文章编号:1671-4598(2019)06-0213-05 DOI:10.16526/j.cnki.11—4762/tp.2019.06.046 中图分类号:TP274 文献标识码:A

基于 SoC FPGA 异构平台的魔方快速还原系统设计与实现

卢 仕,张志文,张 寅,万美琳
(湖北大学 物理与电子科学学院,武汉 430062)

摘要:设计了基于 SoC FPGA 异构平台,充分利用 FPGA 和 HPS 各自的优势,实现了一套高性能的魔方快速还原系统;系统由开发板,魔方还原机械结构,CCD 摄像头以及 VGA 显示器组成;FPGA 端实现摄像头图像采集和魔方色块 RGB 值的获取;HPS 端完成颜色识别,运用二阶段算法还原魔方,然后将还原步骤编码之后回传给 FPGA,由 FPGA 中的并行舵机控制模块实现对魔方还原机械结构精准快速的控制,从而完成实体魔方的还原;测试结果表明,对于任意随机打乱的三阶魔方,整个识别以及还原过程在一分钟内完成。

关键词:SoC FPGA;颜色识别;魔方还原;二阶段算法

Design and Implementation of a Fast Rubik's Cube Solving System Based on SoC FPGA Heterogeneous Platform

Lu Shi, Zhang Zhiwen, Zhang Yin, Wan Meilin
(Faculty of Physics and Electronic Science, Hubei University, Wuhan 430062, China)

Abstract: This design realized a fast Rubik's Cube solving system with high performance based on SoC FPGA heterogeneous platform. The system is consist of a development board, a Rubik's Cube reduction machine structure, a CCD camera and a VGA display. To make full use of the advantages of each other, FPGA and HPS have different work. Firstly, the FPGA captures the image from CCD camera and gets the RGB values of each face of the Rubik's Cube. The HPS obtains all the RGB values of all six faces, then recognizes the color to know the initial state of the Rubik's cube. Finally, to solute the Rubik's Cube by using the two-stage reduction algorithm. After the solution steps code is passed back to the FPGA, the parallel servo control module in FPGA completes the restoration of the real Rubik's Cube reduction by controlling the machine structure precisely. The test results show that for any three-order Rubik's Cube, the whole recognition and restoration process is completed in one minute.

Keywords: SoC FPGA; color recognition; Rubik's cube reduction; two-stage algorithm

0 引言

随着科技的发展,各种自动化作业平台在各行各业发挥着越来越重要的作用。我国对自动化智能平台相关技术的研究投入也非常巨大。本文所研究的快速魔方还原系统正是自动化智能平台的一个典型应用场景。

该快速魔方还原系统基于 SoC FPGA 异构平台实现。其中 SoC 采用的是 ARM 处理器,它作为 SoC 中的佼佼者,兼顾性能、功耗、代码密度、价格等多个方面,且第三方支持非常全面。而 FPGA 则以其设计灵活性和高并行著称。在将 FPGA 和 ARM 核相结合后,可以实现非常高效的设计。本文快速魔方还原系统采用的硬件平台是台湾友晶公司提供的 SoC FPGA 异构 DE1-SoC 开发板。该板卡提供了一个以 Intelcyclone VSoC FPGA 芯片建立的强大的硬件设计平台,结合了嵌入式双核 Cortex-A9 和业界领先的 FPGA 可编程逻辑。同时还包括了诸如高速 DDR3 内存、ADC、以太网络等丰富的功能外设。足以满足快速魔方还原系统设计的需求,兼具高性能和低功耗[1]。

1 系统设计

整个魔方还原系统由开发板、CCD 摄像头、魔方还原机械结构以及 VGA 显示器组成。开发板控制部分基于 Intel SoC FPGA 异构框架,FPGA 端和 HPS 端各自发挥所长[2],分工协作,具体框图如图 1 所示。

