6.111 Final Project Proposal - J. Abel, J. McGuire
Quadcopter Camera Display

Table of Contents

Project description:

We propose to construct a system that will take a video signal from a camera mounted to a quadcopter and display it on a screen. An analog NTSC composite waveform will be transmitted from the camera wirelessly on a 5.8 GHz carrier, received and extracted by an RX receiver chip, and digitized using an NTSC decoder. The incoming pixel data will be transferred in YCrCb colorspace and stored in a buffer in the external ZBT RAM chip until an entire frame has been received, to account for the interlaced nature of NTSC video. Once the entire frame is in the memory, the pixel data will be read in 3x3 chunks, addressed based on its location in the frame, and sent to an image processing module. The image processing module will then apply a series of filters to each chunk as toggled by the used, including a brightness level adjuster, an edge highlighter, and a noise reducer. The processed pixels will then be sent to a VGA controller, hence displayed the view from the camera on the screen.

The analog NTSC composite waveform will be digitized and processed using an ADV7185 NTSC decoder, which samples the waveform at 27 MHz with a bit depth of 10, resulting in a 30 bit data structure storing the data for each pixel in YCrCb space. Additionally, the NTSC decoder can extract the video synchronization signals from the waveform, which are necessary to later display the image on the VGA monitor. This 30 bit data input will be truncated to 18 bits such that two pixel values can fit into one 36 bit memory word. A register will enable the storage of the past value so that both two pixel values can be latched into a 36 bit register on every odd numbered pixel clock count. The address translator module will coordinate the latching of this register and send a ready pulse to the memory deconflicter module when a new word is available. Additionally, a color bar generator module will be selectable by a switch to generate a consistent image for testing purposes. Similar to the other input chain, this will generate pixel values, locations, and an input clock. Note that the address translator latches out values on the system clock so that synchronization with the memory deconflicter will be acheived.

The memory read logic operates on a two-by-three word buffer that allows for the storage of 12 pixel values. On every odd pixel clock, the buffer will advance, and on every even clock, the 3x3

pixel output frame will shift. In this manner, nine adjacent pixel values will be output at a time to the convolutional processor. These scheme requires the reading of three word values from the external memory for every two output pixels. Therefore, there are 10 clock cycles available in this space to do three reads and the single write operation from the pixel input. The memory deconfliction module will accept the three addresses from the output controller for each two-pixel cycle serially and will therefore take priority for memory access. In the seven cycles that occur between these accesses, the input word will be saved to memory. Delay registers will be used to delay the latch outputs from the memory unit (which is delayed by two cycles) such that the proper data value is placed into the proper register on the memory output buffer before being latched into the 2x3 word buffer for output. The memory deconfliction module will use read/write and trigger lines to control when reads and writes should occur and which values should be multiplexed to the actual address lines.

Another feature of the input unit is the average and variance calculation, which will be useful for establishing thresholds later on in the image processing. This operates off of a two-frame average computation that receives pixel inputs. The difference between this two-frame average and the current input pixel value will be averaged over an entire frame to determine the variance for the current frame, which will be latched with the vertical sync pulse such that it arrives at the output one frame delayed.

For initial tests, this video feed will be directly supplied to the decoded via a wired connection, to ensure that a clean signal is entered into the system. However, to increase the applicability of the system in the real world, this feed will be replaced by a wireless feed received from a RunCam Owl Plus camera mounted to a quadcopter. This camera transmits the signal on an AM-modulated 5.8 GHz carrier wave, which can be decoded using a RX5808 receiver with an antenna attached to the video decoder.

As each pixel is received, its data will be truncated from 10 bits per channel to 6 bits per channel and stored in a register on the FPGA's ZBT SRAM chips. Since NTSC video signals are interlaced, the entire frame will need to be stored in this buffer before processing can be applied to the image, hence necessitating the external SRAM chips. Assuming that each frame will have a resolution of 640x480 pixels, we expect that around 700kB of memory will be required to store each frame. The system will contain two frame buffers; one to store the incoming pixels, and one to store the outgoing pixels as they are transferred to the image processing modules. A flag will be used to switch the roles of these two buffers between each frame, such that data does not need to be transferred between two buffers. The outgoing data will be sent in 3x3 chunks of pixels alongside Cartesian addressing information, which makes it easier to later apply convolutional filters to the pixels.

Filters will be applied to each 3x3 chunk of pixels before they are sent to the VGA controller. These filters will be toggled by the user using the switches on the FPGA. Since many of the filters work by manipulating the Y channel of each pixel, filtering will occur in the YCrCb color space. The following filters will be available to process the image

- An edge highlighting filter based on a 3x3 Sobel edge detection convolutional filter, which will highlight the edges of obstacles in the frame to make them more visible to the user. This filter will only act on the Y channel.

- Two brightness level adjustment filters will be added to increase the contrast in the image when overexposed (e.g. in direct sunlight) or underexposed (e.g. in twilight). These filters will work by remapping the values of the Y channel of each pixel, such that the distribution of brightness levels for each pixel is more even. These two behaviors will be attached to two different switches; the behavior that is chosen will be the switch that was most recently switched on (i.e. the filter cannot increase and decrease brightness).

- A noise reduction filter, which will smooth the effect of noise due to errors in the signal transmission, hence increasing the effective range of the system. The 3x3 pixel chunk will be convolved with an approximated Gaussian mask, which has been modified such that all values are factors of 2 to remove the need for floating point arithmetic. This mask will act on all three channels of the pixel.

