基于 FPGA 的快速原地转置算法

李汉清
(中国科学院长春光学精密机械与物理研究所 长春 130033)

摘 要：随着合成孔径雷达成像系统对实时性要求的提高，基于 FPGA 平台的 SAR 成像系统越来越多。图像矩阵转置作为 SAR 成像系统的必要处理过程之一，其效率一直制约着 SAR 成像系统的实时性。本文提出了一种基于 FPGA 的快速原地转置算法，首先将矩阵分割为大小相等的子矩阵块，利用上位机预先计算出子矩阵块转置前后地址并生成地址序列表，在 FPGA 上依据地址序列表顺次移动子矩阵块。实验结果表明，相比非原地转置算法，这种原地转置算法使用的辅助空间仅为矩阵空间的 0.2%，多耗费 2%的时间。这种算法提高了转置效率，降低了转置存储开销，对 SAR 实时成像系统的有重要意义。

关键词：分块；原地转置；地址序列表
中图分类号：TP312 **文献标识码**：A **国家标准学科分类代码**：520.4010

Fast in-place matrix transposition algorithm based on FPGA

Li Hanqing
(Changchun Optical Precision Machinery and Physics Institute, Changchun 130033, China)

Abstract: With the development of synthetic aperture radar real-time imaging system, more and more SAR imaging systems are based on FPGA. The transposition of image matrix data is one of the essential procedure of SAR imaging system, and its efficiency has been restricted the SAR real-time imaging system. This paper proposes a fast in-place matrix transposition algorithm based on FPGA. Firstly, matrix is segmented into sub-matrices of the same size. Secondly, an address sequence table which indicates sub-matrix address in pre and post transposition is calculated by computer in advance. Lastly, according to the address sequence table, sub-matrices are transposed on FPGA. The result of analysis and simulation shows that this algorithm occupies an extra 2% time span and 0.2% matrix space. This algorithm improves the efficiency of matrix transposition, reduces the burden of storage, which is important to the SAR real-time imaging system.

Keywords: matrix partition; in-place matrix transposition; address sequence table

1 引 言

数据矩阵转置是多维图像处理和信号处理过程中的基本运算之一，特别在合成孔径雷达实时(synthetic aperture radar)成像系统中，矩阵转置是必不可少的[1-2]。合成孔径雷达是现代雷达领域发展的重要方向，有着广泛的应用前景和发展潜力，一直受到世界各国的高度重视。合成孔径雷达成像系统中的两个主要难题是高实时性和数据量大[3]，在矩阵转置时既要保证系统实时性又要降低存储负担。在 SAR 成像处理中大量的数据参与相关运算，从而导致运算量巨大，要实现系统实时处理就要求高速的数据吞吐率和系统运算速度。随着 FPGA 技术在数字信号处理领域的逐渐成熟，越来越多的 SAR 成像系统基于 FPGA 平台设计[4]。在矩阵规模较小时，利用 FPGA 内部存储完成转置相对容易；在矩阵规模较大时，矩阵数据通常存储于 SDRAM 等外部高速存储器中，此时转置算法的性能直接影响系统实时性[5]。

目前，针对大规模矩阵转置常用的方法有两页式或三页式转置法。这两种方法都需要转置存储器配合，两页式将转置存储器分为大小相等的两块，三页式将转置存储器分为三块。两页式结构与三页式结构各有优劣，两页式节省存储，三页式控制逻辑简单[6-7]。但是这两种方法都需要额外的转置存储器，增加了系统硬件设计负担[8-9]。文献[10]提出的补齐式准原地转置算法能够进一步提高转置效率，但是需要仍需要较大的辅助存储空间，是一种准原地转置算法。

收稿日期：2015-09

2　算法原理

对于高阶矩阵的处理运算,可以将其划分为若干块低阶矩阵,划分后的低阶矩阵为子矩阵。

$$\boldsymbol{A}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & & \cdots \\ a_{ml} & a_{ml} & \cdots & a_{mn} \end{pmatrix}= \begin{array}{c} \begin{matrix} n_1 & n_2 & \cdots & n_s \end{matrix} \\ \begin{matrix} m_1 \\ m_2 \\ \vdots \\ m_r \end{matrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \cdots & \boldsymbol{A}_{1s} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \cdots & \boldsymbol{A}_{2s} \\ \cdots & \cdots & & \cdots \\ \boldsymbol{A}_{r1} & \boldsymbol{A}_{r2} & \cdots & \boldsymbol{A}_{rs} \end{pmatrix} \end{array} \tag{1}$$

式中:每个 $\boldsymbol{A}_{ij}$ 是 $m_i \times n_j$ 的子矩阵,且 $i=1,\cdots,r; j=1,\cdots,s; \sum_{i=1}^{r} m_i = m; \sum_{j=1}^{s} n_i = n$。则矩阵 $\boldsymbol{A}$ 的转置 $\boldsymbol{A}^{\mathrm{T}}$ 表示为:

$$\boldsymbol{A}^{\mathrm{T}}= \begin{array}{c} \begin{matrix} m_1 & m_2 & \cdots & m_r \end{matrix} \\ \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_s \end{matrix} \begin{pmatrix} \boldsymbol{A}_{11}{}^{T} & \boldsymbol{A}_{21}{}^{T} & \cdots & \boldsymbol{A}_{r1}{}^{T} \\ \boldsymbol{A}_{12}{}^{T} & \boldsymbol{A}_{22}{}^{T} & \cdots & \boldsymbol{A}_{r2}{}^{T} \\ \cdots & \cdots & & \cdots \\ \boldsymbol{A}_{1s}{}^{T} & \boldsymbol{A}_{2s}{}^{T} & \cdots & \boldsymbol{A}_{rs}{}^{T} \end{pmatrix} \end{array} \tag{2}$$

矩阵 $\boldsymbol{A}$ 转置时,可以看作是每个 $\boldsymbol{A}_{ij}$ 子矩阵内部转置后,各个 $\boldsymbol{A}_{ij}$ 子矩阵间也进行转置。

矩阵数据在硬件中一般是按行顺序存储结构,按照数据存储地址进行读写,数据存储地址与矩阵元素行列位置一一对应[8]。设矩阵 $\boldsymbol{A}$ 自存储地址 1 开始按行顺序存储,每个元素 a_{ij} 占用一个存储位置,在已知矩阵 $\boldsymbol{A}$ 的行列数(m,n)情况下,矩阵中任意元素 a_{ij} 存储地址 $addr$ 和元素行列位置(i,j)转换公式如下:

$$addr(i,j) = (i-1) \cdot n + j \tag{3}$$

$$\begin{cases} i(addr) = floor((addr-1)/n)+1 \\ j(addr) = mod((addr-1),n)+1 \end{cases} \tag{4}$$

公式(3)根据元素行列位置(i,j)计算存储地址 $addr$;公式(4)根据元素存储地址 $addr$ 计算行列位置(i,j)。其中,$floor(x)$ 为向下取整函数,计算结果为小于参数 x 的最大整数;$mod(x,y)$ 为求余函数,计算结果为参数 x 对参数 y 取模数值。

在逻辑结构上,矩阵转置是互换所有元素的行列位置。矩阵 $\boldsymbol{A}$ 中每个元素 a_{ij} 经过转置后的位置可以由元素行列位置(i,j)和矩阵行列数(m,n)计算得出。对于任意元素 a_{ij} 经过转置后的存储地址 $addr'$ 为:

$$addr'(i,j) = (j-1) \cdot m + i \tag{5}$$

结合公式(4),可获得矩阵 $\boldsymbol{A}$ 中任意元素 a_{ij} 转置前后存储地址函数关系:

$$addr'(addr) = mod((addr-1),n) \cdot m + floor((addr-1)/n)+1 \tag{6}$$

根据公式(6)在已知矩阵行列数(m,n)情况下,可以由存储地址值直接计算转置后地址,不需要考虑矩阵行列位置(i,j)。对于矩阵非原地转置,将所有元素由地址 $addr(a_{ij})$ 读出数据后存入另一存储区的 $addr'(a_{ij})$ 地址即完成矩阵 $\boldsymbol{A}$ 的转置。矩阵原地转置情况下,在每次向存储区写入数据前应确保该区域数据已经读出或者已经存入其他区域,即保证数据不会丢失。