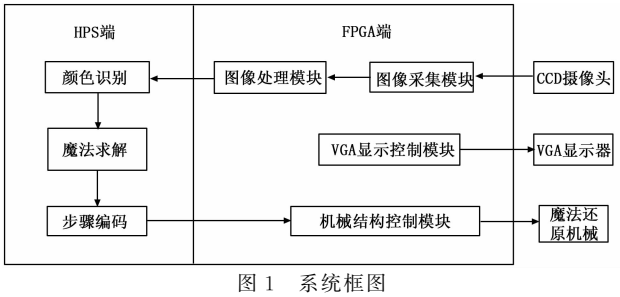

图 1 系统框图

收稿日期:2018-11-29; **修回日期**:2018-12-17。
基金项目:国家自然科学基金(61704050)。
作者简介:卢 仕(1990-),男,硕士,助理实验师,主要从事 FPGA 应用、数字 IC 设计方向的研究。
通讯作者:万美琳(1988-),男,博士,主要从事数模混合 IC 设计方向的研究。

在 HPS 端，分为以下 4 个步骤：

第 1 步，控制 FPGA 部分的图像处理模块和舵机转动模块，依次获取魔方 6 个面的色块中心区域的 RGB 均值。

第 2 步，调用颜色识别算法对魔方每个色块的颜色进行判断，得出初始魔方。

第 3 步，调用魔方还原二阶段算法，得出 30 步阈值以内的魔方还原步骤。

第 4 步，将还原步骤转换并编码后，发送给 FPGA 端舵机转动模块。

在 FPGA 端，主要分为 4 个模块：CCD 摄像头图像采集模块，图像处理模块，VGA 显示模块，舵机控制模块。

1.1 CCD 图像采集模块

CCD 摄像头与 CMOS 摄像头相比，在不管是强光还是弱光的不利条件下，图像画质都要更高[3]。CCD 摄像头输入的是模拟信号，需要经过一系列的处理最终转换成 RGB 图像，整个图像采集的过程如图 2 所示。

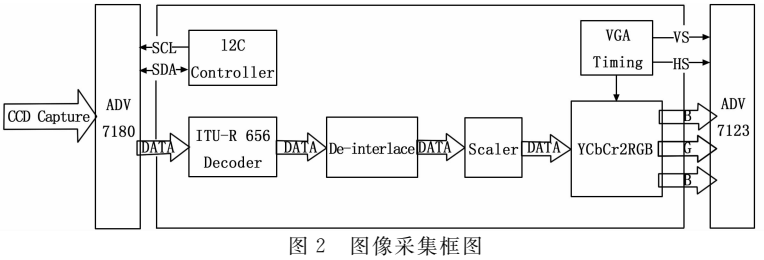

图 2 图像采集框图

CCD 摄像头输入的影像信号送至开发板上的电视译码芯片（ADV7180），译码之后得到 8 位 ITU_R BT. 656 标准接口的影像数据，然后送入 FPGA 芯片。在 FPGA 内，先由 ITU－R 656 译码模块将亮度与彩度信号分解，再送入 De－interlace 模块完成解交错处理，接着送入 Scaler 模块完成缩放处理，得到 YCbCr 格式的图像数据，然后进入 YCbCr2RGB 模块，转换成 RGB 值数据，再根据 640 * 480 像素 VGA 显示需要的 timing，将数字的 RGB 信号送出 FPGA 芯片，进入开发板上的 DA 转换芯片（ADV7123），最终输出三路模拟 RGB 信号到 VGA 显示器。这样，摄像头拍摄到的画面就实时显示在了 VGA 显示器上了。

1.2 图像处理模块

图像采集模块输出的是一帧帧分辨率为 640 * 480 的 RGB 图像，在此图像处理模块中，要做的处理分为三步，首先是要把一帧画面中，魔方表面 9 个色块的中心部分 10 * 10 个像素点的 RGB 值存储到 RAM 中[4]；然后从 RAM 中依次读出每个色块中心区域读出 RGB 数据，求得 R 均值、G 均值、B 均值；最后将 9 组 RGB 值及当前魔方面一起编码后，传送到 HPS 端。

