
基于 Linux 的虚拟试妆系统设计

作者：邢岚 陈锋

导师：何伟

(重庆大学 通信工程学院, 重庆 400044)

摘要: 结合 SOPC 技术裁剪灵活的特点和嵌入式 Linux 操作系统高性能多任务的特性, 设计了基于 Linux 和轻量级图形库 FTK 的虚拟试妆系统。以带 MMU 的 Nios II 处理器和 Linux 操作系统为基础, 采用自行设计的模块实现影像采集和存储, 通过在触摸屏上移植轻量级图形库 FTK 实现交互界面开发, 利用 FTK 的可扩展性编写自定义控件实现特殊交互功能。最终实现了具有商品浏览、模拟化妆、真人上妆等功能的虚拟试妆系统。

关键词: Linux, FTK, 自定义控件, 虚拟试妆

中图分类号: TP316 **文献标识码:** A

Design of Virtual Makeup System based on Linux

Author: XING Lan CHEN Feng

Tutor: HE Wei

(College of Communication Engineering, Chongqing University, Chongqing, 400044)

Abstract: SOPC technology has flexible cutting feature and imbedded Linux operating system is characterized in high performance and multi-task. Combined with the two advantages, a virtual makeup system with the basis of Nios II, Linux operating system and light graphics library was designed in this paper. Based on Nios II processor with MMU and Linux operating system, image acquisition and storage was realized by adopting the self-designed module. The development of interaction interface was realized through the transplantation of light graphics library FTK on LTM. The extensibility of FTK was utilized to compose custom widget and lip-contour extraction algorithm. Finally, the virtual makeup system equipped with the functions of items browsing, simulated makeup and live makeup was realized.

Keywords: Linux; FTK; custom widget; virtual makeup

引言

随着社会的进步, 人们对自身形象愈加重视, 化妆成为女性一门必修课。如何选择合适的化妆品尤其是颜色对于不少女性来说是一项难题。到商场柜台反复试用费时费力, 求助于专业人士则费用昂贵。柜台上反复试妆既繁琐又存在多人试用导致的卫生隐患。近几年针对该问题提出了多种虚拟试妆解决方案。这些方案要么通过浏览器进行本地与远程服务器交互, 要么利用接入 PC 机的影像采集卡和 Visual C++ 开发上位机软件^[1], 由于体积和成本等原因无法在更多应用场合进行推广。

SOPC 技术拥有灵活的设计方式, 具有可裁减、可扩充、可升级等特点^[2]; 嵌入式 Linux 操作系统具有代码量小、实时性高等特点, 支持多种硬件平台和设备, 适合在嵌入式平台上进行大型复杂软件开发。本系统结合两种技术的优势, 以 Altera 公司的 SOPC 技术和 Linux 操作系统为基础, 设计了一款嵌入式虚拟试妆系统。该系统在 SD 卡中按照腮红、眼影、唇彩分类存储信息, 通过摄像头采集用户面部影像, 用户可通过触摸屏浏览、选择商品对影像进行上妆, 上妆效果实时显示在屏上。同时本系统还提供部分模板影像供用户选择, 用户可选择模板观察上妆效果, 最终达到选购化

妆品的目的。系统具有体积小，方便易用等特点，适合于各种商场的化妆品柜台。

1 系统总体结构设计

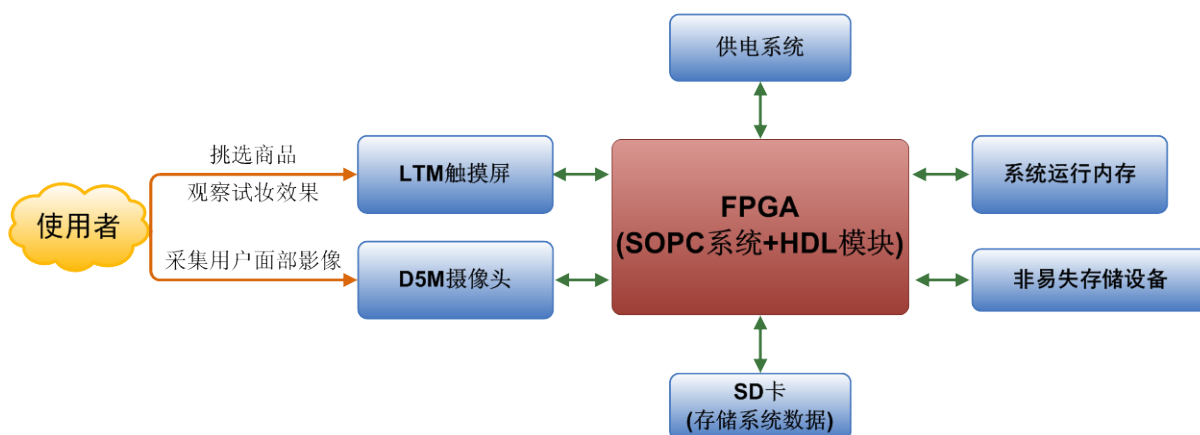


图 1 系统设计示意图

系统采用 FPGA 技术和嵌入式 Linux 操作系统作为方案实现的基础，用户通过触摸屏实现交互操作并得到相应反馈；为使系统具有可扩展的数据库，将所有信息都存储在 SD 卡中，管理者可方便地进行商品更新等操作；另外，除了使用数据库自带的虚拟模特进行试妆效果观察，为使系统能给予用户更直观的信息反馈，增加了摄像头进行用户面部影像采集，使试妆效果能直接与用户本身关联，大大增加了系统的实用性。图 1 展示了本系统设计的示意图。

实际设计中，系统以 DE2-70 为核心板，外接 PS2 键盘录入信息，通过扩展口分别接入 LTM 触摸屏和 D5M 500 万像素摄像头。为使用 Linux 操作系统，构建了基于带 MMU 的 Nios II 处理器的 SOPC。通过 SDRAM 运行内核和 RAM 文件系统，使用 SSRAM 作为显示缓存，将硬件编程代码烧入 EPCS，而将压缩的内核、文件系统与引导代码一起烧入 FLASH。系统采用 Altera 大学计划 IP 中的 PS2 Controller 和 Video IPs 分别管理 PS2 接口和触摸屏的显示部分，自行编写硬件模块实现触摸屏的配置、摄像头采集和 SSRAM 多路复用等功能。此外，系统还使用 SPI 模式管理 SD/MMC 卡以及第三方 IP 进行网络物理芯片管理。系统结构框图如图 2 所示。

