Gestural Remote Control Using FPGA

Project Proposal

Andres Hasfura, Ricardo Jasinski, Vladimir Eremin

1 Overview

Universal remotes are infrared controllers that can be programmed to operate many different appliances. Because they must work with different brands and models, these devices usually have a large number of buttons and an unintuitive user interface. The task of programming and using a universal remote is often convoluted, involving elaborate key sequences.

Aiming at providing a better user experience, we propose a gestural remote control able to operate infrared-enabled devices with simple hand movements. In the proposed system, the user's hands are identified by red and green gloves. The X and Y coordinates of each hand's center of mass are calculated and input into a gesture recognition finite state machine (FSM). This FSM classifies the hand movements into a set of predefined gestures, and sends the corresponding infrared command to the controlled device.

Major design decisions include:

- Video is captured with a standard NTSC camera. A dedicated chip in the labkit decodes the NTSC signal and provides the FPGA with digital pixel data, allowing for a hardware-only solution that would not be achievable with a USB camera.
- Infrared commands are encoded using the Sony™ Infrared Protocol. This allows the remote to be used with a wide range of existing devices.
- The gestural interface recognizes only a predefined set of hand movements. Even though it could be possible to make the controller learn new gestures, this is not an essential feature, and would add unnecessary risk to the project.
- Audio feedback is provided by playing wave files stored in a compact flash card inside the labkit. The main advantage of this solution is that it does not require any external hardware. On the other hand, it requires additional FPGA resources to control the flash memory interface, and to communicate with a host computer for uploading the audio files.

2 Implementation

The proposed design is organized into four major blocks: video acquisition, gestures recognition, IR control, and audio playback. The image acquisition block processes the NTSC video signal acquired from a camera, and provides the X and Y coordinates of the user's hands. The gesture recognition block classifies the hand movements into a set of predefined gestures, and sends the commands associated with each gesture to the IR controller and audio playback modules. Figure 1 shows a detailed block diagram of the proposed system. Each of the system's blocks is described next.

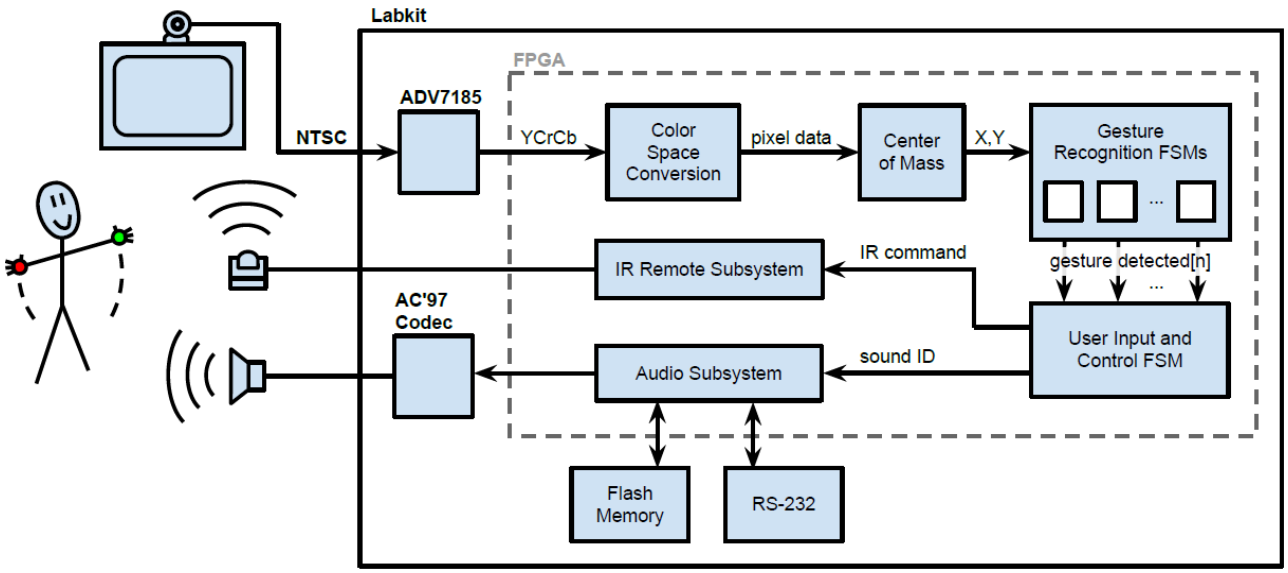

Figure 1. System block diagram.

2.1 NTSC Video Decoder

This module receives the input NTSC signal from the camera and outputs a 30-bit YCrCb value along with other control signals, such as horizontal and vertical synchronism pulses. The actual analog to digital conversion is performed by an external ADV7185 video decoder chip. The main task of this module is to initialize the decoder chip and to adapt the YCrCb pixel data into a format that can be readily used by the subsequent modules.

2.2 Color Space Converter

This module receives YCrCb pixel values from the video decoder and transforms the color encoding from YCrCb to HSV or RGB (the actual format will be determined in the future, depending on the need of implementing color filtering). These color conversions are linear transformations and can be performed using hardware multipliers and instances of the divider module provided by Xilinx. Because the division is usually pipelined, this block may add a considerable delay (many clock cycles) to the acquired video signal.

2.3 Center of Mass Calculation

This module takes in the HSV (or RGB) values of each pixel in an image frame and outputs the center of mass of the user's hands. The x coordinate of the center of mass if found by summing all the products $x \cdot intensity(x)$ and dividing the result by $\sum intensity(x)$. This operation can be performed easily on hardware with the use of accumulators and dividers. As already noted, the dividers adds a delay to the video frame that must be accounted for in the subsequent system modules.

