EE318 Electronic Design Lab Project Report, EE Dept, IIT Bombay, April 2006

Wi-Fi Controlled LCD Projector/Monitor

Group: D1
Ankit Sethi (03D07004) ankits@ee.iitb.ac.in
Nikhil Rai (03D07003) nikhilr@ee.iitb.ac.in
Vinit Gawande (03D07025)vinitg@ee.iitb.ac.in

Supervisor: Prof Abhay Karandikar, Prof U.B.Desai

Abstract

Whenever we need to give a presentation using an LCD projector what we basically need to do is connect a VGA cable between the VGA port of a computer and the projector. The projector displays whatever appears on the monitor of the PC on the projector screen. The length of the cable connecting the PC and the projector cannot be increased to some extent because it is not convenient to handle such a long cable and also because of its cost. Typically in conferences when many people want to show their presentations using the same projector each of them have to connect this cable to their laptops and then continue with their presentations. It becomes difficult to carry a cable from the projector to the place where the user is having his PC and distance becomes a major constraint in this case.

This project aims at removing the cable between the PC and the projector making the interface wireless using a Wi-Fi protocol.

1. Introduction

An LCD projector basically takes analog signals from the VGA port of the PC and projects it on the screen. And whatever comes to the VGA port is the analog form of the data stored in the computers video memory which is in digital format. By capturing the data on the VGA RAM of the PC and then building a hardware which can convert this data into a VGA compatible format we can display the content on the video memory through a projector connected to this hardware.

What we are planning to do is capture the PC's video memory data and transmit it wirelessly to this hardware and then get a display of the PC's monitor screen. In this we are displaying the PC's monitor wirelessly. We are doing the wireless transmission using WiFi protocol.

Wi-Fi (Wireless Fidelity) is referred in relation to 802.11 networking protocol. We will be working on 802.11b protocol which works on 2.4 GHz band and gives a maximum transmission of 11 Mbps.

2. Approach Taken

2.1. Making a transmitter/Digitizer:

We started by trying to make a hardware that can take the analog VGA signals as input, digitize them and convert them into some serial format that can be transmitted over a USB port. We studied how VGA signals are generated and their voltage levels. We also studied the graphic digitizer IC of TI THS8083A95PZP. But later we learnt that the frequency of VGA signals is very high and it is difficult to get IC's with such a high throughput and also not feasible to work on them. So we changed our approach and now tried to take data from the USB port (digital) of the PC and then convert it into analog using a DAC. In this way we could avoid the problem of high throughput.

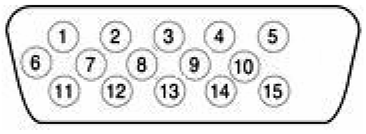

Figure1: VGA Pin out

2.2. Set up of Wireless Network using USB Adapters

The Wi-Fi connection was set up between 2 PC's using the X–Micro WLAN 11b mini USB adapters. The driver was installed on Windows and then required configuration was done. Following are the details of Transmitter and Receiver PC's and wireless network.

Table 1: Configuration of Transmitter PC

IP Address	10.10.10.2
Subnet Mask	255.255.255.0
Default Gateway	10.10.10.10
MAC Address	00-02-72-46-B9-8D

Table 2: Configuration of Receiver PC

IP Address	10.10.10.1
Subnet Mask	255.255.255.0
Default Gateway	10.10.10.10
MAC Address	00 -02-72-46-B9-F8

The wireless configuration was also done to establish the peer to peer connection. Following are the details of the wireless network connection.

Table 3**:** Wireless Network Configuration

Network Mode	Peer to peer(Ad Hoc)
Channel Used	4 (2427Mhz)
SSID	LOCAL
Tx Rate	11 mbps
Antenna	Internal
WEP	Disabled
BSSID	6A-C4-00-2D97-23
Data Encryption	Disabled

We transferred files using the smart ftp client from transmitter PC (10.10.10.1) to receiver PC (10.10.10.2). It was found that we could get a maximum transfer rate of 341.6 kb/sec with small fluctuations in the data transfer rate through the transfer period. The size of the file transferred was 40.7Mb (File name: ‘Top 10 goals’). The transfer rate vs. time graph is shown in Fig 3.