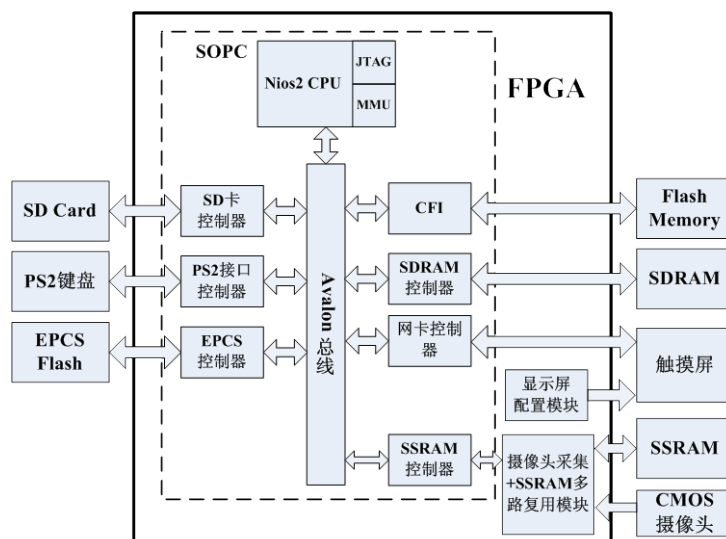


图 2 系统结构框图

2 系统硬件设计

系统硬件设计首先要搭建嵌入式 Linux 运行的最小系统；在最小系统正常运行的基础上通过向 Avalon 总线添加显示子系统相关模块实现 LTM 触摸屏图像显示，同时在顶层设计 HDL 模块配置 LTM 触摸屏使其正常工作；为了使摄像头采集的图像能直接显示在 LTM 触摸屏上，需自行设计模块复用 LTM 的显示缓存，在不影响 LTM 显示前提下将采集到的影像数据导入 LTM 显示缓存中。

2.1 运行 Linux 的最小系统搭建

Linux 操作系统对 SOPC 系统的构建有特殊要求，包括带 MMU 处理器设置、全功能定时器启用、通信终端设定等等。表 1 总结了基于 Linux 系统所需要添加的 SOPC 组件以及添加方法^[3]。

表 1 Linux 系统特性及对应组件搭建

系统需求	模块	实现方法
Linux 虚拟内存管理	Nios II 处理器和双口 RAM	<p>① 在处理器的“Core Nios II”标签页：</p> <ul style="list-style-type: none"> ● 选用“Nios II/f”型处理器； ● 勾选“Include MMU”选项； <p>② 在“Caches and Memory Interfaces”标签页：</p> <ul style="list-style-type: none"> ● 勾选“Include tightly coupled instruction master port(s)”端口数为 1； ● 勾选“Include tightly coupled data mater port(s)”，端口数为 1； <p>③ 新建一个 On-Chip Memory：</p> <ul style="list-style-type: none"> ● 内存类型为 RAM； ● 勾选“Dual-port access”； ● 内存大小设置为 1024 字节； <p>④ 将该内存模块的两个从端口分别与 CPU 的紧耦合指令主端口以及紧耦合数据主端口相连接(见图 3)。再打开处理器的“Core Nios II”属性设置界面，将“Fast TLB Miss Exception Vector”设置为该内存模块。</p>
Linux 系统时钟管理	定时器	<p>选用 SOPC 自带组件“Interval Timer”：</p> <ul style="list-style-type: none"> ● 定时中断周期设为 10ms； ● 计数器位数设为 32 比特； ● 设置为“Full-featured”类型； ● 中断优先级务必设定为最高，即 0。
Linux 通信终端	JTAG-UART 或	内核有且只有一个终端设备，因此二者选一个即可，由

UART	于串口终端反应速度相对较快，故选择该组件作为本系统终端。
Linux RAM 文件系统	Linux 内核在运行时至少需要 8MB 内存空间，由于本系统需要使用图形库进行软件开发，所需内存更多。
SDRAM 控制器	DE2-70 上有两片 32MB 的 16 比特数据位宽 SDRAM，使用时通过设定合适的 SDRAM Controller 参数将其设定为 32 比特数据位宽的 64MB 内存。

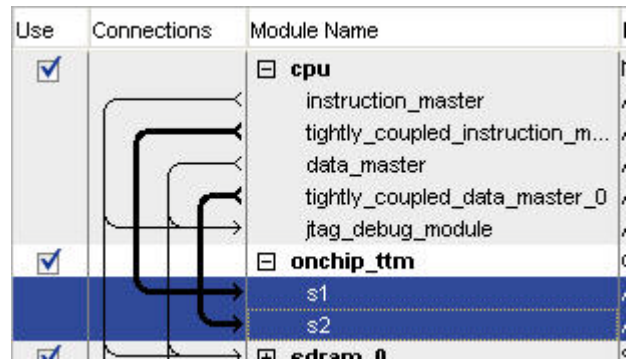


图 3 Nios II 与 TLB 互连

完成最小系统搭建后，先配置内核并下载运行，确保一切工作正常后再进行进一步的设计。

2.2 显示子系统设计

显示部分以 SSRAM 为缓存，Nios II 处理器通过 Avalon 总线将其作为一片连续内存区域进行读写，Altera 大学计划 IP 中的 Pixel Buffer 模块通过总线提取 SSRAM 中图形数据，整合成可显示的数据格式以流式接口传送给 VGA Controller 模块，该模块生成显示所需的行场时序和数据并送入 LTM 显示屏最终获得图像。图 4 展示了显示子系统数据流向。

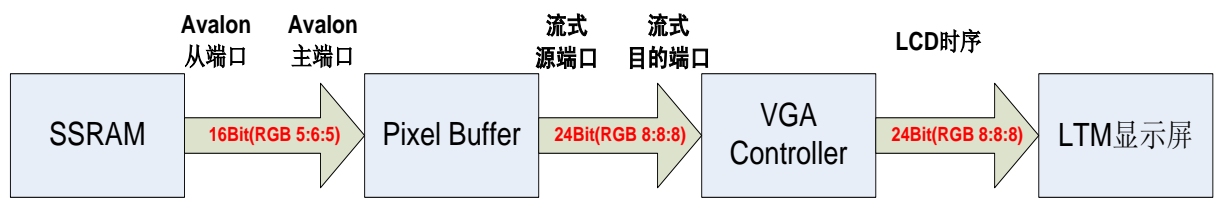


图 4 显示子系统数据流向示意图

Pixel Buffer IP 拥有一个主端口和一个从端口。从端口用于设置工作模式，主端口从显示缓存 SSRAM 中提取图像数据，送至 VGA Controller 模块^[4]。基于软件开发的原因，SSRAM 中单个像素点以 RGB565 格式即 16 比特存储，Pixel Buffer 将 16 比特数据转换为 24-bit 数据满足 VGA Controller 的需要，其中 R、G、B 分量各为 8-bit。Pixel Buffer 支持分辨率硬件放大，即将源数据对应的分辨率通过硬件模块将其扩大 2、4、8 倍。在本系统中，为节省内存设定的触摸屏显示分辨率为 400×240，而 LTM 配置为最大分辨率 800×480，由于 VGA Controller 的分辨率必须与 LTM 一致，因此需要启用 Pixel Buffer 的硬件放大功能，将源数据量放大 4 倍。图 5 为 Pixel Buffer 模块的配置界面。