1.3 VGA 显示模块

本设计采用 VGA 显示器作为显示终端，显示的内容分为两部分，在 VGA 的左侧，呈现的是 CCD 摄像头实时获取的画面，右侧则是绘制的魔方的展开平铺图。HPS 端颜色识别的结果会显示在魔方平铺图内，用于人眼直观判断识别结果。此外，通过读取 ROM 中存放的数据，将湖北大学 logo 也显示在里屏幕上，具体显示效果如图 3 所示。

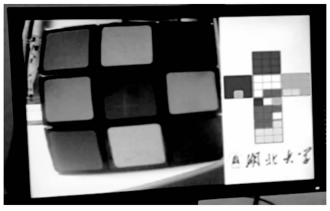

图 3 VGA 显示方式

1.4 舵机控制模块

本设计使用的魔方还原机械结构主要由 4 个机械臂组成，若要机械臂稳定快速的完成魔方面的旋转，对总计 8 个舵机的同步控制尤为重要。本模块的功能就是接收动作编码，通过 8 个 GPIO 端口，同步控制 8 个舵机转动，完成机械手臂的旋转伸缩。

2 FPGA 与 HPS 之间的桥接

在 Intel SoC FPGA 的设计架构里面，FPGA 与 HPS 之间存在两种通信方式，FPGAto SDRAM 和 AXIbridge 接口。FPGAto SDRAM 接口是 HPS 内部的 SDRAM 控制器提供给 FPGA 访问 HPS 内存的接口。AXIbridge 是 FPGA 和 HPS 总线间数据交互的接口，包括 FPGA－to－HPS AXI、HPS－to－FPGA AXI 和 Light－weight HPS－to－FPGA AXI。

HPS 向 FPGA 端需要传输的数据有控制指令、魔方还原步骤编码，数据量较小，选用轻量级的 HPS－to－FPGA AXI Bridge 为传输通路；FPGA 向 HPS 端仅需传输每个面 9 个色块的中心区域的 RGB 均值，通过多组 PIO 传输即可。

3 关键算法

3.1 颜色识别算法

本设计需要做的颜色识别具有两点特殊性：

1）读入的数据是 6 个面，每个面 9 个色块，总计 54 个色块的中心区域 100 个像素点的 R 均值、G 均值、B 均值；

2）每个色块的颜色只可能是红橙黄绿蓝白中的一种。

基于此，我们设计了如下简单高效的颜色识别算法：

1）采集当前环境下，还原后的魔方每个面的每个色块的 RGB 值，存储每种颜色的 9 组 RGB 值；

2）设置一个颜色判定值 $v=a*R+b*G+c*B$，a、b、c 初值均为－128，取值范围从－128～128，变化步径设为 4。三重循环遍历所有可能的 a、b、c 值组合，找到最合适的那组 a、b、c 值。使得某种颜色 9 个色块的 RGB 值算得的 v 值接近，且远大于其余 5 种颜色的色块的 RGB 值算得的 v 值。

3）依据 2）得到的 6 组不同的 a、b、c 值，计算 FPGA 端传过来的 54 组 RGB 的 v 值，依次找出每种颜色的 9 个色块。

3.2 魔方还原算法

3.2.1 魔方的表示方法

采集到魔方状态之后，我们需要用一种方式保存初始

魔方状态。魔方有 8 个脚块，12 个棱块，魔方的摆放为：U 蓝色，F 红色，R 黄色，L 白色，B 橙色，D 绿色。本设计中用如下编码存储。

表 1　角块编码

绿橙白	绿橙黄	绿红白	绿红黄	蓝橙白	蓝橙黄	蓝红白	蓝红黄
0	1	2	3	4	5	6	7

表 2　棱块编码

橙白	橙黄	红白	红黄	绿橙	绿白
0	1	2	3	4	5
绿黄	绿红	蓝橙	蓝白	蓝黄	蓝红
6	7	8	9	♯	11

对于每一个角块，还需要一个参数来确定：顺时针扭转次数。如图 4，对于 3 个魔方前面右上角的的蓝橙黄角块，处于相同的位置，但是处于不同的扭转状态。若选择顶面的蓝色作为参考色，以顶面中心蓝色块为中心，第一个角块的蓝色块已在顶面；第 2 个角块可通过顺时针扭转 1 次，使蓝色块到达顶面；第 3 个角块可通过顺时针扭转 2 次，使蓝色块到达顶面。分别用 0，1，2 表示。

