4×4 upper-matrix as an example, the algorithm is implemented and executed the functional verification adopting Virtex5 device in Xilinx ISE and utilizing 2 dividers, 3 multipliers and 4 adders in 14 cycles to accomplish all matrix inversion. Compared with the classical systolic architecture, resources are just occupied nearly half, while performance gets improved by 26.43%. Compared with systolic architecture integrated more Processing Elements(PE), the performance does not change and the resources reduce to 1/4, which greatly improves the resources utilization.

Keywords：matrix inversion；Field Programmable Gate Arrays；simplification factor

目前在信号处理、合成孔径雷达成像处理与通信方面，不仅要求高吞吐量，而且还要求满足实时性处理的条件。普通的多处理器运算由于资源分配、同步、存储冲突的中断问题，影响实时性的要求[1]。近年来，随着大规模集成电路，尤其是现场可编程门阵列(FPGA)的快速发展，运算电路有了长足的发展[2]。

收稿日期：2016-11-28；**修回日期：**2017-01-04

基金项目：国家自然科学基金资助项目(61704173,61474120)；北京市科技重大专项资助项目(Z171100000117019)；北京工商大学食品安全大数据技术北京市重点实验室开放课题基金资助项目(BKBD-2017KF05)

*通信作者：杨海钢 email: yanghg@mail.ie.ac.cn

矩阵运算是许多要求实时性运算电路的一个重要组成部分，其中矩阵求逆运算应用广泛，但却最难实现。矩阵求逆运算通常采用分解的方法，典型的有上–下(Lower–Upper，LU)三角矩阵分解和正交三角(QR)分解，LU 分解分成一个上三角矩阵与一个下三角矩阵，QR 分解分为一个下三角矩阵与一个对称正定矩阵。无论哪一种分解方法，矩阵求逆的最关键部分是上三角矩阵的求逆运算(下三角可以通过转置变成上三角矩阵)[3]。

为加快矩阵求逆的运算速度，可以采用硬件实现的方式。在这一方面，已有不少研究成果，典型的是[1,4–6]采用脉动阵列，它的乘法、加法与除法单元都集成在处理单元(PE)里，另外[7–8]采用的是心动阵列，它是脉动阵列的改进架构，集成了更多的 PE 单元，能更快提升速度。以上这些文献的共同点是直接对上三角矩阵进行处理，也就是乘法、加法与除法单元都在一起。本文基于脉动阵列，延续脉动阵列易于互连、并行与流水化的特点，采用一种约化因子求逆的算法[9]：首先将上三角矩阵变为对角线为 1 的上三角矩阵，使得除法运算与乘加运算分离开来，进而实现矩阵求逆的运算。

1　算法介绍

1.1 逆矩阵算法

矩阵求逆运算的关键是对三角矩阵的操作。假设 $\boldsymbol{T}$ 为对角线为 1 的二阶上三角矩阵 $\boldsymbol{T}=\begin{pmatrix}1 & x\\ 0 & 1\end{pmatrix}$，则逆矩阵为 $\boldsymbol{T}^{-1}=\begin{pmatrix}1 & -x\\ 0 & 1\end{pmatrix}$。对任意给定的对角线为 1 的 $n\times n$ 上三角矩阵 $\boldsymbol{A}_{n\times n}=\begin{pmatrix}1 & a_{12} & a_{13} & \cdots & a_{1n}\\ 0 & 1 & a_{23} & \cdots & a_{2n}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & 1 & a_{n-1n}\\ 0 & 0 & 0 & \cdots & 1\end{pmatrix}$ 进行初等行变换，$(\boldsymbol{A}|\boldsymbol{E})\rightarrow(\boldsymbol{E}|\boldsymbol{A}^{-1})$，即可得到逆矩阵。每个 2 行 2 列交集所构成的二阶子矩阵，称为操作矩阵，如 $\begin{pmatrix}a_{12} & a_{13}\\ 1 & a_{23}\end{pmatrix}$ 构成一个操作矩阵，每次进行行变换的效果体现在操作矩阵的右上角元素上，即令 $b_{13}=a_{13}-a_{12}a_{23}$，$b_{13}$ 称为约化因子。其通式为：

$$b_{mn}=a_{mn}-\sum_{k=m+1}^{n-1}(a_{mk}b_{kn}),\ n\geqslant 3 \tag{1}$$

a_{mn} 为矩阵 $\boldsymbol{A}_{n\times n}$ 所在行列的元素，a_{mk} 为每次行变换后的结果。经过这样的变换后，得到的矩阵为：

$$A_{n\times n}^{-1}=\begin{pmatrix}1 & -a_{12} & -b_{13} & \cdots & -b_{1n}\\ 0 & 1 & -a_{23} & \cdots & -b_{2n}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & 1 & -a_{n-1n}\\ 0 & 0 & 0 & \cdots & 1\end{pmatrix} \tag{2}$$