VGA Controller 模块通过流式接口接收 Pixel Buffer 传输的数据流产生 LTM 所需的时序信号, 包括行、列同步信号等等^[4]。该模块直接与 LTM 相接, 除此之外还需要给 LTM 提供一个 25MHz 的时钟源, 为产生正确的时序, VGA Controller 需 50MHz 的时钟, 是 LTM 的两倍。为保证显示屏显示图像稳定清晰, Pixel Buffer、VGA Controller 以及 SSRAM 控制模块必须使用同一个时钟且为 LTM 控制时钟的 2 倍, 即 50MHz。VGA Controller 的配置界面参数只有两个: 开发板类型和视频输出设备, 开发板类型选择 DE2-70, 视频输出设备则选择 LTM。

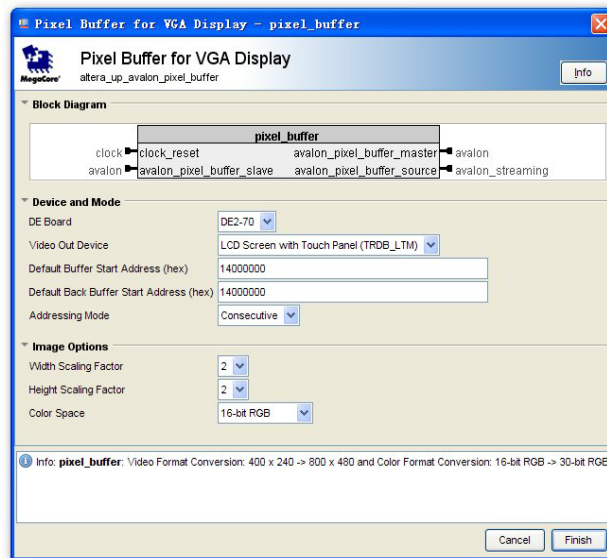


图 5 Pixel Buffer 配置界面

图 6 展示了构建完毕的 SOPC 系统组件列表。值得一提的是, 为优化系统频率, 提高处理器对各个慢速外设的访问速度, 在 Nios II 处理器和各外设间添加了“Avalon-MM Pipeline Bridge”, 使得综合编译后所的系统实际运行频率提高了 8.2MHz。

Target		Clock Settings		
Device Family: Cyclone II		Name	Source	MHz
		clk_50	External	50.0
		clk_25	External	25.0
		pll_c0_system	pll.c0	100.0
		pll_c1_memory	pll.c1	100.0

Use	Co...	Module Name	Description	Clock	Base	End	Tags	IRQ
<input checked="" type="checkbox"/>		cpu	Nios II Processor	pll_c0_system	0x01001000	0x010017ff		
<input checked="" type="checkbox"/>		onchip_ttm	On-Chip Memory (RAM or ROM)	multiple	multiple	multiple		
<input checked="" type="checkbox"/>		sdram_0	SDRAM Controller	pll_c0_system	0x10000000	0x13ffffff		
<input checked="" type="checkbox"/>		ssram	Cypress CY7C1380C SSRAM	pll_c0_system	0x14000000	0x141fffff		
<input checked="" type="checkbox"/>		tristate_bridge_ssram	Avalon-MM Tristate Bridge	clk_ttm				
<input checked="" type="checkbox"/>		epcs_flash_controller	EPCS Serial Flash Controller	pll_c0_system	0x01001800	0x01001fff		
<input checked="" type="checkbox"/>		pipeline_bridge	Avalon-MM Pipeline Bridge	pll_c0_system	0x00000000	0x00ffffff		
<input checked="" type="checkbox"/>		tristate_bridge_flash	Avalon-MM Tristate Bridge	pll_c0_system				
<input checked="" type="checkbox"/>		ext_flash	Flash Memory Interface (CFI)	pll_c0_system	0x00000000	0x007fffff		
<input checked="" type="checkbox"/>		dma	DMA Controller	pll_c0_system	0x008000a0	0x008000bf		
<input checked="" type="checkbox"/>		uart_0	UART (RS-232 Serial Port)	pll_c0_system	0x00800000	0x0080001f		
<input checked="" type="checkbox"/>		pll	PLL	clk_50	0x00800020	0x0080003f		
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART	pll_c0_system	0x00800150	0x0080015f		
<input checked="" type="checkbox"/>		mmc_spi	SPI (3 Wire Serial)	pll_c0_system	0x00800040	0x0080005f		
<input checked="" type="checkbox"/>		sysid	System ID Peripheral	pll_c0_system	0x00800160	0x0080016f		
<input checked="" type="checkbox"/>		Ltm_ctl	VGA	clk_ttm				
<input checked="" type="checkbox"/>		pixel_buffer	Pixel Buffer for VGA Display	clk_ttm	0x00800100	0x0080010f		
<input checked="" type="checkbox"/>		ps2_0	PS2 Controller	clk_50	0x00800168	0x0080016f		
<input checked="" type="checkbox"/>		touch_panel_pen_irq_n	PIO (Parallel I/O)	pll_c0_system	0x00800120	0x0080012f		
<input checked="" type="checkbox"/>		touch_panel_spi	SPI (3 Wire Serial)	pll_c0_system	0x00800060	0x0080007f		
<input checked="" type="checkbox"/>		timer_0	Interval Timer	pll_c0_system	0x00800080	0x0080009f		
<input checked="" type="checkbox"/>		button	PIO (Parallel I/O)	pll_c0_system	0x00800130	0x0080013f		
<input checked="" type="checkbox"/>		ccd_start	PIO (Parallel I/O)	pll_c0_system	0x00800140	0x0080014f		