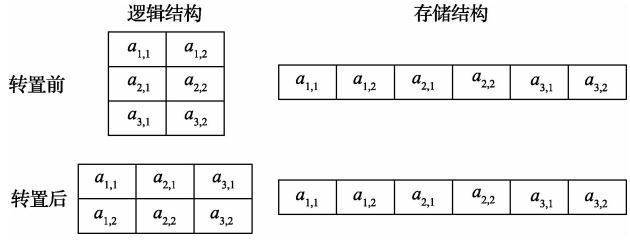

图 1　3×2 矩阵转置前后示意图

以 3×2 矩阵原地转置过程为例进行说明,如图 1 所示为 3×2 矩阵转置前后的逻辑结构和存储结构示意图。逻辑结构上 3×2 矩阵转置为 2×3 矩阵,交换对应元素位置;在按行顺序的存储结构上,矩阵转置交换元素位置时涉及多个元素位置调整。对照矩阵转置前后存储结构可以发现,存储地址 1、6 中数据不需要改变,地址 2 中数据存储到 4,地址 3 中数据存储到 2,地址 5 中数据存储到 3,地址 4 中数据存储到 5。

如图 2 所示为应用本文提出算法的 3×2 矩阵转置过程示意图。具体转置过程为:根据公式(6),对于地址 1 中数据转置后存储地址 $addr'=1$,无需改变地址 1 中数据;地址 2 中数据转置后存储地址为 $addr'=4$,为避免数据丢失,必须先读出地址 4 中数据存储到临时存储区,向地址 4 写入地址 2 中数据;同样方法移动地址 5 和地址 3 中数据;对于地址 3,其转置后地址为地址 2,地址 2 中数据已经经过转置,所以不需读出地址 2 中数据,写入地址 3 中数据到地址 2;最后对于地址 6,经过计算,无需改变地址 6 中数据。值得注意的是,在每次写入数据前需检查该写入地址中数据是否已经被移动,若移动过则不需要继续移动,否则需要计算该地址转置后地址,依次移动。因此,需要为每个存储地址设置标志,组成矩阵转置标志表,标示对应地址中元素是否已经移动过。

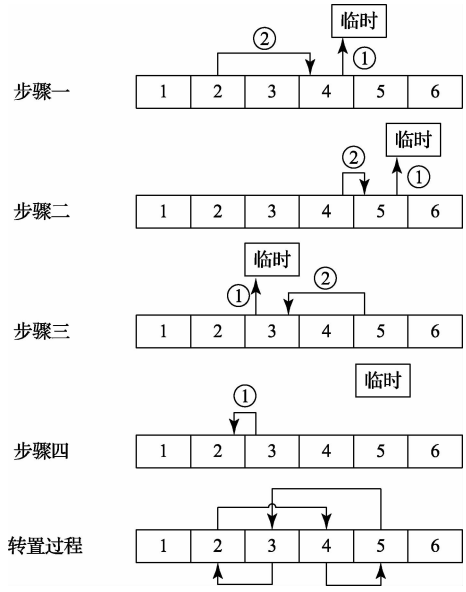

图 2　3×2 矩阵转置过程示意图

本文提出的算法流程如图 3 所示。1) 将矩阵存储位置首地址赋值给 *addr*；2) 判断矩阵转置是否结束，即判断地址 *addr* 是否已经超出矩阵最后一个元素地址，若超出则转置结束，否则进入步骤 3)；3) 根据公式(6) 计算出与 *addr* 对应的转置后地址 *addr'*；4) 检测 *addr'* 是否需要转置，不需要转置的情况有两种，第一种是源地址 *addr* 和转置后地址 *addr'* 相同，另一种是 *addr'* 地址中存储的元素已经转置过，后一种情况的判断需要依靠矩阵转置标志表，若检测不需要转置，则处理下一个存储地址；5) 若需要转置，则读出 *addr'* 中数据到临时存储区，写入 *addr* 的数据，至此完成了 *addr* 中存储的矩阵元素的转置，下一步需要完成 *addr'* 中数据元素的转置；6) 为完成 *addr'* 中元素转置，将 *addr'* 赋值给 *addr*，进入步骤 3)。