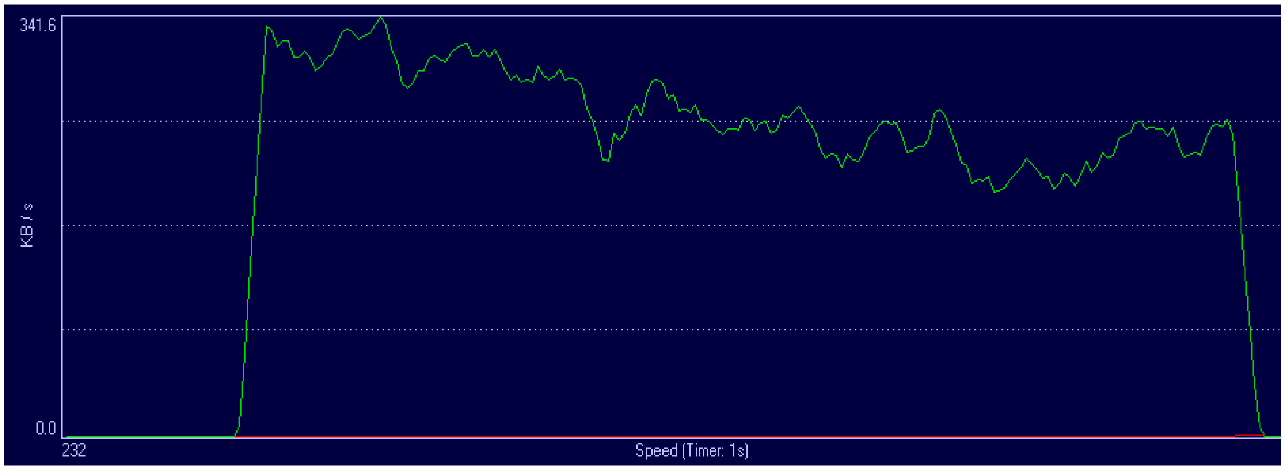

Figure 2: Transfer rate vs. Time

2.3. Approach at Receiver End

Our next step involved capturing the digital data stored in the video RAM and then transmitting it through the Wi-Fi adapter. At the receiver end we planned to use a USB interfaced microcontroller in which we would install the driver of the adapter. As it is very difficult to get the source code in windows therefore we tried to install the driver in Linux. With this we could cross compile the driver code into the microcontroller at receiver end.

We installed the latest 2.6.15 kernel which had the driver zd1202 tree for the wlan on our fc4 OS (which had 2.6.11 kernel). The driver also required the firmware compatible to our X-Micro WLAN 11b USB Adapter which would be loaded by the hotplug utility in Linux. The modules for which were also loaded. 'modprobe zd1201' loaded the driver built as a MODULE in the firmware.

Firmware was put in a directory /lib/firmware. 'dmesg' showed that everything was fine in /var/log/messages but the installation didn’t work. Though the hotplug agent was working fine but still the firmware was not being loaded. This was the main problem which we faced and got stuck. If it would have went right then we would have seen wlan0 in '/sbin/ifconfig –a

The block diagram is shown in Fig. 3

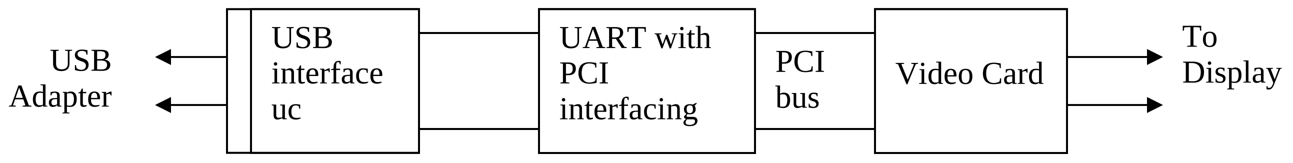

Figure 3: Block Diagram at receiver end

2.4. Software Approach:

After successfully setting up a wireless network using the Wi-Fi adapters we first decided to find a software solution to capture image of the transmitting PC and display it on the monitor of the receiving PC.

We went through following three approaches:

a. Socket Programming: We wrote a socket program through which data can be sent to the USB port of the PC which the adapter can transmit.

b. Batch Programming: We also tried batch programming in DOS. Using this we were able to open any file on the receiving PC but with the help of respective applications like notepad, Acrobat reader etc. This approach didn't prove to be useful because ultimately everything has to be done in hardware and thus making an application available in software doesn't serve our purpose.

c. VNC: VNC (Virtual Network computing) is open source software distributed as Server and Viewer. The Server should be installed on the PC whose screen image has to be projected and the viewer has to be installed on the receiving PC.