图 6 SOPC 系统组件列表

2.3 影像采集和存储方案设计

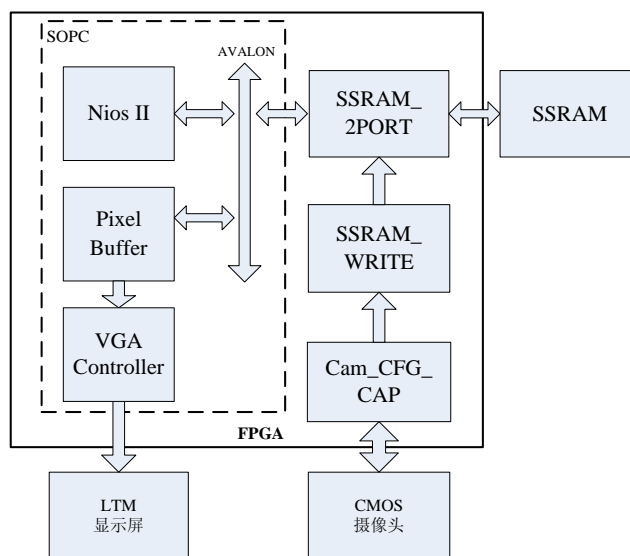


图 7 系统影像采集存储和显示功能示意图

为使系统支持真人影像采集和存储功能，需设计硬件模块将采集到的图像数据导入 LTM 缓存 SSRAM 同时不影响 SOPC 系统对 SSRAM 的控制。图 7 展示了本系统使用的解决方案：在 SOPC 系统外，通过 Cam_CFG_CAP 模块配置和采集摄像头数据，经由 SSRAM_WRITE 模块合成指定格式数据并生成 SSRAM 写时序送入 SSRAM_2PORT 进行仲裁；SSRAM_2PORT 实现二选一的功能，既接收 SSRAM_WRITE 的数据写入 SSRAM 的指定区域，又允许 Nios II 和 Pixel Buffer 通过 Avalon 总线依靠 SSRAM Controller 访问 SSRAM 获取相应数据。下面简要介绍一下两个核心模块 SSRAM_2PORT 和 SSRAM_WRITE 的实现：

① SSRAM_WRITE

当处理器发出图像采集命令时，且 Dval（摄像头的行有效信号）置高时，该模块开始接收 Cam_CFG_CAP 发送的图像数据，将其重新拼接成 32 比特的 RGB 数据，同时生成对应的 SSRAM 写地址。

由显示子系统的设计可知，SSRAM 中存储的单个像素值为 16 位，而 Cam_CFG_CAP 传递的数据格式为 R、G、B 三个分量各 12 位，该模块分别提取 RGB 分量的高 5、6、5 位数据，拼接成 16 位数据，由于 SSRAM 的数据位宽为 32 位，因此一次要写入两个像素点，该模块读入两个像素合成 32 位，同时生成对应的 21 位 SSRAM 地址，将其传递给 SSRAM_2PORT。图 8 展示了该模块的顶层图和主要功能。

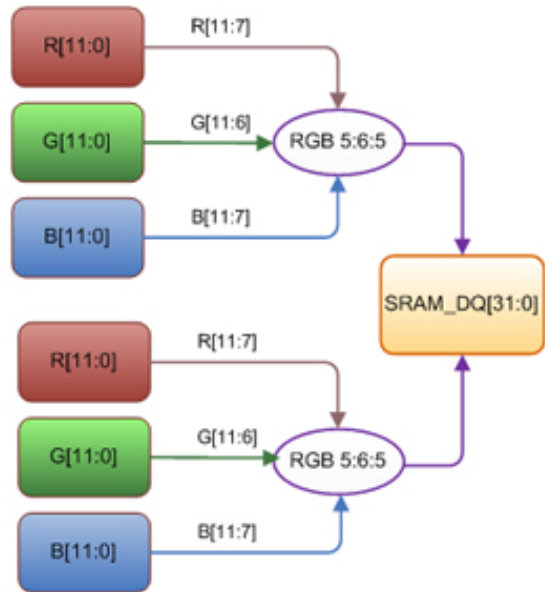
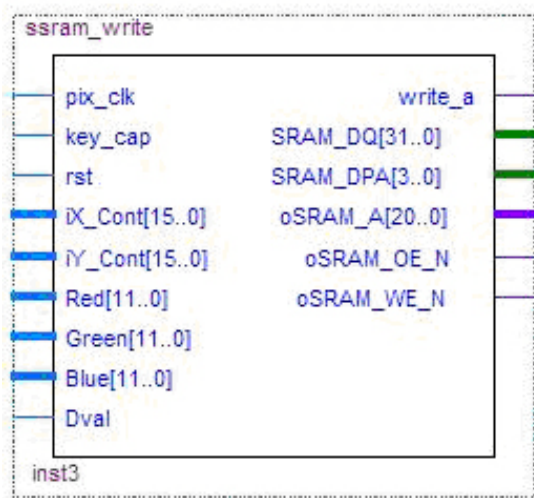


图 8 SSRAM_WRITE 顶层信号和主要功能示意图

② SSRAM_2PORT

该模块对来自 SOPC 系统的 SSRAM 控制器和 SSRAM_WRITE 的读写请求做出仲裁。CPU 读请求有效期间，SSRAM_WRITE 模块读取 SSRAM 中的数据发送给 Avalon 总线；控制器的读请求无效时，SSRAM_WRITE 模块将图像数据写入 SSRAM，此时该模块将忽略控制器的任何写请求。由于 SSRAM 的控制时钟(CLK_SSRAM)与摄像头采集数据的工作时钟(CMOS_PIXCLK)不一致，在该模块中添加两个 DCFIFO 分别作为实时图像数据及其对应地址的缓冲空间，实现不同时钟域的数据同步。图 9 展示了该模块的顶层信号和主要功能示意图。

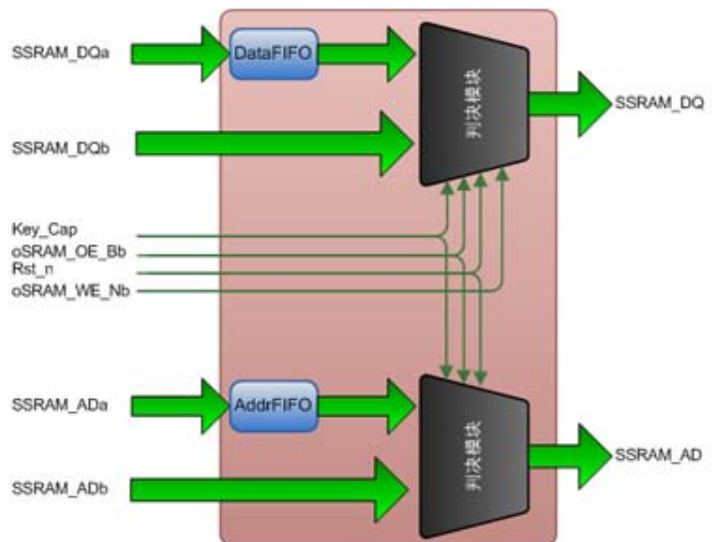
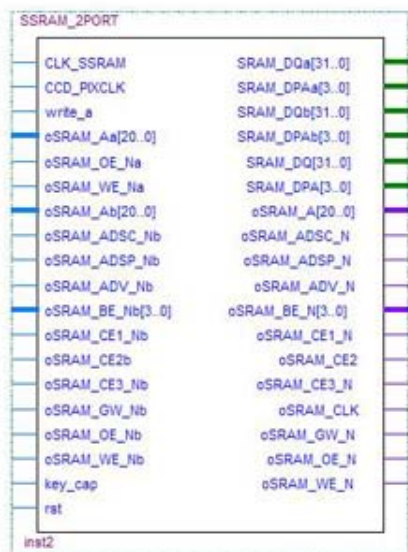