1.2 上三角矩阵单位化

假设 $\boldsymbol{A}$ 为 $n\times n$ 阶上三角矩阵，$\boldsymbol{A}$ 可逆，则其对角线元素非零，可通过下面变换变为对角线为 1 的上三角矩阵，$\boldsymbol{CA}=\boldsymbol{A}'$，$\boldsymbol{A}_{n\times n}=\begin{pmatrix}c_{11} & c_{12} & c_{13} & \cdots & c_{1n}\\ 0 & c_{22} & c_{23} & \cdots & c_{2n}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & c_{n-1n-1} & c_{n-1n}\\ 0 & 0 & 0 & \cdots & c_{n\times n}\end{pmatrix}$，$\boldsymbol{A}_{n\times n}{}'=\begin{pmatrix}1 & a_{12} & a_{13} & \cdots & a_{1n}\\ 0 & 1 & a_{23} & \cdots & a_{2n}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & 1 & a_{n-1n}\\ 0 & 0 & 0 & \cdots & 1\end{pmatrix}$。

通式为：

$$a_{mn}=\frac{c_{mn}}{c_{mm}},\ m\geqslant 1,\ n\geqslant 2 \tag{3}$$

$\boldsymbol{C}$ 为对角阵，$\left(c_{mm}{}^{-1}\right)_{n\times n}$，则 $\boldsymbol{A}$ 的逆矩阵为：

$$\boldsymbol{A}^{-1}=\boldsymbol{A}'^{-1}\boldsymbol{C} \tag{4}$$

2 硬件实现

为在 FPGA 上实现提出的基于约化因子的求逆运算算法，本章提出了系统实现的架构，并对其实现的时间复杂度进行分析。

2.1 上三角矩阵单位化

对于大阶数的矩阵，通常采用分块矩阵的方式分成小阶数的矩阵。以4×4阶矩阵 $\boldsymbol{A}$ 为例，深入探究其硬件实现。系统架构如图 1 所示。整个系统由上三角矩阵单位化模块、对角线为 1 的逆矩阵算法实现模块和上三角逆矩阵输出模块组成，内部包含 2 个除法处理单元(Dividing Processor，DP)、3 个乘法处理单元(Multiplying Processor，MP)、4 个加法处理单元(Adding Processor，AP)、1 个多路选择器(Multiplexer，MUX)和静态随机存储器(Static Random Access Memory，SRAM)存储模块构成。为充分利用 FPGA 的并行性，采用流水线方式进行实现。

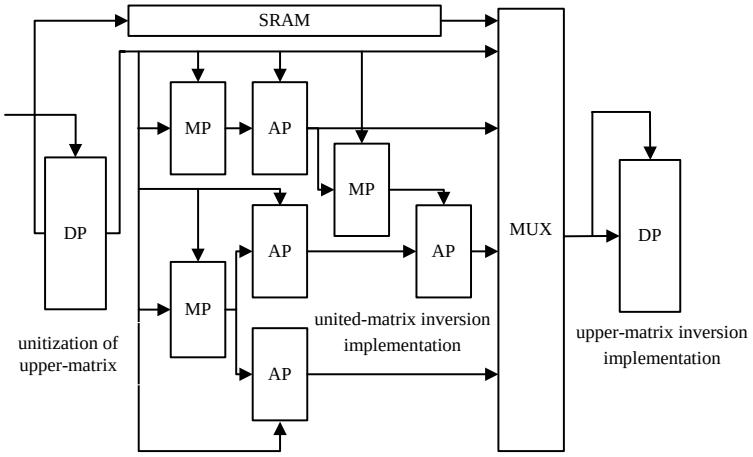

Fig.1 Architecture of 4×4 upper-matrix inversion
图 1 关于 4×4 阶矩阵求逆的系统架构

其中，上三角矩阵单位化模块将上三角矩阵元素进行单位化，同时按照数据流的顺序将上三角元素的值写入 SRAM 里。为遵循对角线为 1 的逆矩阵算法高效流水化的实现，按照图 2 所示的数据流顺序，进入除法处理单元并送入 SRAM 里进行数据存储。

$$\begin{pmatrix} ① & ④ & ⑧ & ⑨ \\ 0 & ② & ⑤ & ⑥ \\ 0 & 0 & ③ & ⑦ \\ 0 & 0 & 0 & ⑩ \end{pmatrix}$$

Fig.2 Data flow of 4×4 upper-matrix unitization
图 2 关于 4×4 阶矩阵单位化的数据流顺序

$$\begin{pmatrix} 1 & ① & ⑤ & ⑥ \\ 0 & 1 & ② & ③ \\ 0 & 0 & 1 & ④ \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig.3 Data flow of 4×4 united upper-matrix inversion
图 3 关于 4×4 阶矩阵单位化逆矩阵的实现数据流

对角线为 1 的逆矩阵算法实现模块对单位化的上三角矩阵实现求逆运算。它采用串行化流水方式，数据流顺序遵循的原则是：输入数据所在的行算法复杂度越高，其相关的输入优先度越高，乘法的数据优先于加法数据。如图 3 所示，第 1 行的复杂度最高，所以第 1 个输入来自 a_{12}；乘法优先的原则，故 a_{23} 紧随其后，a_{24} 与第 1 行第 1 个输入相关，其优先于 a_{34}，a_{13} 为加法数据，且为第 2 级输入。

该模块包含乘法处理单元与加法处理单元，图 4 为对角线是 1 的逆矩阵实现结构，它清晰显示了数据传输流，乘法顺序流和加法顺序流，有 4 个乘法运算，由于数据流的输入顺序，$a_{12}\times a_{24}$ 与 $a_{23}\times a_{34}$ 可以共享一个乘法器，结果通过寄存器进行存储，所以如图 1 所示，只用了 3 个乘法处理单元。最后，对所产生的结果进行取反，即可得到对角线为 1 的逆矩阵。

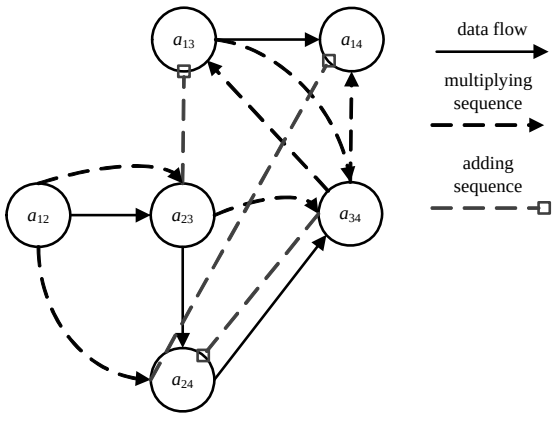

Fig.4 Structure of 4×4 united upper-matrix inversion implementation
图 4 有关 4×4 阶单位化逆矩阵的实现结构

上三角逆矩阵输出模块：前一个模块得到的是对角线为 1 的上三角矩阵的逆矩阵，为了得到上三角矩阵的逆矩阵，需要乘以对角阵 $\boldsymbol{C}$，也就是除以 $(a_{ii})_{4\times4}$，不需要在上三角矩阵单位化模块先进行除法处理再进行乘法处理，而是直接进行除法处理。除法处理的数据流一方面来自对角线为 1 的上三角矩阵的逆矩阵的结果，另一方面读取 SRAM 里对角阵 $\boldsymbol{C}$ 的值。

2.2 时间复杂度

对各运算单元的延迟分别进行标示，设定加法器延时为 a，乘法器延时为 m，除法器延时为 d。由于上三角矩阵单位化需要通过 3 个周期来完成对角线数据的输入，故延迟 T_0 为：$T_0=3+d$。

对角线为 1 的上三角矩阵求逆周期 T_1 为：

$$T_1=\begin{cases}6+2\mathrm{a}, & m=1\\ 2+2a+2m, & m\geqslant 2\end{cases} \tag{5}$$

上三角逆矩阵输出模块的周期 T_2 为：$T_2=1+d$，则整个求逆过程所需要的周期为：$T=T_0+T_1+T_2$，即

$$T=\begin{cases}10+2a+2\mathrm{d},\ m=1\\ 6+2a+2m+2d,\ m\geqslant 2\end{cases} \tag{6}$$

此时，如果当 $a=m=d=1$，T 为 14 个周期。

3　仿真结果及分析

为验证本文提出的优化矩阵求逆算法，在 Xilinx ISE 上采用 Virtex5 器件实现该算法，并在 Modelsim 工具里进行仿真，得到仿真后的结果，同时对其资源与性能进行评估分析。

3.1 仿真结果

给定任意一个4×4阶的上三角矩阵 $\begin{pmatrix}-2&1&-9&-18\\0&2&2&9\\0&0&-2&6\\0&0&0&2\end{pmatrix}$，将其输入上三角矩阵单位化模块，通过 Modelsim 进行仿真，可得如图 5 所示的结果。