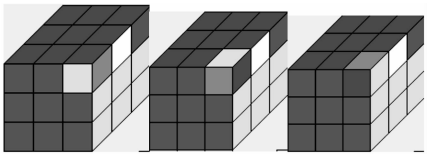

图 4　3 种不同的蓝橙黄角块

而对于棱块，只有两种可能，翻转或者正常。如图 5，两个魔方的前面上方的红蓝棱块。第一个处于正确位置，用 0 表示，第 2 个通过翻转之后可到正确位置，用 1 表示。

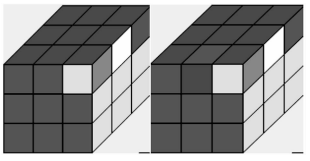

图 5　2 种不同的红蓝棱块

这样，定义一个魔方的数组 int MF［3］［3］［3］［2］［b］［c］（［第 n 行］［第 n 列］［第 n 层］［编号或扭转数］［第 b 步］［第 c 次循环］），用于保存魔方的状态。

魔方有 6 个面，每个面存在 3 种操作（正对该面，顺时针旋转、逆时针旋转、180 度旋转）一共 18 种操作。本设计采用如下方式的简称：F＝front face，前面；B＝back face，后面；R＝right face，右面；L＝left face，左面；U＝up face，上面；D＝down face，下面。以前面 F 为例，顺时针旋转 90°表示为［F］；逆时针旋转 90°表示为［F’］；180 度旋转表示为［F2］。后面则相应为［B］［B’］［B2］，其余面依此类推。

3.2.2　二阶段算法

本设计采用的魔方还原算法，是一种迭代加深启发式的搜索算法（IDA＊）。过程规则很简单，没有很复杂的状态判断，就只是重复循环：对每个阶段的魔方不断重复尝试不同的旋转，然后进行判断是否达到目标状态，如果没有，则根据一个估价函数，选择估价最低的操作继续尝试[5]。

因为魔方每一次的旋转都有 18 种可能，如果每次都做 18 种尝试，循环次数过于庞大。二阶段算法正式为了解决这一问题，将魔方还原分为两个阶段，在第一阶段需要用所有 18 种可能去尝试，但是第二阶段只需要用其中一部分可能转法就确定可将魔方还原。

群是一种特殊的集合，对于一种操作，用它作用于一个集合的元素的时候，得到的结果还是这个集合里的元素时，这个集合对于某个操作构成一个群。而魔方的所有状态中就有着这样的特殊集合，群[6]。对于一个未打乱的魔方，如果你使用 R2、L2、F2、B2、U、U’、U2、D、D’、D2 这 10 种转法来转动它，能生成的状态仅是魔方所有可能状态群中的一个子群。这个子群表示为 G1 ＝ ＜U，U’，U2，D，D’，D2，R2，L2，F2，B2＞。在这个子群中，角块和棱块的扭转是不能被改变的。也就是说，当一个棱块或是角块处在一个特定位置时，它的翻转数和扭转次数是一样的，为 0。同时，UD 夹层（U 层和 D 层中间的那一层，即中层 E）上的棱块始终位于该夹层上。

在第一阶段中，用所有 18 种可能的操作去作用于初始状态，当所有块的翻转数，扭转数为 0，且中间棱块都在中间的时候，则到达 G1 群，第一阶段完成。第二阶段，仅用群的 10 种可能操作作用于此时状态，直到所有块的位置都正确，则魔方还原。