图 9 SSRAM_2PORT 顶层信号和主要功能示意图

3 系统软件设计

系统采用 NiosWiki 社区（现为 AlteraWiki）发布的 nios2-linux-20090929.tar 内核开发包作为软件开发基础。系统硬件设计完毕后，依据由 SOPC 系统组件生成的 C 语言头文件对开发包中内核进行裁剪配置，针对当前系统应用对开发包的源代码、配置文件进行修改，编写相应驱动程序实现外设正常运行；在内核完全支持本系统之后，移植图形库进行虚拟试妆系统应用程序开发。这里着重介绍开发包修改及内核配置和虚拟试妆软件开发。

3.1 开发包修改及内核配置

3.1.1 开发包修改

SOPC 系统构建好后，通过 “sopc-create-header-files --single custom_fpga.h” 生成自定义硬件对应的 C 语言头文件，将该文件拷贝到 “nios-linux/linux-2.6/arch/nios/include/asm/” 路径中。开发包通过 custom_fpga.h 和 nios.h 两个文件中的宏定义管理 SOPC 组件与 Linux 内核驱动代码对设备的映射，同时通过 config.c 文件内置了大量设备的驱动支持，这些设备被定义为平台设备，由内核配置生成的 config.h 文件和之前提到的两个头文件共同控制。图 10 解释了这三个文件之间的关系。

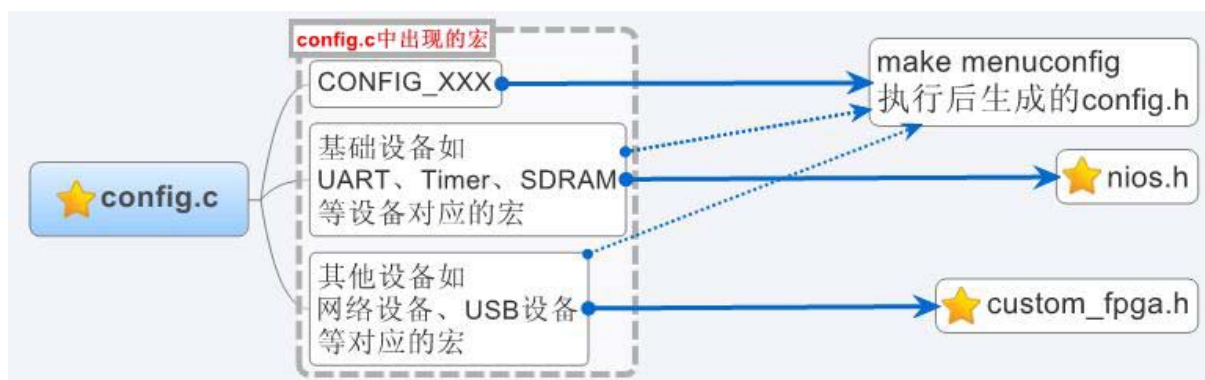


图 10 三个重要文件之间的关系

默认的内核开发包并不能直接使用，需针对本系统进行移植修改，主要内容如下：

- 内核将 UART 设置为终端时发现上电后不打印任何消息。查看代码发现 custom_fpga.h 中的串口宏定义和 config.c 中使用的宏定义不同。将 config.c 中关于串口 0 的宏定义 UART0_* 改为 UART_0_* 即可。
- 内核启动后发现 SD 卡无法挂载。检查发现 /dev/mmcblk* 系列设备的主设备号与注册时不一致，导致挂载操作失败，解决办法是将 "nios2-linux/vendor/Altera/nios2/device-table.txt" 中关于 mmcblk* 的主设备号修改为与注册时一致，即 179。
- 内核启动时早期控制台只能使用 JTAG_UART，这给代码调试和之后的脱机工作带来不便，在 "nios2-linux/linux/arch/nios2/kernel/early_printk.c" 中添加如下代码即可增加串口为早期控制台：

```
#elif defined(CONFIG_SERIAL_ALTERA_UART_CONSOLE)
#if UART_BASE > 0x20000000
#error Cannot map UART directly to IO_REGION
#error please disable early console
#endif
```

```

#define ALTERA_UART_TXDATA_REG          4
#define ALTERA_UART_STATUS_REG         8
#define ALTERA_UART_STATUS_TRDY       (0x0040)
static void early_console_write(struct console *con, const char *s, unsigned n)
{
    unsigned long base = UART_BASE + IO_REGION_BASE;
    int crlf = 0;
    while (n-- && *s)
    {
        while (!(__builtin_ldwio((void *) (base + ALTERA_UART_STATUS_REG))
        & ALTERA_UART_STATUS_TRDY));
        crlf = (*s == '\n' && !crlf);
        __builtin_stwio((void *) (base + ALTERA_UART_TXDATA_REG),
        crlf ? (n++, '\r'): *s++);
    }
}

```

3.1.2 内核配置

开发包的配置菜单为两级，与普通嵌入式 Linux 不同。这是由于开发包中既包含内核源码又包含各种函数库和应用程序源码，因此第一级菜单负责配置整体信息，决定是否进行内核及应用程序配置。图 11 展示了本系统进行的内核配置。