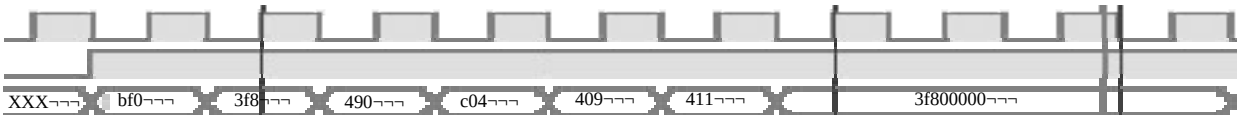

Fig.5 Simulation results of 4×4 upper-matrix unitization module
图 5 关于 4×4 阶上三角矩阵单位化模块的仿真结果

单位化后数据的定点与浮点分别表示为：

$$\begin{pmatrix}1&-0.5&4.5&9\\0&1&1&4.5\\0&0&1&-3\\0&0&0&1\end{pmatrix}\begin{pmatrix}3\mathrm{f}800000&\mathrm{bf}000000&40900000&41100000\\0&3\mathrm{f}800000&3\mathrm{f}800000&40900000\\0&0&3\mathrm{f}800000&\mathrm{c}0400000\\0&0&0&3\mathrm{f}800000\end{pmatrix}$$

将上述单位化后的 4×4 阶矩阵输入对角线为 1 的上三角矩阵求逆模块进行实现，通过 Modelsim 工具进行仿真，可得如图 6 所示的结果。

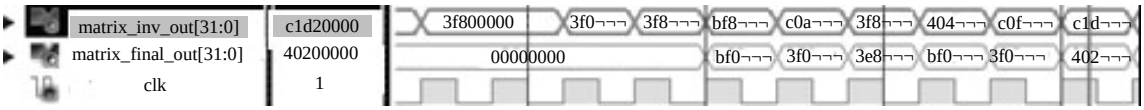

Fig.6 Simulation results of 4×4 united upper-matrix inversion module
图 6 关于 4×4 阶单位化上三角矩阵求逆模块的仿真结果

对角线为 1 的上三角矩阵求逆后，定点与浮点分别表示为：

$$\begin{pmatrix}1&0.5&-5&-26.25\\0&1&-1&-7.5\\0&0&1&3\\0&0&0&1\end{pmatrix}\begin{pmatrix}3\mathrm{f}800000&3\mathrm{f}000000&\mathrm{c}0\mathrm{a}00000&\mathrm{c}1\mathrm{d}20000\\0&3\mathrm{f}800000&\mathrm{bf}800000&\mathrm{c}0\mathrm{f}00000\\0&0&3\mathrm{f}800000&40400000\\0&0&0&3\mathrm{f}800000\end{pmatrix}$$

将上述数据结果输入上三角矩阵的逆矩阵模块，通过 Modelsim 工具进行仿真，结果如图 7 所示。

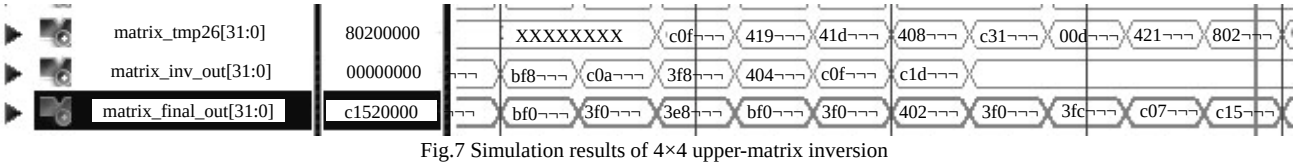

Fig.7 Simulation results of 4×4 upper-matrix inversion
图 7 关于 4×4 阶上三角矩阵求逆的仿真结果

即可得到数据处理后的最终结果，其定点与浮点分别表示为：

$$\begin{pmatrix} -0.5 & 0.25 & 2.5 & -13.125 \\ 0 & 0.5 & 0.5 & -3.75 \\ 0 & 0 & -0.5 & 1.5 \\ 0 & 0 & 0 & 0.5 \end{pmatrix} \quad \begin{pmatrix} \text{bf000000} & \text{3e800000} & \text{40200000} & \text{c1520000} \\ 0 & \text{3f000000} & \text{3f000000} & \text{c0700000} \\ 0 & 0 & \text{bf000000} & \text{3fc00000} \\ 0 & 0 & 0 & \text{3f000000} \end{pmatrix}$$

3.2 分析

在 Xilinx ISE 平台上采用 Virtex5 XC5VFX100T 器件实现了整个上三角矩阵求逆运算，通过资源分析，共需 2 个除法器，3 个乘法器和 4 个加法器，当运算单元延迟为 1 时，所需周期数为 14。与现有的研究工作进行比较，如表 1 所示。