具体流程如图 6 所示。

通过分析计算，第一阶段最高 12 步还原，当代价超过 12 步则舍弃后序操作，尝试另一种操作。第二阶段最高 18 步还原，当代价超过 18 步则舍弃后序操作，尝试另一种操作。算法第一次搜索出的结果一般很快，在 30 步之内，然而，它不会马上停止，而是继续搜索，搜索还没有尝试的操作，如果所有操作都搜索完成，则改变预先的剪枝深度，比如第一阶段为 13 步，虽然第一阶段会提高步数，但是第二阶段可能会减少许多步数，比如从原来的 15 步降低到 8 步，最终步数会进一步减少。当时间达到设定阈值，或者步数达到设定阈值的时候，算法停止工作，并输出结果。

在具体代码实现过程中，用如表 3、表 4 对每一步的操作进行编码。

4　机械结构设计

4.1　关于舵机

舵机由一个步进电机、一个基准电路以及其他的一些部件组成。信号线进来不同的信号时会和基准电路进行比

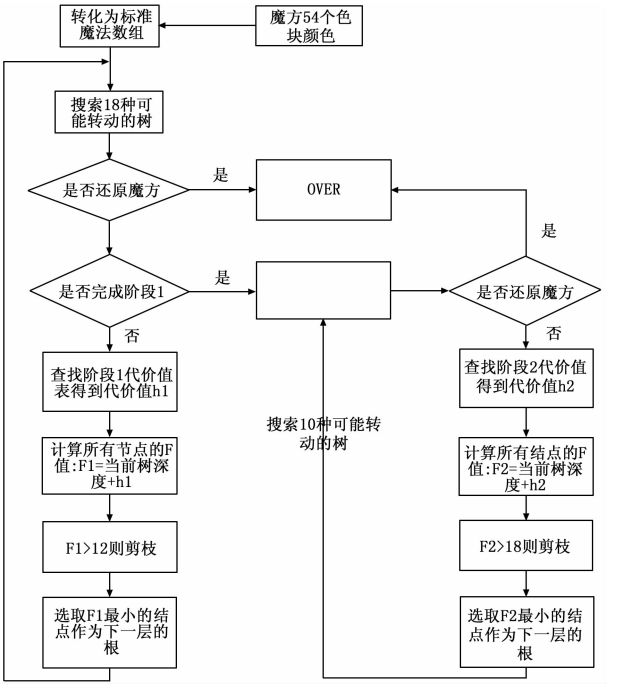

图 6 魔方还原算法流程图

表 3 第一阶段还原步骤编码方式

U	D	R	L	F	B	U′	D′	R′
0	1	2	3	4	5	6	7	8
L′	F′	B′	U2	D2	R2	L2	F2	B2
9	10	11	12	13	14	15	16	17

表 4 第二阶段还原步骤编码方式

U	U′	D	D′	U2	D2	R2	L2	F2	B2
0	1	2	3	4	5	6	7	8	9

较，从而来决定舵机的转动方向。通过改变输入脉冲信号的高电平时间即可控制舵机旋转的角度。在一个周期为 20 ms 的脉冲里面高电平持续的时间决定了舵机转动的角度，对于本设计中使用的 180°舵机，对应关系如表 5。

表 5 高电平脉冲时间和转动度数关系

高电平时间/ms	0.5	1	1.5	2	2.5
转动度数/(°)	0	45	90	135	180

4.2 魔方还原机械机构

本设计采用的魔方还原机械结构由 4 个同样的机械手臂，四面对称摆放构成[7]，单个机械手臂的结构如图 7 所示，一前一后装有 2 个舵机。一个控制机械爪的伸缩，另一个控制机械爪的旋转。整个魔方还原机械结构的实物图如图 8 所示。

4.3 魔方还原动作设计

如第四章所述，魔方还原总计有 18 种不同的操作，但是我们的机械结构只能做到前后左右面每个面的 3 种不同

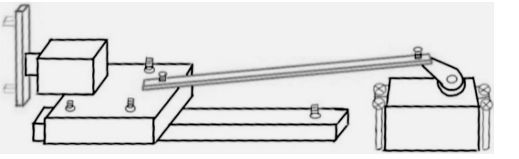

图 7 机械手臂结构图

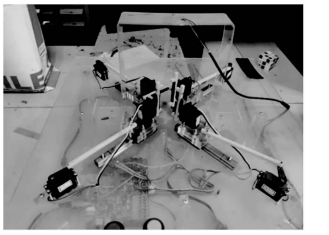

图 8 魔方还原机械结构实物图

翻转，总计 12 种不同操作。同时，若前后两个动作分别是前面和后面或者左面和右面的操作，这两个动作是可以同时执行的。我们对所有可能的魔方操作进行了如表 6、表 7、表 8 所示的编码。

表 6 单面旋转方式编码

F	F′	F2	B	B′	B2
8'h00	8'h01	8'h02	8'h03	8'h04	8'h05
L	L′	L2	B	B′	B2
8'h06	8'h07	8'h08	8'h09	8'h0a	8'h0b

表 7 前后面同时旋转方式编码

F+B	F+B′	F+B2	F′+B	F′+B′
8'h10	8'h11	8'h12	8'h13	8'h14
F′+B2	F2+B	F2+B′	F2+B2	
8'h15	8'h16	8'h17	8'h18	

表 8 左右面同时旋转方式编码

L+R	L+R′	L+R2	L′+R	L′+R′
8'h20	8'h21	8'h22	8'h23	8'h24
L′+R2	L2+R	L2+R′	L2+R2	
8'h25	8'h26	8'h27	8'h28	

对于机械结构无法做到的 U，U’，U2 以及 D，D’ D2 这 6 种操作，需要借助整体转动魔方，转换为可做到的 12 种操作，因此我们定义了表 9 中的魔方整体旋转操作：

表 9 魔方整体旋转编码

夹紧魔方（爪子全部伸出）	魔方整体顺时针旋转 90 度	魔方整体逆时针旋转 90 度	魔方整体顺时针旋转 180 度
8'h30	8'h31	8'h32	8'h33
松开魔方（爪子全部收回）	魔方整体向下旋转 90 度	魔方整体向上旋转 90 度	魔方整体向下旋转 180 度
8'h3f	8'h34	8'h35	8'h36

5　实验结果与分析

根据上述分析，设计实现了魔方还原系统，如图 8 所示。系统工作流程为，打开电源，4 个方向的机械爪全部收回，置于机械结构顶部的摄像头开始工作；放入魔方，按下开始按键，机械爪首先夹紧魔方，然后整体转动魔方，依次将 6 个面暴露在摄像头下；摄像头捕获到的画面传入 FPGA，一方面提取出色块中心区域 RGB 均值送入 HPS 端，另一方面将画面实时显示在 VGA 显示器上；HPS 端执行颜色识别程序，识别得到的完整魔方传入魔方还原程序；还原步骤经编码后送回 FPGA 端的机械结构控制模块，最终由机械结构完成对打乱魔方的复原；魔方还原完毕后，机械抓全部收回，魔方掉出。

为了验证系统的性能，首先单独测试了关键的颜色识别算法和魔方还原算法，然后对整套系统进行了实际的魔方还原测试，具体测试结果如下。

5.1　颜色识别测试结果

基于系统实际工作的环境不同，而光照对于摄像头图像影响非常大，我们分别在光照良好的白天室内、白光灯良好照明下的晚上室内以及黄光灯良好照明下的晚上 3 种不同光照条件下，分别任意打乱魔方 50 次，调用颜色识别算法测试模块检测魔方所有色块的颜色。得到如表 10 的识别结果。

表 10　3 种光照条件下的颜色识别测试结果

环境	白天室内	晚上室内(白光)	晚上室内(黄光)
测试色块数	450	450	450
出错色块数	0	18	9
识别正确率/%	100	96	98

由此可见，系统设计的颜色识别算法，具有很好的适应性。无论是在哪种光照条件下，识别正确率都在 95% 以上。

5.2　魔方还原算法测试结果

设定的步数阈值是 30 步，时间阈值是 5 秒。算法若在 5 秒内找到 30 步以下的还原步骤就会结束搜索，返回结果，若时间超过 5 秒，则直接终止搜索，返回 0。输入 100 个不同的随机打乱的三阶魔方，得到还原时间的分布如图 9 所示。