2.5 VNC (Virtual Network Computing):

2.5a. The VNC Protocol

The VNC protocol is a simple protocol for remote access to graphical user interfaces. It is based on the concept of a *remote frame buffer* or *RFB*. The VNC protocol is also referred as the RFB protocol. The protocol simply allows a server to update the frame buffer displayed on a viewer. Because it works at the frame buffer level it is potentially applicable to all operating systems, windowing systems and applications. The protocol will operate over any reliable transport such as TCP/IP.

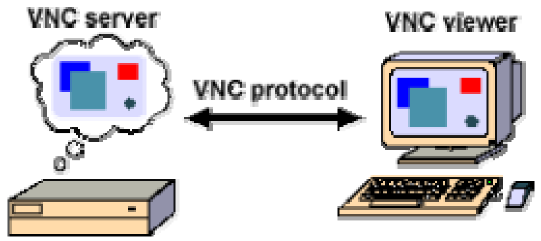

Figure 4: VNC network

2.5b. Rectangular updates

The display side of the protocol is based around a single graphics primitive: *"put a rectangle of pixel data at a given x, y position"*. Using different *encoding schemes* for the pixel data, we can select the appropriate scheme for each rectangle we send, and make the most of network bandwidth, client drawing speed and server processing speed.

The lowest common denominator is the so-called *raw* encoding, where the rectangle is simply pixel data sent in left-to-right scanline order.

The *copy rectangle* encoding, for example, is very simple and efficient and can be used when the client already has the same pixel data elsewhere in its framebuffer. The server simply sends an X, Y coordinate giving the position from which the client can copy the rectangle of pixel data. This means that operations such as dragging or scrolling a window, which involve substantial changes to the screen, may only require a few bytes. This encoding thus saves bandwidth.

A workstation desktop has large areas of solid colour and of text. Effective encodings take advantage of this by efficiently describing rectangles consisting of one majority (background) colour and 'sub-rectangles' of different colours. We might use a JPEG encoding for still images or MPEG for efficient transmission of moving images. An encoding which uses some kind of caching of pixel data would be good for rendering text, where the same character is drawn in the same font multiple times. Subsequent occurrences of the same character would be encoded simply by reference to the first occurrence.

2.5c. Adaptive update protocol

A sequence of these rectangles makes a *framebuffer update* (or simply *update*). An update represents a change from one valid framebuffer state to another, so in some ways is similar to a frame of video, but it is usually only a small area of the framebuffer that will be affected by a given update.

The update protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client. This gives the protocol an adaptive quality. The slower the client and the network are, the lower the rate of updates becomes. Each update incorporates all the changes to the 'screen' since the last client request. With a slow client and/or network, transient states of the framebuffer are ignored, resulting in reduced network traffic and less drawing for the client. This also improves the apparent response speed.

2.5d. Input protocol

The input side of the protocol is based on a standard workstation model of a keyboard and multi-button pointing device. Input events are sent to the server by the client whenever the user presses a key or pointer button, or whenever the pointing device is moved.

2.5e. Connection Setup and Shutdown

When the connection between a client and a server is first established, the server begins by requesting authentication from the client using a challenge-response scheme, which typically results in the user being prompted for a password at the client end. The server and client then exchange messages to negotiate desktop size, pixel format, and the encoding schemes to be used. The client then requests an update for the entire screen, and the session begins.

2.5f. VNC Clients

Since VNC viewer is a thin-client system. Writing a client requires only a reliable transport (usually TCP/IP), and a way of displaying pixels (either directly writing to the framebuffer, or going through a windowing system).

2.5g. VNC Servers

The protocol is designed to make the client as simple as possible, so it is usually up to the server to perform any necessary translations. For example, the server must provide pixel data in the format the client wants. Each VNC desktop is like a virtual X display, with a root window on which X applications can be displayed. Servers simply mirror the real display to a remote client.

3. Proposed Hardware design

In this design we propose TMS320DM642 Video/Imaging Fixed point digital signal processor. It is a 32 bit processor with following desired features:

1. 128 KB L1 program cache
2. 128 KB L1D Data cache
3. 2M-Bit (256K-Byte) L2 Unified Mapped Ram/Cache (Flexible Ram cache Allocation)
4. 10/100 Mb/s Ethernet MAC (EMAC)
5. Three configurable Video port providing a Glueless I/F to common Decoder and Encoder Devices

This processor can communicate with the Wireless Access Pont (WAP) through Ethernet Interface. The data coming from the transmitter P.C., that is the data transmitted by the VNC server software can be received through the WAP. In the processor the equivalent client software of VNC should be written which would produce the required video signals on any of its video port as were produced by the client VNC software program in P.C. to P.C. transmission. This video signals can be interfaced with the VGA port through an interfacing circuit (described in section 3). Also the Ethernet port needs to be interfaced with this processor (described in section 2). The processor works at high speed (clock frequency 500, 660 and 720MHz) therefore extra care needs to be taken to remove noise signals from the external circuit so that it doesn’t interfere with the processor’s performance.

3.1Power and Clock Circuitry

For generating high frequency clock for the processor we may use PLL ICS512.

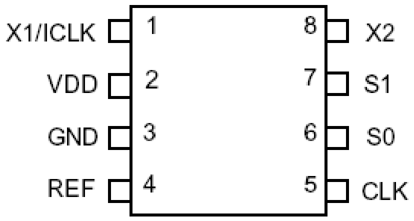

Figure 5: Pin Diagram of ICS512

60 MHz is its input crystal frequency. Relevant circuitry is shown in Figure 7.

For DSP operations a separate clock needs to be given to the processor (DSP CLK) involving the same circuitry as for normal clock but with crystal frequency of 25MHz (Figure 7).

A separate supply needs to be given to VDD PLL of the processor as shown in Figure 8.

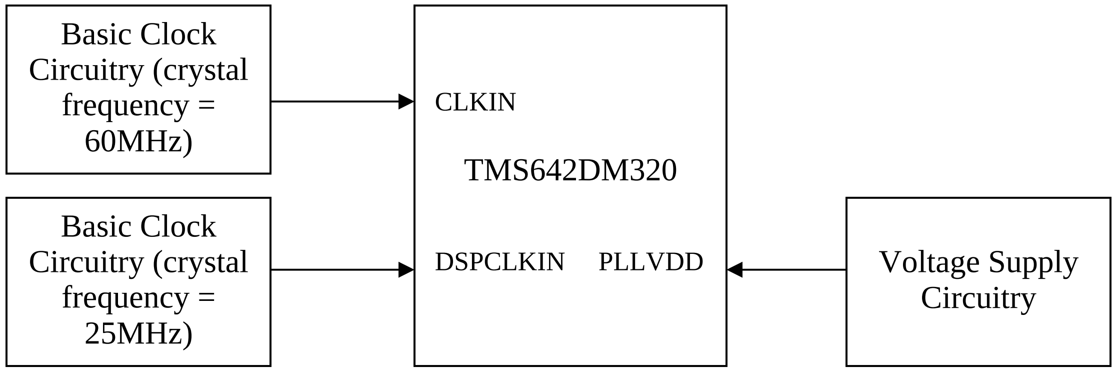
Basic Clock Circuitry (crystal frequency = 60MHz)
Basic Clock Circuitry (crystal frequency = 25MHz)
CLKIN
TMS642DM320
DSPCLKIN
PLLVDD
Voltage Supply Circuitry

Figure 6: Block Diagram of the Clock and Voltage Circuitry

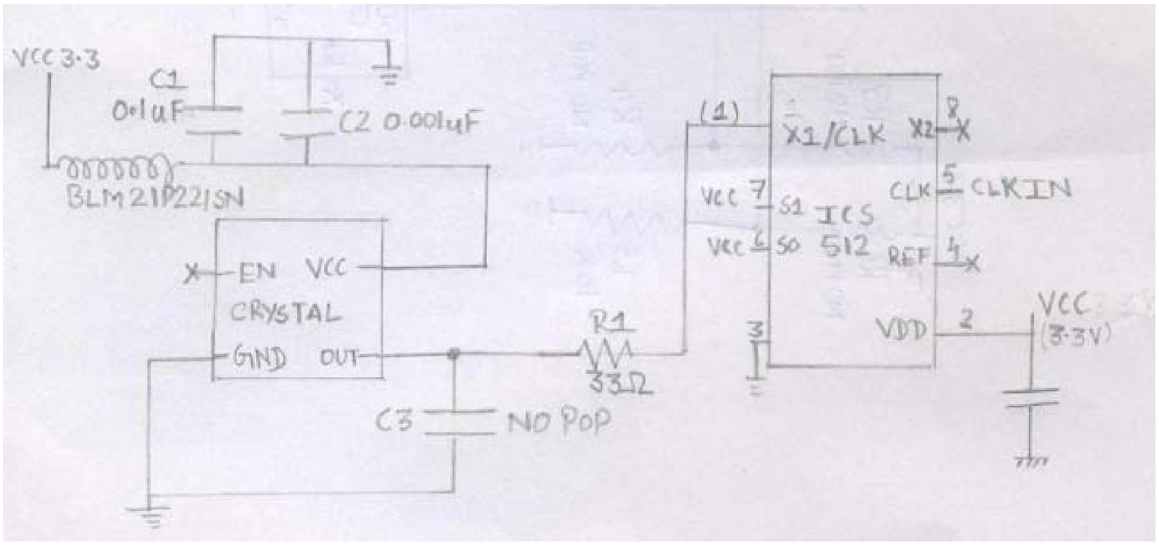

Figure 7: Circuit Diagram showing the relevant clock circuitry

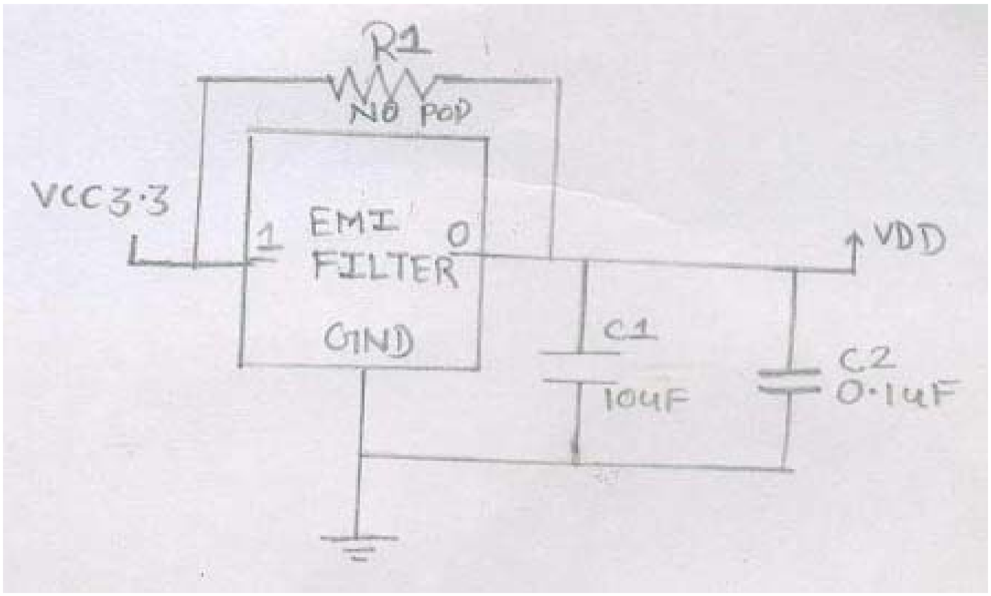

Figure 8: Circuit Diagram showing the generation of VDD signal (fed to VDD PLL)

3.2Interfacing TMS640DM642 and Ethernet port using LXT971ALC

The EMAC of TMS320DM642 has to be interfaced with the signals in the physical layer i.e. signals coming on the Ethernet port. This may be done by using LXT971ALC, which is an IEEE complaint Fast Ethernet PHY Transceiver that directly supports both 10Base-TX and 100BASE-T applications. It provides Media Independent Interface (MII) for easy attachment to 10/100 Media Access Controllers (MACs).

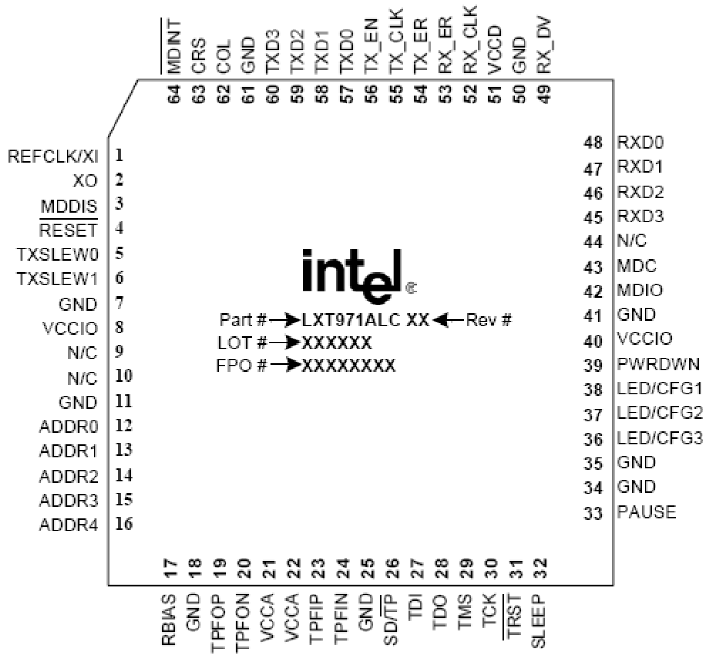

Figure 9: Pin diagram of LXT971ALC

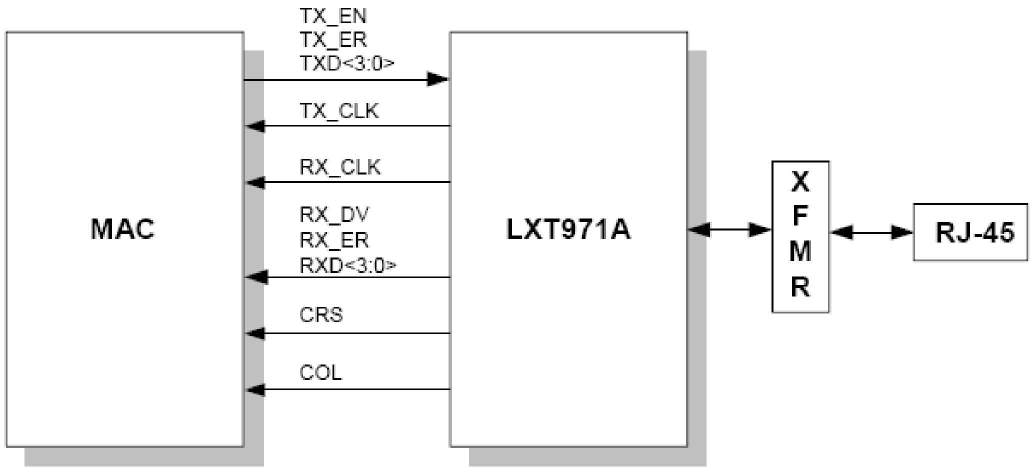

Figure 10: Block Diagram of MII Interface

A block diagram of the interfacing circuit is shown above. The MAC and LXT971ALC are to communicate with each other in half duplex mode. The signals of the Ethernet port are to be directly given to LXT971A at its respective receiver (RX) pins. The output (TX) at transmitter pins are to be given to the EMAC interfacing pins of the processor.

For our purpose pins corresponding to TXSLEW0, TXSLEW1 are pulled high with 10k resistors. PAUSE and SLEEP are grounded. At the input port pins the corresponding Ethernet port pins should be connected. The output port of LXT971ALC should be connected to the EMAC interface of TMS320DM642