表 1 与现有研究关于周期数与运算单元数的比较

Table1 Compared with existing research about cycles and numbers of computation

references	architecture	cycles	number of computing units		
			adder	multiplexer	divider
[1,4–6]	classic systolic	19	6	6	4
[7–8]	systolic with more PEs	13	16	16	16
this paper	lite-systolic	14	4	3	2

与基于脉动阵列的相关研究相比较，无论是运算单元数目还是所需周期数，本文提出的精简脉动阵列实现方式都明显占优。与心动阵列相关研究工作相比，本文实现方式在仅牺牲一个周期的代价下，运算单元数目减少到原占用资源的 1/4，大幅度提升了资源利用率。

基于约化因子的求逆算法不仅适用于 4×4 阶矩阵求逆，也适用于任意阶矩阵求逆。对于任意阶矩阵求逆，通常有 2 种方法：一种是直接对整个矩阵按照约化因子算法实现；另一种是将任意阶矩阵分块，得到多个低阶数的分块矩阵，再按照分块矩阵的求逆运算，实现对整个矩阵的求逆。第 1 种方法对任意高阶矩阵的实现比较复杂，第 2 种方法能够将任意高阶矩阵的求逆转化为低阶矩阵的求逆。所以，任意高阶数矩阵求逆可以采用分块矩阵的方法转化为低阶矩阵的求逆，而任意低阶矩阵的求逆可以用本文提出的基于约化因子算法直接实现。

4 结论

基于脉动阵列，在 FPGA 上采用一种约化因子求逆的算法，将除法运算与乘加运算分离开来，实现了任意一个 4×4 阶上三角矩阵变为对角线为 1 的上三角矩阵，通过在商用 FPGA Virtex5 上进行实现，验证了功能正确性。通过该算法，在 14 个周期内，使用了 2 个除法器，3 个乘法器和 4 个加法器即实现了整个 4×4 阶上三角矩阵的求逆运算。相比脉动阵列实现方式，在占用更少资源的情况下，可将实现周期减小 26.32%。相比心动阵列实现方式，在仅牺牲一个周期数的情况下，使占用的资源数缩减到 1/4，大幅度提升了资源利用率，提高了芯片的矩阵求逆运算的实现能力。

参考文献：

[1] 孙泉,赵明,张秀君. 基于脉动阵列的 LU 算法矩阵求逆 VLSI 结构[J]. 微电子学与计算机, 2007,24(3):138–141. (SUN Quan,ZHAO Ming,ZHANG Xiujun. Implementation matrix inversion of LU algorithm with systolic array[J]. Microelectronics & Computer, 2007,24(3):138–141.)

[2] 郭春煊,毛志刚,谢憬. 矩阵求逆运算的 VLSI 实现[J]. 计算机技术与发展, 2008,18(5):219–223. (GUO Chunxuan,MAO Zhigang,XIE Jing. VLSI realization of matrix inversion[J]. Computer Technology and Development, 2008,18(5):219–223.)

[3] 颜庆津. 数值分析[M]. 北京:北京航空航天大学出版社, 2006:14–65. (YAN Qingjin. Numerical analysis[M]. Beijing: Beijing University of Aeronautics and Astronautics Press, 2006:14–65.)

[4] ELAMAWY A. A systolic architecture for fast dense matrix inversion[J]. IEEE Transactions on Computers, 1989,38(3):449–455.

[5] LI G J,WAH B W. The design of optimal systolic arrays[J]. IEEE Transactions on Computers, 1985,34(1):66–77.

[6] ELAMAWY A,DHARMARAJAN K. Parallel VLSI algorithm for stable inversion of dense matrices[J]. IEE Proceedings E Computers and Digital Techniques, 1989,136(6):575–580.

[7] 王前,吴淑泉,李韬,等. 三角矩阵求逆的 ASIC 实现研究[J]. 微电子学与计算机, 2004,21(8):135–136,140. (WANG Qian,WU Shuquan,LI Tao,et al. Study on inverse matrix calculation and its implementation in ASIC[J]. Microelectronics & Computer, 2004,21(8):135–136,140.)

[8] 宋一鑫,姚振东. RMMSE 雷达脉冲压缩快速算法中矩阵求逆的 FPGA 实现[J]. 成都信息工程学院学报, 2009,24(5): 435–439. (SONG Yixin,YAO Zhendong. FPGA implementation of matrix inversion in RMMSE radar pulse compression express algorithm[J]. Journal of Chengdu University of Information Technology, 2009,24(5):435–439.)

(下转第 362 页)