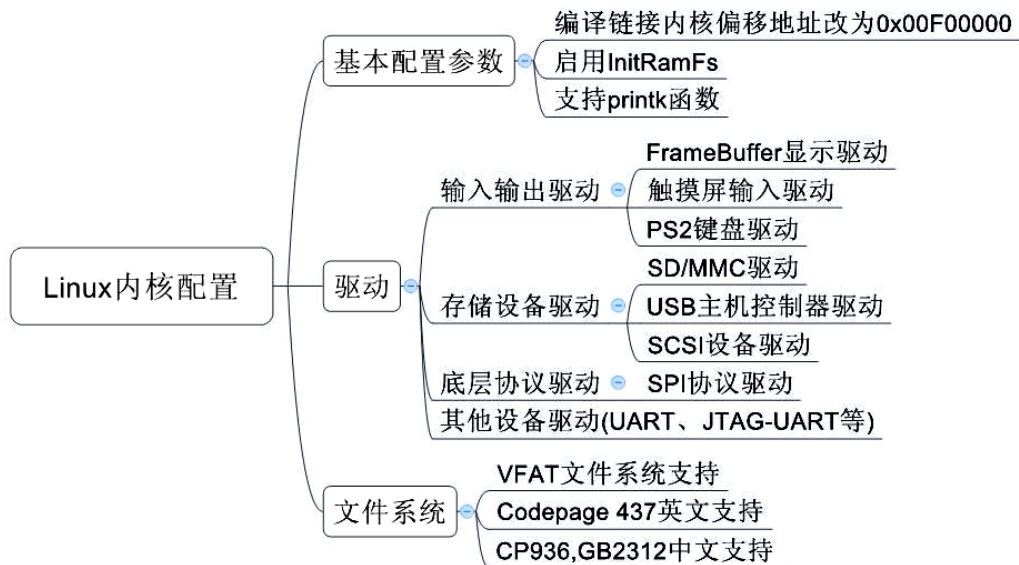


图 11 内核配置

在所有配置中，需要特别注意以下几个部分：

- 内核配置菜单中，需将默认的“Nios II Configuration--->Nios II FPGA Configuration (MMU DEFAULT)”修改为“Nios II FPGA Configuration (CUTOM FPGA)”，使用 custom_fpga.h 作为驱动对应头文件；
- 系统上电将从 Flash 将压缩的内核和文件系统拷贝到内存当中，解压缩内存起始地址为 0xC0000000，而默认拷贝起始地址为解压缩地址加上偏移量 0x00500000(5MB)，这意味着如果内核大小大于 5MB，解压缩的内容将会覆盖当前正在解压缩的文件，导致内核解压缩失败。为防止这种情况发生，通常将偏移地址改得比内核大一倍或更多。故设置“Nios II Configuration--->Link address offset for booting”为 0x00F00000；

-
- 内核开发包发布时有默认的驱动配置，默认使用的网卡驱动与系统不符，要将驱动配置菜单中的“Altera Triple Speed Ethernet MAC Support”去掉，否则内核编译时将由于语法错误导致编译失败；

3.2 应用程序开发

3.2.1 图形库移植

硬件系统设定的运行频率是 100Mhz，为了让用户得到良好的交互体验，需要采用轻量级、代码量小、界面美观，运行速度快的图形库进行应用程序开发，FTK 正好满足这些条件。

FTK 是 Funny Tool Kit 的缩写，它来自于深圳一位工程师，是一个专门为嵌入式系统开发的图形库，其核心代码只有几百 KB，在传统控件基础上可开发自定义控件，具有主题设置功能^[5]。FTK 支持 Linux 系统，因此在发行版 Linux 操作系统上安装 FTK 函数库只需下载相应代码，编译安装即可。而 Nios II 平台移植 FTK 需解决以下几个问题：

- FTK 支持 png、jpeg 和 bmp 三种图像格式，默认使用 PNG 格式作为主题图片，而开发包中 Nios II 平台的 png 和 jpeg 编解码函数库无法使用，同时 FTK 的 bmp 解码函数采用定义整形指针的方式如“`(*(unsigned int*)0x01000000)`”对内存进行访问在 Nios II 处理器中会触发异常中断。
解决方法：首先，移除 FTK 对 png 和 jpeg 格式的支持，在 src 目录下将 rules.mk 脚本中“-DHAS_PNG -DHAS_JPEG”宏定义去除；其次，修改 src 目录下 ftk_image_bmp_decoder.c 代码将 4 字节指针访问修改为 4 个单字节指针访问。
- PC 机上使用鼠标对屏幕进行操作。而 Nios II 平台则通过触摸屏设备进行人机交互。需要 Nios II 平台下 tslib 的支持。解决办法：内核开发包中开启 tslib 函数库，而 FTK 源代码进行 configure 配置时添加“--enable-tslib”参数。
- 实测触摸屏时发现噪点较多。经过调试观察，发现 tslib 在处理物体离开触摸屏那一刻计算得到的数据与处于触摸状态时值偏差较大。解决办法：修改 src/backend 目录下 ftk_source_tslib.c 代码，将物体离开触摸屏一刻采样的值获取为上一个状态值。

系统开发时采用如下模式：在主机上的 Linux 系统中将 FTK 配置为与触摸屏显示相同的分辨率进行应用程序开发，工程开发测试完毕后，直接将代码移植入开发包中交叉编译即可。

3.2.2 交互界面设计

虚拟试妆系统包含唇彩试妆和自由试妆两大功能。在唇彩试妆功能中，用户首先要选择进行试妆所需的唇部影像，可从系统自带的虚拟模特中挑选，也可通过摄像头获取自身唇部影像；选定图像后用户需确定唇部区域，考虑到系统的主频不高且算法识别唇线的时间和准确率都不太理想，因此系统选择由用户使用触摸笔来描绘唇部区域；一旦区域得到确认，用户便可进入商品挑选环节，商品本质上是 RGB 颜色的特定组合，因此商品列表是一系列颜色与指定名称的集合，此外通过 FTK 提供的 alpha 通道进行颜色透明度的调节从而实现商品在嘴唇上浓淡效果的不同。自由试妆同样也要经历以上步骤，不同的是自由试妆可获取整个脸部的影像，所选商品包括腮红、眼影等。用户可在面部自由发挥。

通过对软件功能分析发现，两个功能都需要经历模板选择和商品挑选等步骤。系统分辨率为

400×240，为确保影像清晰设定其分辨率为 300×204，因此单个界面均无法完成两种功能，需将每一个步骤设计成一个界面，图 12 展示了试妆软件功能界面划分的思路及程序流程图。

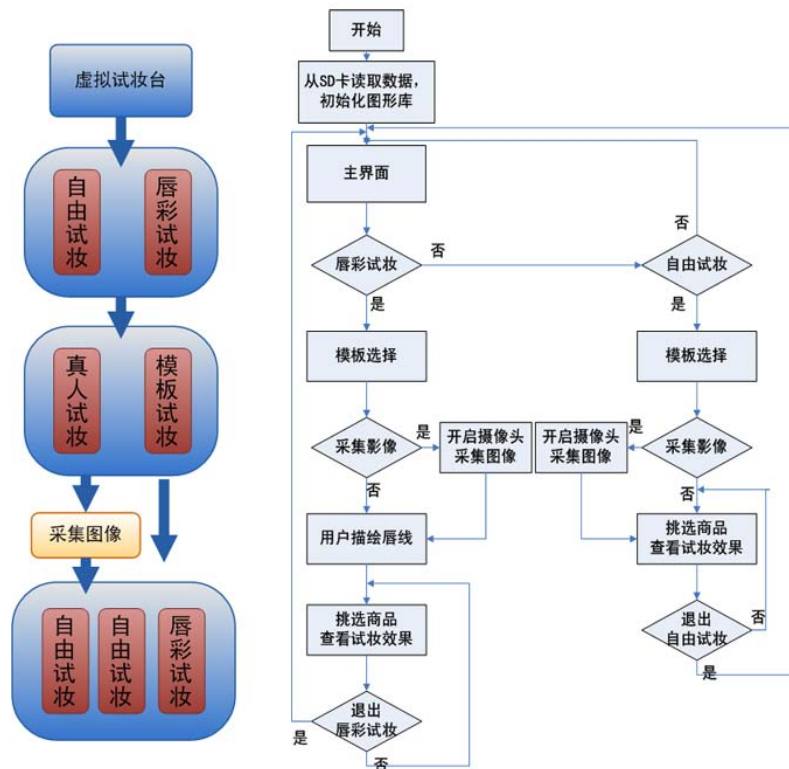


图 12 虚拟试妆软件功能界面划分和软件流程图

由于 FTK 是一个新的开源项目，不具备集成开发环境，因此开发基于 FTK 的应用程序需熟悉图形库的源代码，了解 API 函数如何使用，FTK 中基本函数及作用见表 2。

表 2 FTK 基本函数介绍

API	功能
ftk_init	FTK 初始化函数。在应用程序中必须在其 API 被调用前调用，且只能被调用一次。
ftk_app_window_create	创建一个窗口。
ftk_dialog_create	创建对话框。
ftk_widget_set_text	设置控件的标题。
ftk_button_create	在(x, y)创建宽为 width，高为 height 的按钮。
ftk_button_set_clicked_listener	设置按钮对应的触发函数。

FTK 界面开发包含以下几部分工作：

- 调用 ftk_init 进行初始化。调用 ftk_app_window_create 函数创建窗口，并调用各种控件相关函数设置控件的大小及摆放位置。FTK 没有 GUI 设计工具，只能通过编写代码进行界面的微调，这也是主机 FTK 函数库分辨率和 Nios II 平台必须一致的原因。
- 使用类似 ftk_XXX_set_clicked_listener 函数关联控件和响应函数。试妆系统每个功能下均有多

个界面通过按钮进入，因此在响应函数中需通过调用 `ftk_app_window_create` 或 `ftk_dialog_create` 创建新的界面。

- 设计自定义控件，包括用户浏览图片控件、描唇控件等。

FTK 具有良好的扩展性，通过对基础控件的继承可实现个性化的功能控件。所有 FTK 控件都有 `struct_PrivInfo` 私有结构体及五个基本函数，通过设计自定义的私有结构体以及基本函数行为可实现自定义控件的设计。表 3 展示了控件的基本函数。

设计图像切换控件时，以 `ftk_painter.c` 为基础，在私有结构体中设计一个循环链表，将需要浏览的图像作为链表的单个元素插入链表中，用户触摸屏幕时控件的 `On_Event` 函数会收到 FTK 系统发送的事件信息，信息中包含有用户触摸的坐标值，通过触摸开始到触摸结束时坐标的变化得到用户运动方向进而从链表中取出相应的图像进行绘制工作。

表 3 控件基本函数

方法	Create	Destroy	Set_Listener	On_Paint	On_Event
描述	创建一个该控件的实例	注销一个控件实例	关联一个监听函数	收到图形系统信号，绘制本控件	本控件接收触发事件，并响应

3.2.3 唇部区域搜索

唇彩试妆功能中需要用户描绘唇线，通过对用户描绘曲线所包围的封闭区域进行不同透明度颜色的上色可实现各种唇彩试妆效果。这里简要描述一下如何搜索获取整个唇部区域。

- 进入手动描唇界面，创建与图像同样大小的数组 `mask[width][height]`，设定初始值为 0。
- 用户描唇过程中，自定义控件获取当前坐标点及绘笔大小，对 `mask` 数组相同坐标区域赋值 `0xFF`，并记录该过程取得的最小坐标值 `Pmin.x`、`Pmin.y` 和最大坐标值 `Pmax.x`、`Pmax.y`。
- 勾勒唇线完毕后，确定封闭区域生长点。根据形状可知 `Pmax` 和 `Pmin` 所组成矩形的中点一定在封闭区域中，故选定该点作为生长点。
- 基于生长点，先纵向向上生长，同时对应 `mask` 中的值填充为 `0x7F`，生长遇到 `mask` 相应位置为 `0xFF` 时停止，转而以生长点纵向下一个点开始向下生长并标记 `mask`。遇到 `mask` 相应位置时停止。然后将生长点水平左移一个单位点，进行同样的操作。
- 经过这样生长后，来到封闭区域左边最初生长点所在行的边界处。然而此时并不一定到封闭区域的左边边界，如图 5 所示。此时需要做进一步的处理。
- 取得左边边界处纵向生长的边界点纵坐标，求其平均，以该平均值为新的生长点纵坐标，以左边界点的左边一点横坐标为生长点横坐标，若当前值已为边界，则跳回最初生长点向右生长，通过 e、f 生长直到结束。若不为边界，则继续 d、e、f 直到条件不满足。

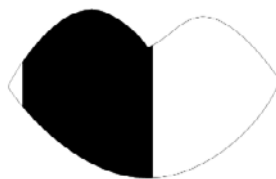


图 13 区域生长示意图

FTK 应用程序要在系统硬件平台上运行，RAM 文件系统中至少需要具备四部分内容：FTK 动态函数库、tslib 函数库、FTK 配置文件(字体、主题、输入法)以及应用程序，动态函数库在编译后会出现在 RAM 文件系统的 lib 目录中，而 FTK 配置文件需通过修改 rc 脚本使 Linux 内核初始化完毕后从 SD 卡中拷贝到内存中。图 14 总结了 FTK 应用程序在 Nios II 平台上运行必备条件及设置方法。