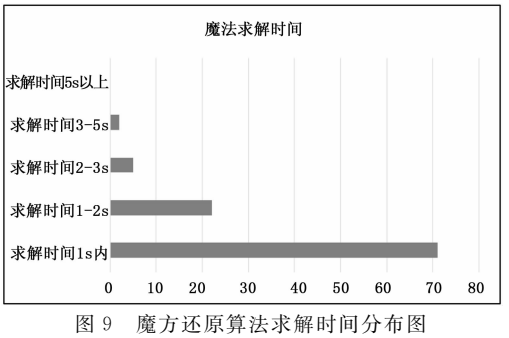

图 9　魔方还原算法求解时间分布图

70%以上的情况，都可以在 1 s 内得到 30 步以内的正确还原步骤，平均还原步数为 25 步。HPS 端系统主频仅有 800 M，算法无疑是非常高效的。

5.3　还原系统整体测试结果

为测试整套系统的稳定性，进行了 100 次打乱魔方的还原。每次还原结束后，记录下结果，然后立刻随机打乱后放入系统启动还原。99 次还原成功，仅在第 92 次还原时，因为前面累计的偏差，导致魔方顺时针旋转 90°时，多转了 10°左右，魔方被转乱，还原失败。魔方采集 6 个面颜色，耗时都在 3 s 左右，求解时间平均在 1.5 s，机械结构还原时间分布如图 10 所示。

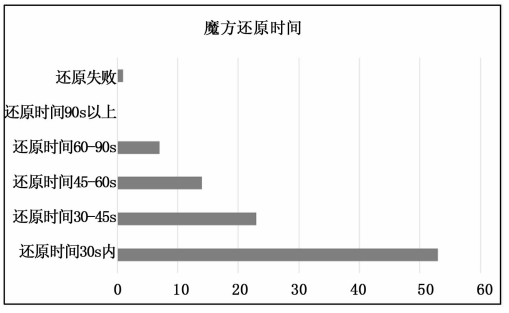

图 10　机械结构还原魔方耗时分布图

99%的还原成功率，验证了系统机械结构的稳定可靠。平均 40 s 的魔方还原时间，验证了系统的高效，也让我们见证了 FPGA 和 SoC 协同工作所能产生的神奇效力。

6　结束语

本文在采用 FPGA 和 ARM 核异构平台的基础上，耗时 3 个月完成了快速魔方还原系统，该快速魔方还原系统由 FPGA 端实现摄像头图像采集和魔方色块 RGB 值的获取；HPS 端完成颜色识别和魔方还原算法，再由 FPGA 实现对魔方还原机械结构精准快速的控制，从而完成实体魔方的还原。测试结果表明，该快速魔方还原系统能够快速地还原魔方，充分验证了 SoC FPGA 协同设计的高效。在未来，我们将进一步发掘两者各自的优势，实现更有价值的智能控制设计。

参考文献：

[1] 张　雄，周艳玲，张子佳，等. 基于 DE1－SoC 的磁性介质显示系统设计与实现 [J]. 计算机测量与控制，2017，25 (3)：184－186，194.

[2] 邓海涛，吴　捷，等. DE1－SoC 开发平台上的图像采集系统设计 [J]. 单片机与嵌入式系统应用 2017，17 (1)，44－46.

[3] 齐　彪，刘　杨，饶　裕，等. 基于 CCD 摄像头的目标跟踪系统设计 [J]. 自动化技术与应用，2017，36 (9)：97－99.

[4] 黄　铭，陆思良，孔凡让. 魔方还原机器人的视觉子系统设计与实现 [J]. 机械与电子，2013 (5)：60－64.

[5] 梁小龙. 解魔方算法的研究和系统实现 [D]. 沈阳：东北大学，2013.

[6] 曹学军. 魔方中的数学 [J]. 学园，2014 (11)：73－74.

[7] 左国玉，刘洪星，顾凌云，等. 四面对称结构的解魔方机器人 [J]. 实验技术与管理，2018，35 (6)：83－86，92.