2.4 Gesture Recognition FSM

The Gesture Recognition (GR) FSM classifies the user's hand movements into a set of predefined

gestures. The inputs to this module are the X and Y coordinates for each hand, and the outputs are a set of control signals indicating when each gesture has been recognized.

Internally, the GR module is composed of a series of minor state machines, one for each gesture. The state machines are reset when the hands leave the field of view (FOV) or are placed in a default position (e.g., both hands down, or waving to the camera). Each machine transitions through its states as the hands are moved, and asserts a status line when a gesture is completed. Table I shows some examples of gestures recognized by the GR FSMs, and the corresponding infrared commands.

Table I. Sample gestures recognized by the GR FSM and corresponding functions.

Gesture	**Function**
Waving to the camera	Activate gesture recognition
Holding hands together	Deactivate gesture recognition
Clapping hands	TV Power
Swiping left and right	TV Channel -/+
Holding right hand up	TV Volume +
Holding left hand up	TV Volume -

This module should not present a performance bottleneck to the system, since its inputs are simply four integer numbers (the X and Y coordinates for each hand) and no complex computations are necessary. Additionally, because hand movements occur at a relatively low speed, we expect that 10 to 20 updates per second will be enough to provide a good responsiveness.

2.5 User Input and Control Module

The User Input and Control Module (UICM) initiates and controls a sequence of events when a user gesture is recognized. The inputs to this module are the hand movements identified by the gesture recognition FSM, and the outputs are the control signals to communicate with the infrared and audio subsystems.

This module operates with simple control lines as inputs and outputs, and is not expected to present a performance bottleneck to the system. Differently from the other modules in the system, the UICM is configurable and stores its settings in internal registers, in a table-like data structure. Each row associates a gesture ID with an infrared command and an audio file, which are forwarded to the corresponding subsystems when the gesture is detected.

2.6 Infrared Remote Subsystem

The infrared remote subsystem is used to send the desired commands to the controlled device. Its inputs are a command signal (7 bits wide), a device address (5 bits wide), and a start pulse. With these inputs, the subsystem will send the infrared sequence to produce the requested command. This module is very similar to lab 5b, and can reuse most of its source code.

2.7 Audio Subsystem

The audio subsystem provides an audio feedback to the user whenever a command is issued by the UICM module. Its inputs are a sound selection signal (8 bits identifying a specific sound file) and a start pulse. The audio subsystem uses the sound select signal to index into the compact flash memory and pick the corresponding wave file. The sound is loaded from memory and played through an external speaker.

This module is significantly complex due to its interface requirements. During the initial system configuration, it must interface with a PC via a serial port and write the audio files to the non-volatile flash memory. During normal operation, it must read the flash memory and output audio data to the AC'97 codec. The codec interface can be reused from existing lab 5 source code; however, the flash memory interface will have to be implemented from scratch.

We expect that this module will not present any performance bottleneck to the system for two reasons. First, after receiving a start pulse, this module operates in parallel and is independent from the rest of the system. Second, write operations (which are typically slow in flash devices) are required only during the initial system configuration; during normal operation, only reads are necessary.

3 External Components

The proposed design requires few additional components. The main external peripheral is a standard NTSC camera, which provides the video input for the system. Because the labkit has a composite video input and a NTSC decoder, no extra hardware is required to use the camera.

Audio and visual feedback also exploit the existing labkit resources. The LEDs and hex displays will be used to provide debug and status information. Sound will be generated with the built-in AC'97 codec and output with the standard speakers available in the 6.11 lab.

Finally, the user will be required to use a pair of colored gloves (red for the right hand, and green for the left hand). These gloves are the only external component that must be purchased for the project.

4 Testing

Our testing approach will be based on three principles: extensive behavioral testbenches for data-oriented blocks, simple testbenches and early hardware testing for interface blocks, and continuous integration for the whole project.

In order to provide for better testability, the system was divided into small modules with well-defined functions and clear interfaces. The more data-oriented blocks (e.g., color converter, center of mass calculation, gesture recognition FSMs) will be thoroughly tested with behavioral testbenches to ensure their correctness. On the other hand, blocks that interface with external components (e.g., NTSC decoder, flash memory controller, AC'97 interface) will be tested with

simpler testbenches that only verify their basic operation. However, these modules will be coded and tested first, in order to detect potential problems early and to exercise them for as long as possible in the course of the project.

The source code for project will be put under version control (using SVN) and every developer will have access to the latest version of each file. The synthesis tool will be configured to use source files directly from SVN, providing for continuous integration of all system modules. This will allow us to detect problems early when a change inadvertently breaks the build.

5 Task Breakdown

The modular approach used in the system design facilitates the division of labor, because each block can be handed to an individual team member. The development of each block will be considered complete only after the block has been coded and tested with the respective testbench and physically in the labkit. Table II shows the initial tasks assigned to each team member.

Table II. Division of labor.

System Block	**Team Member**
NTSC Decoder	Vladimir
YCrCb to RGB/HSV	Vladimir
Center of Mass	Vladimir
Gesture Recognition FSM	Ricardo
User Input and Control FSM	Ricardo
IR Remote Subsystem	Ricardo
Audio Subsystem	Andres
PC↔FPGA Serial communication	Andres
Flash Memory Interface	Andres