This processing must occur within one period of the VGA clock (the inverse of 12.588 MHz, assuming a resolution of 640x480 pixels with a frame rate of 30 fps). Moreover, up to nine hardware multipliers will be required to implement all of these filters. Once the selected filters have been applied, the pixels will be converted from 18 bit YCrCb values to 18 bit RGB values and sent to the VGA controller to be displayed on the screen.

Additionally, the displayed image will be overlaid with text indicating the total power of the video signal. A second ADC will be used to sample the incoming NTSC waveform, and calculate the overall power of the signal. This value will then be displayed as text sprites stored as ROMs in the FPGA's on-chip memory as overlays on the image. The user will be able to select the position of this overlay from 6 locations on the edges of the screen using the buttons on the FPGA. An FSM will be used to drive the logic controlling the position of this text overlay.

System Components:

The system architecture will consist of the following components:

NTSC Camera	Data from the surroundings will be captured using an NTSC camera. Initially we plan to use a wired camera, which directly sends an analog feed to an NTSC decoder chip. Since this setup has minimal losses and requires no additional circuitry, it is ideal for testing and demonstrating the functionality of the rest of the system. However, we hope to eventually replace this with a RunCam Owl Plus camera mounted to a quadcopter, which transmits video data over a 5.8GHz AM modulated RF signal, which will give our system greater real-world applicability.
ADV7185 NTSC Decoder	This chip converts the analog analog NTSC waveform into a digital waveform that can be processed by the FPGA. Essentially the chip acts as

	an ADC, sampling the video waveform from the camera at 27 MHz with a bit depth of ten. Once digitized, the chip then extracts the pixel information and outputs it as a 30 bit value in YCrCb color space. Additionally, the chip is able to extract the HSync and VSync signals necessary to control the timing of the video display.
RX5808 RF Receiver with Antenna	This chip will receive the RF signal from the camera and extract the NTSC composite waveform off the 5.8 GHz carrier wave. The video signal wave will then be fed into the NTSC decoder as if it were from a direct feed. The signal would also be fed into an additional low-frequency ADC, which would sample the signal to determine its overall power. For simplicity, we intend to use a single antenna to receive the signal, however there is potential to expand this to a multidirectional antenna array if time permits. This array could be controlled using a switching matrix powered by logic from the FPGA, and would give a stronger signal.
Labkit FPGA with 4MB ZBT SRAM	An FPGA will be used to perform all digital signal processing, as well as to apply the filters to the resultant image. Since NTSC video signals are interlaced, a frame buffer is needed to store complete frames for processing using convolution filters. We expect to need to store up to 4 frames to perform the necessary processing, which requires around 2.5MB of memory. Hence, we have chosen to use the older labkit FPGA, which can be interfaced to two built-in ZBT SRAM chips, providing 4MB of memory. The user will interact with the system using the buttons and switches present on the FPGA. The switches will be used to toggle between the different filters, while the buttons will be used to select the location of the data overlay on the screen. The labkit's built-in VGA bus will be used to connect the system to a monitor to display the images.
VGA Monitor	A VGA monitor will be used to display the processed images from the camera. To match the frame rate of the camera and resolution of the camera, as well as to reduce the memory and clock speed requirements for the video processing, the image will be displayed at 30 frames per second with a resolution of 640x480 (since VGA by default uses 60 frames per second, each generated frame will be displayed for two frame cycles).

Project Goals:

	Camera Signal Processing	Image Processing
Core	Video from an NTSC camera will be received by the NTSC decoder via a direct wired feed. The resultant pixel data will be addressed and stored in memory until an entire frame has been received. Once the entire frame is in memory, a flag will allocate this buffer to be processed, while new data will be added to a second buffer. The outgoing pixel data will then be sent to the video processing module in 3x3 chunks suitable for convolutional filtering.	The pixel data in YCrCb space will be received from the SRAM as an addressed 3x3 chunk. The center pixel of each chunk will be converted to RGB color space, and sent to the VGA control module. This module will take in this pixel data along with its position, and generate the view from the camera on the monitor.
Expected	Decode video from NTSC wireless feed, determine signal power Instead of a wired feed, a video signal will be received wirelessly from a quadcopter-mounted camera. An RX receiver chip will be used to extract the signal from the carrier wave, which will be sent to the same NTSC decoder in the same manner as the direct feed. Additionally, a separate low-frequency ADC will be used to sample the incoming signal and determine its overall power. This value will be sent to the VGA controller and displayed as a text overlay on the screen.	Before converting each pixel from YCrCb to RGB, the system will apply a series of filters to each chunk, which will be toggled by switches on the FPGA. Depending on the user's preference, a brightness level adjustment filter, a Sobel edge detection filter, or a Gaussian noise reduction filter will be applied. Since this processing mostly operates on the Y channel of each pixel, it will occur before the pixel is converted to 18 bit RGB. Additionally, a text overlay will be added onto the frame indicating the signal power. This value will be measured by sampling the overall voltage of the incoming signal. The position of this text on the screen will be controlled using the buttons.
Stretch	The single antenna used to receive the NTSC signal will be replaced with a multidirectional antenna array, controlled by a switching matrix. The switching matrix would select which antenna to read video data from by comparing the signal strength at each antenna. The FPGA will be used to both estimate signal power and control the logic for the switching matrix.	Instead of applying the filters when manually toggled, an option will be available to automatically apply filters to the image based on aggregate values of pixel data. For example, the system may calculate the overall average brightness of an image and automatically adjust the levels of the Y channel to compensate. Additionally, this may be combined with thresholding, which would allow different adjustments to be applied to regions of the image with different brightnesses.

Block Diagram: