SIGNAL PROCESSING ALGORITHMS FOR DSP IMPLEMENTATION OF ANALOG TV RECEIVERS

Hua Ye^{*}, Daniel Iancu^{*}, John Glossner^{†,*}, Vladimir Kotlyar^{*}, Andrei Iancu^{*}

*Sandbridge Technologies, Inc.

1 North Lexington Ave.

White Plains, NY 10601 USA

{huaye,diancu, jglossner, vkotlyar, aiancu}@SandbridgeTech.com

ABSTRACT

This paper presents a novel power efficient software implementation of the NTSC analog TV receiver using the Sandbridge Technologies Software Defined Radio (SDR) platform. The Radio Frequency signal specific to a TV channel is down converted to base band and sampled at 14.318 MHz. From this stage, the entire receiver function is executed in SW, including the horizontal and vertical synchronization, and video processing. The analog TV receiver implemented in DSP can coexist with other communication protocols such as WLAN, GSM/GPRS, GPS etc. allowing analog TV broadcasting on mobile platforms.

1. INTRODUCTION

Nowadays it's becoming increasingly attractive to provide mobile platforms with the capabilities of viewing analog TV broadcasting. Although digital TV and HDTV start being deployed, the analog TV broadcasting (NTSC, PAL, SECAM) is still the most common TV system. A traditional TV receiver consists mostly of analog circuits. At the same time, supporting multiple communication protocols on the same mobile platform becomes attractive. Since each communication protocol requires a separate chipset, PC board space and overall cost are the limiting factors. The Sandbridge Technologies multiprocessor multithreaded SB3010 DSP [3][4] is capable of executing most of the existing communication protocols in SW, including the analog TV receiver. We present the analog TV receiver algorithms and the corresponding SW implementations in the following order: Section 1: Introduction, Section 2 analog TV system description, Section 3: Algorithm description, Section 4: Conclusions, followed by References in Section 5.

[†]Delft University of Technology EEMCS Delft, The Netherlands

2. ANALOG TV SYSTEM DESCRIPTION

The system block diagram of the analog TV receiver implemented in the SB3010 DSP is illustrated Figure 1. The received analog TV signal is first down converted to Base Band. The Base Band signal, or the composite video signal, is then digitized and passed to the DSP. From this stage, the entire video processing is executed digitally.



Figure 1. System Block Diagram of Analog TV Receiver.

In NTSC systems [1][2], each video frame consists of 2 color fields or 525 lines. As shown in Figure 2, the first 20 lines of color field I, is the field-blanking period that carries the vertical synchronization pulses. The vertical synchronization pulses indicate the start of a new video frame. The color field II starts from the middle of line 263, carrying another 20 lines for the second field-blanking period indicating the middle of the video frame. The rest of lines in color field I and II are the displayable video lines.

Figure 3 illustrates the detailed video line signal timing and DC level information. The various information carried by the video line must be constructed at predefined signal levels and timing to allow the receiver to reconstruct the RGB signals correctly. Each displayable video line consists of the following two parts: The "Blanking Period" carrying the "Front Porch", the "Sync Tip" or horizontal sync pulse" and the "Back Porch" with 9 +-1 cycles of color burst riding on it.



Figure 2. Color Field Blanking Sequence for NTSC

The horizontal sync pulse are used by the receiver for line synchronization and frequency / timing offset tracking purposes. The color burst is a sine wave reference signal at the video carrier frequency. It is used to extract the phase and frequency information for decoding the I/Q modulated chroma signal. Following the back porch, is the second part of the video line carrying the actual displayable video information (luminance and I/Q modulated chroma signal).



Figure 3. Details of Video Line Timing / Levels Information

The analog TV receiver digital signal processing blocks implemented in the SB3010 DSP are illustrated in Figure 4. First, the horizontal synchronization and vertical synchronization sequences are detected and tracked by a Delay Lock Loop (DLL). The input video composite signal is then adjusted to achieve proper DC and IRE (Institute of Radio Engineers) scaling levels [1][2]. The active video information is extracted line-by-line as follows: the Y/C, or luminance and color, are separated first, followed by the the RGB color I/Q demodulation and signals reconstruction. The RGB signals for the current video frame are finally converted to pixels and displayed on the LCD screen.

3. ALGORITHM DESCRIPTION

As shown in Figure 5, the receiver goes through a synchronization process to achieve horizontal

synchronization first, vertical synchronization next, and then video extraction and processing can be performed.

In *Receiver State 1* (HSYNC) the receiver goes through two sub-states to achieve horizontal video line synchronization.



Figure 4. Signal Processing Block Diagram of NTSC Receiver

The horizontal synchronization pulses are processed by the Delay Lock Loop (DLL) processing block for correcting any frequency offset that may exist between the transmitter and receiver due to transmitter receiver frequency misalignment or sampling clock drift.

In the sub-state 1 the receiver performs the initial blind search for HSYNC pattern on the sampled video composite signal r_{in} . The HSYNC pattern search is performed using the following equation:

$$R_{rH}[i] = \sum_{k=0}^{N-1} r_{in}[k+i]$$

where $N=MF_{s}$, *M* is the line duration in microsecond, F_s is the sampling frequency in MHz, R_{rH} is the cross correlation of the HSYNCH pattern with the sampled video complex signal at i=0,1,...,L-1, where *L* is the number of samples in a line. The HSYNCH position is found at the sample position *p* for which R_{rH} is the maxim.

Sub-State 2 starts HSYNC Tracking. Once detected, the HSYNC has to be continuously located for over a certain number of lines around the initially detected HSYNC sample position *p*:

$$R_{rH}[p+k] \ge 0.9 \times R_{rH}[p], k = -2, -1, 0, 1, 2$$

If the above condition is not satisfied, the detected HSYNC is declared to be false, a fall back logic is activated allowing the receiver to fall back to the initial HSYNC detection sub-state.

The fall back logic is designed to handle the special situation when the receiver happens to start the HSYNC detection during vertical synchronization periods.

Starting from this sub state, a DLL logic is activated to track the movement of the HSYNC position caused by frequency offset between the transmitter and the receiver. The DLL tracking range is 2 samples left and 2 samples right of the current HSYNC starting position at p as expressed in the above equation.



Figure 5. NTSC Receiver Synchronization Process

The maximum of the HSYNC match results for k=-2,-1,0,1,2 is found and the receiver buffer pointer is manipulated to adjust the maximum HSYNC match position to p. The task of the DLL tracking loop is to skip or repeat up to 2 samples for each video line to track the frequency offset that may exist between the transmitter and receiver. The DLL tracking loop should be enabled once the HSYNC is detected. The DLL tracking loop is temporally disabled during the vertical synchronization period.

In this sub-state, the receiver can also calculate an estimate of an average timing offset in samples. This is meant to handle any constant frequency offset between the transmitter and receiver to allow the DLL to only track any residual dynamic timing drifts. The average timing offset can be estimated as follows:

$$\Delta t = round(\frac{\sum_{i=0}^{19} p_i}{20})$$

Where p_i is the sampling position for $R_{rHMAX}(i)$, for line *i*. **Receiver State 2** (VSYNC): The receiver has achieved line synchronization entering the state, HSYNC / DLL tracking runs continuously to keep the line synchronization and to correct any residual dynamic frequency offset. The main task of this state is vertical synchronization pattern search to achieve video frame synchronization. Since the receiver has achieved horizontal synchronization, the line start sampling position is known. We choose to search for VSYNC pattern as displayed by VSYNC pulses in line 4, 5 and 6 as shown in Figure 2. As the VSYNC patterns for line 4, 5 and 6 are exactly the same, The VSYNC pattern search does not need to correlate at one time for all the three lines of VSYNC signal pattern, instead, partial correlation can be performed separately for each line, a summation of the partial correlation results for the current line and the previous 2 lines will give the final correlation result for 3 continuous lines. The following equation shows how the partial VSYNC pattern search can be performed

$$R_{rV}[i] = -\sum_{k=0}^{N_L-1} r_{in}[k] + \sum_{k=0}^{N_H-1} r_{in}[k + N_L] - \sum_{k=0}^{N_L-1} r_{in}[k + N_L + N_H] + \sum_{k=0}^{N_H-1} r_{in}[k + 2 \times N_L + N_H]$$

Where N_L and N_H are the number of samples for the VSYNC low and high pulses in line 4, 5 and 6.

The VSYNC search is performed for duration of one video frame, the maximum matched result can be found as follows:

$$\max\left[\sum_{i=0}^{2} R_{rV}[i]\right]$$

The line count that runs from 0 to 524 can be set accordingly to achieve vertical (frame) synchronization. Once VSYNC is achieved, the receiver can keep updating the line count, since HSYNC is tracked continuously by the DLL, vertical synchronization will be kept automatically. The receiver will stay in this state until the line count reaches 0 indicating that a new video frame is starting, at this point, the receiver can transition to the steady state to start video frame processing.

Once vertical sync is detected, the DSP will also measure the VSYNC high and low pulse levels, the measured levels are used to adjust the r_{in} composite signal to be at the right IRE levels as shown in Figure 3. This adjustment is necessary since the video composite signal from the A/D is normally AC coupled, but the composite video signal includes also a DC component. The VSYNC and HSYNC pulse levels, blanking levels and reference white levels all need to be adjusted to the correct IRE levels to properly recover the RGB signals. Assuming that the measured low pulse level for VSYNC is V_L , the measured blanking level for VSYNC is B_L , the DC level and scaling factor are calculated as follows:

$$g = \frac{40}{B_L - V_L}$$
$$DC = -g \times B_L$$

The video composite input from the A/D is adjusted by g and DC as below:

$$g \times r_{in} + DC$$

After the DC level and gain adjustment, the video composite signal samples will be at the proper IRE levels for further video signal processing as shown in Figure 3.

Receiver State 3 (VIDEO) is the receiver steady state. It continues HSYNC / DLL tracking to keep synchronization with the transmitter in the presence of frequency offset. Video data extraction and processing for reconstruction of RGB signals are performed line-by-line and frame-by-frame. There are 484 displayable video lines for each video frame.

Figure 5 shows the signal processing blocks for video processing per line. For each displayable video line, the receiver needs to first extract the color burst to reconstruct the I/Q demodulation phase and chroma demodulation frequency. Since the receiver keeps line synchronization by the HSYNC DLL tracking loop, one can safely assume the demodulation frequency to be the nominal chroma carrier frequency.



Two novel algorithms for extracting the chroma phase are presented. The first algorithm is based on extracting the chroma phase in time domain while the second algorithm is based on extracting the chroma phase in frequency domain. First, in order to facilitate the estimation of the chroma phase, assume that the A/D sampling rate F_s is set to be four times of the chroma carrier frequency: $F_s=4*f_{sc}$, or 14.318MHz, there should be 4 samples per each color burst cycle, the phase change between the adjacent color burst samples is exactly 90 degree. The time domain algorithm simply takes any one color burst sample x(k) as sin(k), and x(k+1) as cos(k). The receiver can calculate the chroma phase at sampling position k by the following equation:

$$\phi[k] = \tan^{-1}(\sin(k)/\cos(k)) = \tan^{-1}(x(k)/x(k+1))$$

Although the time domain algorithm is extremely simple to implement, the drawback is that the chroma phase estimated directly from the time domain samples will be sensitive to the noise disturbance in the color burst.

The second algorithm extracts chroma phase in frequency domain. First, we extract N=24 time domain color burst samples, a Fourier Series is generated for frequency domain index k=N/4=6 as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, k = 6$$

The chroma phase can then be estimated from the Fourier Series:

$$\phi[k] = \tan^{-1}(imag(X(6))/real(X(6)))$$

Simulation results show that the chroma phase estimated from frequency domain is much more robust to noise disturbance than the chroma phase estimated from the time domain due to the averaging (smoothing) effect of the Fourier Series.

The I/Q demodulation phase for the current video line is then set to the following [1]:

$$\phi_D(k) = \phi(k) + 180^{\circ} + 33^{\circ}$$

A sin/cos table can be pre-calculated with demodulation phase starting from this initial phase and with 90-degree phase increment. Once the chroma phase is recovered, the rest of the video line processing will become quite straightforward: Y/C separation, I/Q demodulation and RGB reconstruction.

4. CONCLUTIONS

We presented an all software defined NTSC analog TV receiver, implemented in DSP, on the Sandbridge Technologies Software Defined Radio (SDR) platform. The total computing requirement for executing the entire analog TV protocol and displaying one quarter VGA is less than 1200 million instructions per second, which represents less than one half of the SB3010 DSP performance. The Analog TV receiver can coexist with other communication protocols on the same mobile platform.

5. REFERENCES

[1] Recommendation ITU - R BT.470-4 "Television Systems"

[2] Online information site: <u>http://www.ntsc-tv.com/</u>

[3] J. Glossner, D. Iancu, G. Nacer, S. Stanley, E. Hokenek, and M. Moudgill, "Multiple Communication Protocols for Software Defined Radio", *IEE Colloquium on DSP Enable Radio*, 227-236, September 22-23, 2003, ISIL, Livingston, Scotland.

[4] J. Glossner, D. Iancu, J. Lu, E. Hokenek, and M. Moudgill, "A Software Defined Communications Baseband Design", *IEEE Communications Magazine*, vol. 41, no. 1, pp. 120-128, January 2003.