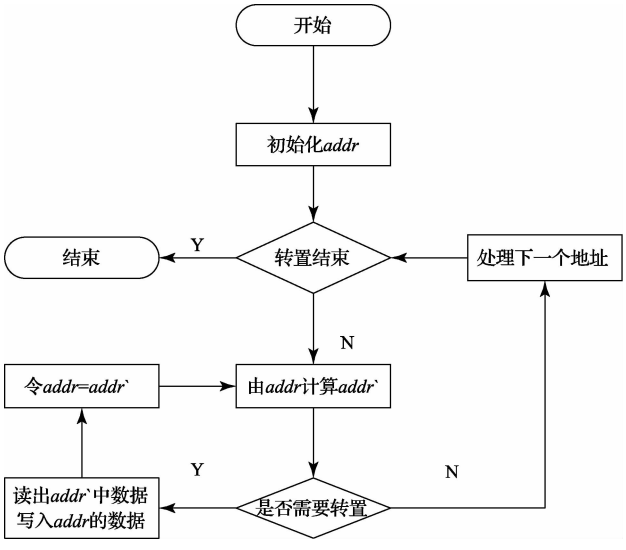

图 3　算法流程图

采用 *addr'* 查找表方式可以大幅度降低硬件处理复杂度，提升处理速度。但是在转置过程中存在查询转置标志表命中率大幅降低情况，尤其在后半段转置过程中，多数子矩阵块已经完成转置，导致查询命中率降低，*addr'* 查找速度下降。为进一步提升转置速度，提高查询命中率，使用地址序列表替换 *addr'* 查找表。在使用 *addr'* 查找表过程中，存在子矩阵已经转置完成而选空情况，剔除选空情况形成新的地址序列存储在地址序列表中。同时，为标识一次转置移动过程结束，设置末尾标志位。

3　硬件实现

在以 FPGA 为控制核心的处理系统中，由于片上存储容量有限，为满足高速数据处理，通常使用挂载片外 SDRAM 存储器方法提高处理性能[11-12]。SDRAM(同步动态随机存储内存)是一种具有同步接口的动态随机存储器。在访问 SDRAM 数据时，支持突发读写模式，突发读写长度有 2、4、8 几种模式，当突发长度为 8 时读写速率最快[13]。将整个矩阵划分为若干个 8×8×32 bit 的子矩阵，矩阵的转置即是完成所有子矩阵间的位置交换，和子矩阵自身的转置。

本文以矩阵元素位宽 32 bit 存储在数据位宽 32 bit 的 SDRAM 情况说明算法实现。如图 4 所示，为本文算法硬件实现示意图。算法核心为两个状态机：地址转换状态机和读写 SDRAM 状态机；片上存储空间包括：地址序列表和乒乓 RAM 缓冲区。由于地址序列计算过程中涉及乘除法运算，若采用 FPGA 计算需要占用较多处理时间。在上位机预先计算地址序列表，将计算结果初始化在 FPGA 片上 RAM 中，顺次检索地址序列表获得转置地址。乒乓 RAM 缓冲区大小设置为 2×8×8×32bit，即相当于两个子矩阵大小。乒乓 RAM 缓冲交替保存当前地址上的子矩阵元素和转置地址上子矩阵元素。

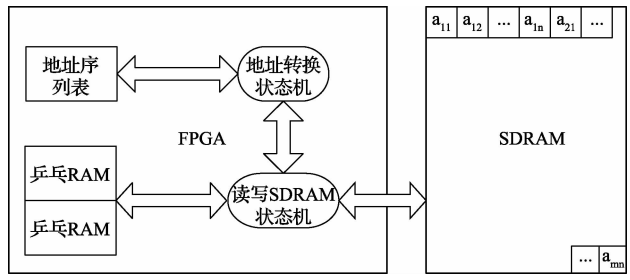

图 4　本文算法硬件实现示意图

地址转换状态机包括 5 个状态，如图 5 所示。在 S0 状态时顺次产生读取地址序列表的索引，若该索引没有超过最大索引，则进入 S1 状态，否则停止在当前状态。S1 状态获取转置地址，若该地址对应的末尾标志有效则进入 S4 状态，否则进入 S2 状态。在 S1 状态中根据上次末尾标志与当前末尾标志分为四种情况，若上次值和当前值都有效，当

前地址子矩阵原地转置;若上次值有效,当前值无效,记录当前地址,获取下一地址;若上次值无效,当前值有效,当前地址子矩阵移动到之前记录地址;若上次值和当前值无效,获取下一地址。在 S2 状态产生地址序列表的下一索引,进入 S3 状态。S4 状态等待读写 SDRAM 状态机接受移动前后地址,若地址接受确认信号有效,则将移动源地址和目的地址输出,进入 S0 状态,否则停在当前状态等待。

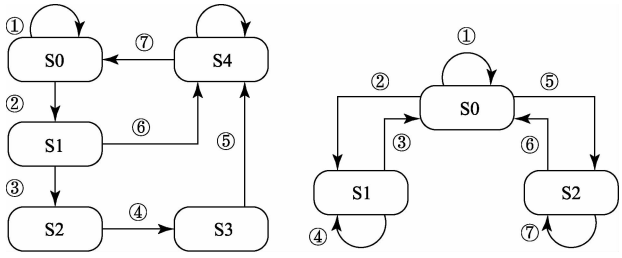

图 5　地址转换状态机和读写 SDRAM 状态机

读写 SDRAM 状态机包括 3 个状态。在 S0 状态时,等待地址转换状态机转换完成信号有效并且乒乓 RAM 缓冲区不为满,则记录移动源地址和目的地址,进入 S1 状态读取源地址数据和目的地址数据;否则若乒乓 RAM 缓冲区不为空时,进入 S2 状态。在 S1 状态时,根据地址转换状态机产生的地址(根据分块方式变化,以 8×8 分块为例),分别产生 8 次 SDRAM 突发读地址,每次突发读出 8 个数据,8 次突发读共 64 个数据组成一个子矩阵块,当一个子矩阵数据完成读入到乒乓 RAM 缓存区后,进入 S0 状态。在 S2 状态时,将乒乓 RAM 缓存区中数据写入 SDRAM 新地址,在每次突发写数据时需要按照转置后数据格式写入,如第一次突发写数据依次子矩阵的同一列元素,当一个子矩阵数据都写入到 SDRAM 中后,进入 S0 状态。

4　实　　验

4.1　空间效率

本文算法在转置过程中使用的辅助数据存储空间包括:乒乓 RAM 和地址序列表。矩阵大小和子矩阵划分方式是影响辅助数据存储空间的主要因素。

$$MEM_{aux}=2\times t^2\times D+r\times s\times ceiling(\log_2(r\times s)+1) \tag{7}$$

对于行列数为(m,n)的矩阵 $\boldsymbol{A}$,将其划分为行列数为(r,s)的分块矩阵,每个子矩阵大小都为 $t\times t$。设每个矩阵元素占用存储空间为 D *bit*,则本文算法使用的乒乓 *RAM* 大小为 $2\times t\times t\times D$ *bit*;地址序列表大小为 $r\times s\times ceiling(\log_2(r\times s)+1)$ *bit*,其中 *ceiling* 表示向上取整函数。则辅助数据占用的存储空间可根据式(7)计算,对于行列数已知的矩阵,可求出辅助数据占用存储空间。

当矩阵 $\boldsymbol{A}$ 为 1024×1024 矩阵,每个元素占用 32 bit 存储时,矩阵分块方式与辅助数据存储空间大小关系如图 6 所示。由图可知,当分块方式为 16×16 时,辅助空间最小,占用 64 Kbit,相比矩阵 $\boldsymbol{A}$ 占用 32 Mbit,辅助数据占用存储空间为矩阵空间 2.08‰。

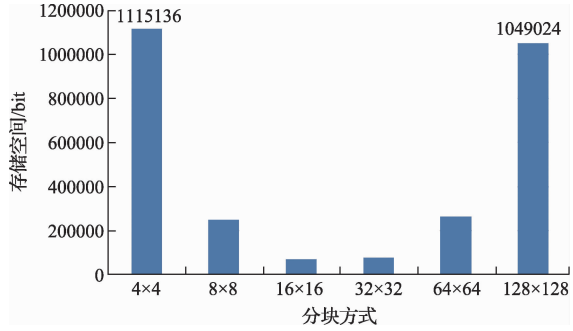

图 6　分块方式与辅助数据存储空间关系图

在公式(7)中,矩阵的行数和列数变化对辅助数据存储空间影响相同。当矩阵 $\boldsymbol{A}$ 行数固定为 1024,分块方式为 16×16,每个元素占用 16 bit 存储时,辅助数据存储空间与矩阵数据存储空间比值随列数变化关系如图 7 所示。由图可知,在矩阵列数不超过 512 列时,随着矩阵列数增加,存储空间比值急剧下降,乒乓 RAM 占辅助数据空间比例由 97.34%下降至 40.00%;当矩阵列数大于 512 后,存储空间比值变化缓慢,但在列数大于 4096 后,存储空间比值开始上升,地址序列表占用空间成为辅助数据的主要部分。

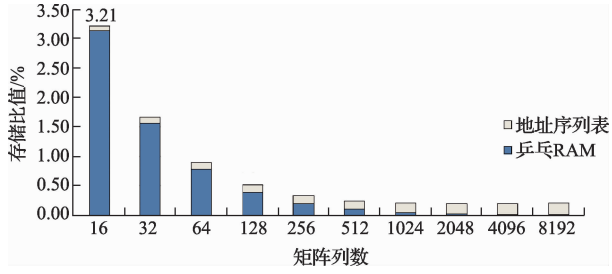

图 7　存储空间比值与矩阵列数关系图

在空间效率上,本文提出的原地转置方法相比非原地转置方法能够大幅减少存储使用,针对不同行列数的矩阵,根据公式(7)选择最优分块方式能够进一步提高空间效率。

4.2　时间效率分析

SDRAM 存储器在突发读写长度为 8 时,读写速率可达到最快。因此,为缩短转置过程耗时,取得最高的转置速度,以突发长度为 8 的读写作为最小数据单元。在转置过程中分块方式最小为 8×8,并且分块方式以 8 为最小递增单元,即分块方式可以为 8×8、16×16、24×24 等(但分块方式必须使矩阵能划分为整数个分块)[14-15]。

本文算法包括两个状态机,地址转换状态机和读写 SDRAM 状态机。两个状态机同步并行工作,它们以一个子矩阵块处理过程为周期运行,周期长短取决于最慢状态机耗时。读写 SDRAM 状态机在一个处理周期内包括一次子矩阵块的读写 SDRAM 操作和转置操作。本文算法与非

原地转置方式相同，都需要经过读 SDRAM 中的子矩阵块、子矩阵块转置和子矩阵块写入 SDRAM 三个步骤，两种方法耗时相同。以 8×8 分块方式为例，不考虑 SDRAM 刷新和潜伏周期等因素，最短需要经过 128 个周期可以完成子矩阵一次读写操作。而在最差情况下，地址转换状态机经过 10 个周期可以完成地址转换工作。因此，本文算法的时间效率取决于读写 SDRAM 效率，与非原地转置方法时间效率相同。

4.3　仿真实验

仿真实验环境为 ModelSim10.1a，以 1024×1024 矩阵，单个元素占 32 bit，分块方式为 8×8 为例，SDRAM 无刷新，一次突发读写占用 12 个周期。地址转换状态机仿真结果如图 8 所示，地址转换状态机(st_addr_trans)在最差情况下经过 10 个周期完成地址转换，产生子矩阵移动前后地址(id_rd 和 id_wr)。与此同时，读写 SDRAM 状态机处于读 SDRAM 状态，产生读 SDRAM 命令(SDRAM_RDEN)并顺次采集 SDRAM 数据(SDRAM _DATA_IN)。仿真结果显示算法完成矩阵转置耗费 3 211 268 个周期，理论上非原地转置算法最少需耗费 3 145 728 个周期，仿真结果比理论值多用 2%时间。

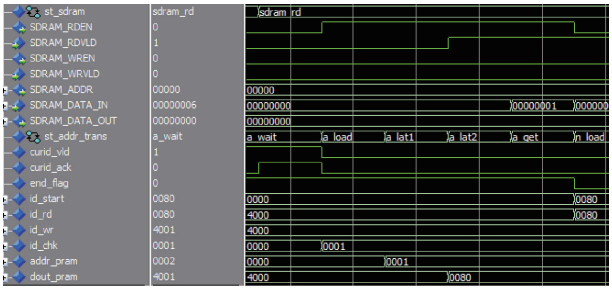

图 8　仿真结果图

5　结　　论

本文提出了一种基于 FPGA 的快速原地转置算法，根据 SDRAM 突发模式特点和待转置矩阵规模划分子矩阵块，在 FPGA 上依据预先计算的地址序列表顺次移动转置子矩阵块。分析仿真结果表明，空间效率上，使用的辅助空间为矩阵空间的 0.2% ；时间效率上，本文原地转置算法比非原地转置算法多好用 2%时间。本文提出的算法空间效率与时间效率都能够满足合成孔径雷达成像系统的实时性要求。

参考文献

[1]　邓磊，桂晓雷，吴兆阳. 基于 FPGA 的整数三维 DCT 变换的实现[J]. 电子测量技术，2013，36(12)：68-70.

[2]　殷亚男，王晓东，李丙玉. 基于预测和 JPEG2000 的 MODIS 红外辐射多光谱图像无损压缩算法[J]. 液晶与显示，2013，28(6)：922-926.

[3]　王林泉. SAR 实时成像系统中的压缩和矩阵转置研究[D]. 西安：西安电子科技大学，2015.

[4]　王一鸣. 基于 FPGA 的 SAR 转置存储器设计[D]. 西安：西安电子科技大学，2012.

[5]　王丽，魏巍，吴林钢，等. SAR 图像目标识别新方法[J]. 液晶与显示，2014，29(3)：429-434.

[6]　卢世祥，韩松，王岩飞. 合成孔径雷达实时成像转置存储器的两页式结构与实现[J]. 电子与信息学报，2005，27(8)：1226-1228.

[7]　李早社，禹卫东，汪亮，等. 基于 SDRAM 的星载 SAR 星上实时成像转置存储器[J]. 信号处理，2007，23(3)：433-436.

[8]　沈煌辉，王贞松，郑为民. 面向图像转置和分块处理的一种高效内存访问策略[J]. 计算机研究与发展，2013，50(1)：188-196.

[9]　边明明，毕福昆，汪精华. 实时 SAR 成像系统矩阵转置方法研究与实现[J]. 计算机工程与应用，2011，47(22)：117-119.

[10]　杜高明，王赵亮，杨欣，等. 一种补齐式准原地转置算法[J]. 电子测量与仪器学报，2015，29(1)：111-118.

[11]　赵欣博，陈星. DDR SDRAM 与 FPGA 的高速接口设计[J]. 电子测量技术，2008，31(11)：182-183.

[12]　ZEMCIK P. Hardware acceleration of graphics and imaging algorithms using FPGAs[C]//Proceedings of the 18th spring conference on Computer graphics. ACM，2002：25-32.

[13]　张刚，贾建超，赵龙. 基于 FPGA 的 DDR3 SDRAM 控制器设计及实现[J]. 电子科技，2014，27(1)：70-73.

[14]　贺柏根，刘剑，刘伟宁，等. 基于 TMS320C6455+FPGA+ SDRAM 的快速视频跟踪系统设计[J]. 液晶与显示，2014，29(6)：1111-1116.

[15]　孙航，韩红霞，郭劲，等. 基于均值偏移快速算法的红外目标跟踪[J]. 仪器仪表学报，2012，33(05)：1122-1127.

作者简介

李汉清，工程硕士，研究实习员，主要研究方向为嵌入式图像处理。

E-mail：lihanqing00@gmail.com