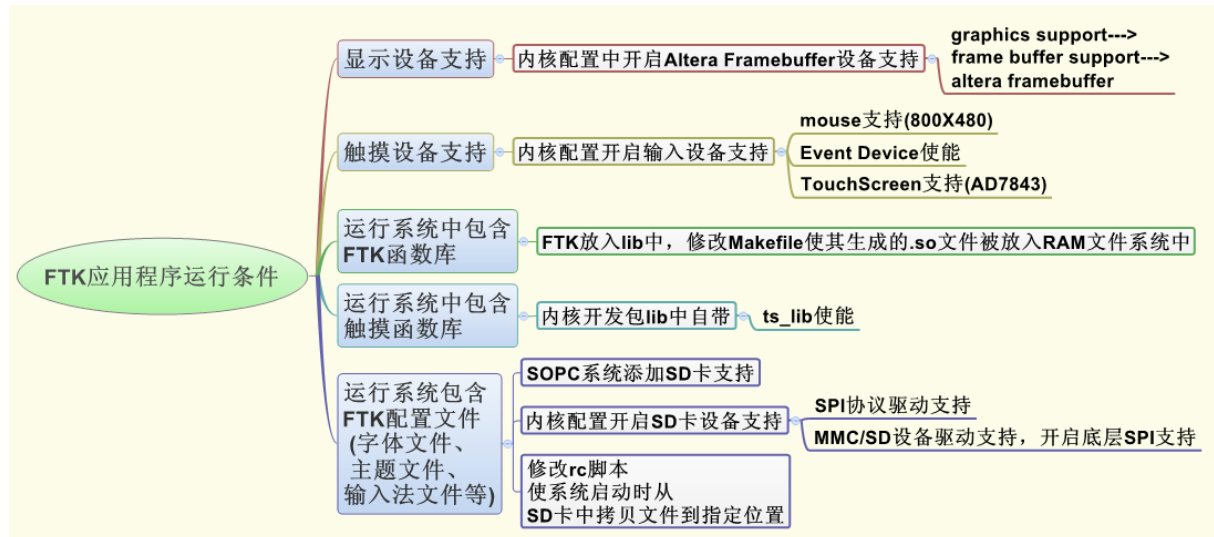


图 14 FTK 应用程序运行条件及设置方法

4 效果展示

4.1 系统硬件搭建实现

系统整体搭建如图 15 所示：由一块 DE2-70 开发板、一块 4.3 英寸 LTM 触摸屏、一个摄像头、一个标准键盘以及一片 SD 卡组成。系统使用 FPGA 为 Cyclone II 系列的 EP2C70F896C6 芯片。如图 16 所示，使用的逻辑单元总数为 9405 个，占 FPGA 总逻辑单元数的 14%，使用了两个锁相环（为系统提供 100Mhz 时钟；给 LTM 触摸屏控制模块和触摸屏本身分别提供 50Mhz 和 25Mhz 的时钟）。生成的压缩内核和文件系统只有 2.9M（图 17）。



图 15 系统实物图

Flow Status	Successful - Tue Aug 10 18:31:44 2010
Quartus II Version	9.0 Build 132 02/25/2009 SJ Full Version
Revision Name	DE2_70_NIOS
Top-level Entity Name	TOP
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	9,405 / 68,416 (14 %)
Total combinational functions	7,588 / 68,416 (11 %)
Dedicated logic registers	5,938 / 68,416 (9 %)
Total registers	6149
Total pins	355 / 622 (57 %)
Total virtual pins	0
Total memory bits	122,232 / 1,152,000 (11 %)
Embedded Multiplier 9-bit elements	4 / 300 (1 %)
Total PLLs	2 / 4 (50 %)

图 16 系统资源使用情况

5.1M	linux.initramfs.gz
3.6M	rootfs.initramfs
16K	rootfs.initramfs.contents
1.7M	rootfs.initramfs.gz
564K	System.map.initramfs.gz
1.3M	vmImage
3.5M	vmlinux
0	zImage
2.9M	zImage.initramfs.gz

图 17 压缩的 Linux 内核+文件系统

4.2 系统效果展示

上电后，系统将自动进入试妆系统主界面。该界面分为两部分：唇彩试妆和自由试妆。点击其中的一个将进入相应部分的操作。



图 18 虚拟试妆系统主界面

4.2.1 唇彩试妆效果展示

①选择模板或影像采集

唇彩试妆的主界面默认展示存储在 SD 卡中的模板，通过在图像区域左右滑动可选择不同模板 (图 19 左)，点击“影像采集”可进行摄像 (图 19 右)。接下来以真人试妆进行介绍。



图 19 模板选择和影像采集

通过 DE2-70 板上的按键可以调节曝光度。



图 20 不同曝光度下采集到的影像

② 描唇

描唇时可选择画笔颜色和大小。



图 21 描唇界面

③ 挑选商品并查看试用效果

在商品选择界面可通过上下拉动滑动条来浏览选择商品，还可以对选择好的商品进行浓度调节。

图 22 展示了商品选择界面以及适中和较浓两种不同浓度下试妆效果。



图 22 商品选择及不同浓度试妆效果

4.2.2 自由试妆效果展示

自由试妆提供给用户一个发挥个人想象力的平台，用户可以随意选取腮红、眼影、唇彩各类上百种颜色使用触摸笔在屏幕上随意勾勒。自由试妆功能中采集影像、选择模板的步骤和唇彩试妆相同，不同之处在于商品的选择，以及上色。采用模板上妆后的效果如下图所示。



图 23 自由试妆效果

结语

详细介绍了本系统的硬件架构和 Linux 开发方法，侧重阐述了 Linux 系统运行的最小 SOPC 系统搭建，利用 Altera 大学计划 IP 构建触摸屏显示子系统、摄像头影像采集和存储模块的实现以及 Linux 内核配置和应用程序开发的重要细节。采用移植的 FTK 图形库完成虚拟试妆软件编写。实践表明，系统运行稳定流畅，效果良好。

参考文献

- [1] 祝秀萍, 刘文峰, 张海峰. 人脸虚拟化妆系统的研究[J]. 计算机与信息技术, 2008(8): 38—42.
- [2] 江晋剑. 基于 SOPC 实验平台的创新型实验方法研究[J]. 微型电脑应用, 2009, 25(1): 28—30.
- [3] Kyledunn. Creating a Nios II Design with an MMU[EB/OL].
http://www.alterawiki.com/wiki/Creating_a_Nios_II_Design_with_an_MMU, 2011.
- [4] Altera. Video Out IP Cores for Altera DE Boards[EB/OL]. <http://www.altera.com/education/univ/unv-index.html>, 2009.
- [5] 李先静. FTK 开发者邮件列表[EB/OL]. <https://groups.google.com/group/funnytoolkit>, 2011.

论文原创性声明:

我们声明所呈交的论文是在何伟老师的指导下进行的研究工作及取得的研究成果，除了文中的参考文献外，论文中不包含其他人已经发表或撰写过的研究成果。

作者签名: 邢 岚 陈 锋

导师签名: 何 伟

撰写日期: 二零一一年四月十二日