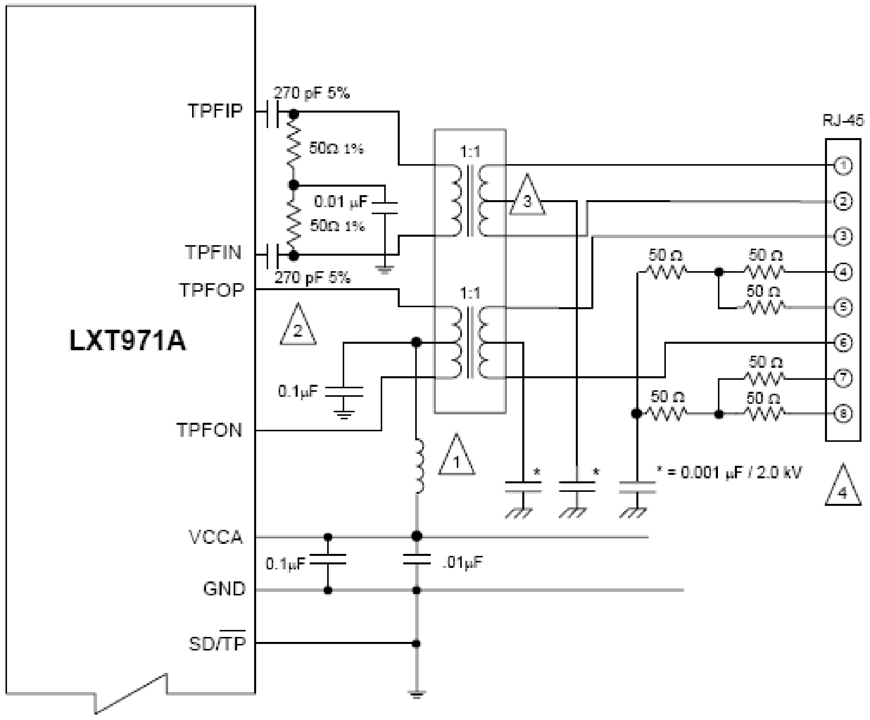

Figure 11: Interfacing LXT971A with twisted pair network

3.3 Interfacing TMS320DM642 and VGA port with SAA7104H

The signals coming from the video port of TMS320DM642 cannot be given directly to VGA port. These signals need to be converted to a set of RGB signals and with other synchronization signals that can be given to VGA port. To interface it with VGA port SAA7104H (Digital Video Encoder Chip) may be used. It supports VGA resolution up to 1280x1024 graphics data on monitor at refresh rate of 50Hz or 60Hz.

For our purpose

- Pins VSSD1 to VSSD4 (7,13,26,57) must be grounded (13,36,57 analog ground and 7 digital ground)
- Power supply of +3.3Volts to VDD2-VDD4 (de-noising it through capacitor of 0.1uF)
- Crystal of frequency 27 MHz is to be used with coupling capacitance of 10pf and 1nF. Also coupling inductance of 0.1uH should be used.
- VDD1 is to be connected to DVO supply (de-noise it).

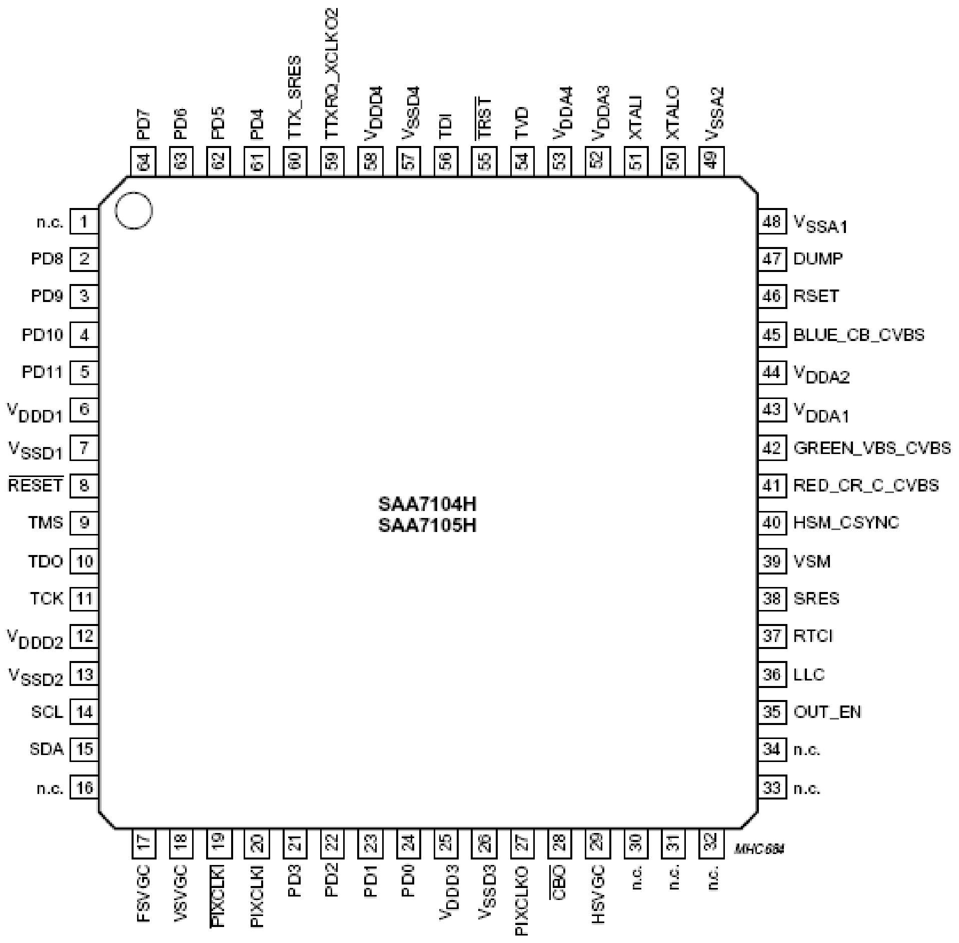

Figure 12: Pin Diagram of SAA7104

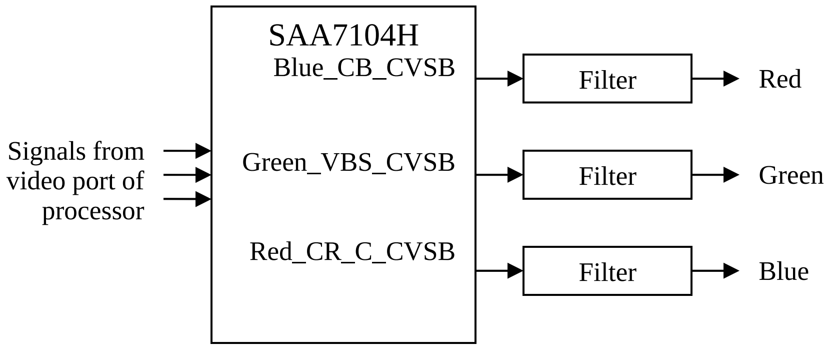

Figure 13: Block Diagram of working of SAA7105

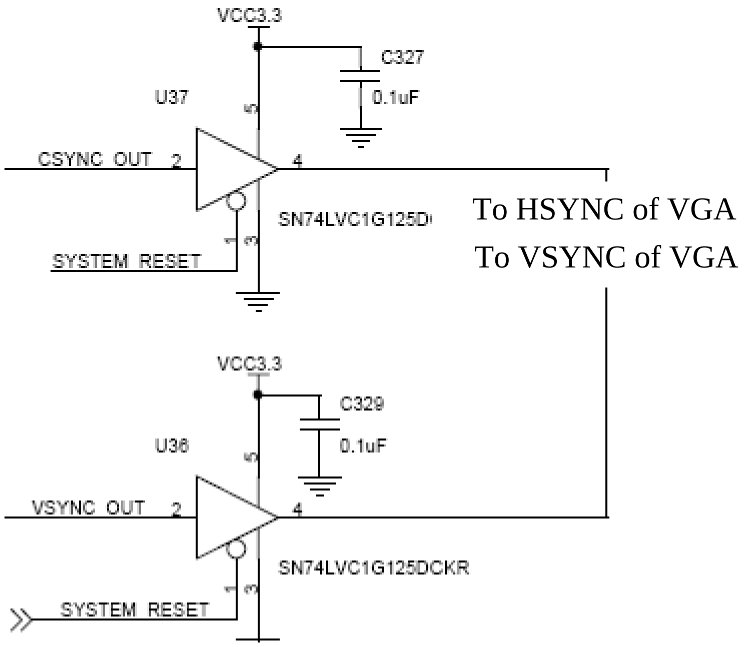

Figure 14: Block Diagram showing how HSYNC and VSYNC signals are produced

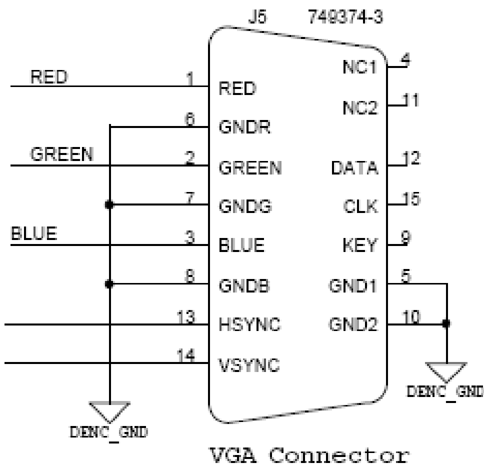

Figure 15: Pin Diagram of VGA connector

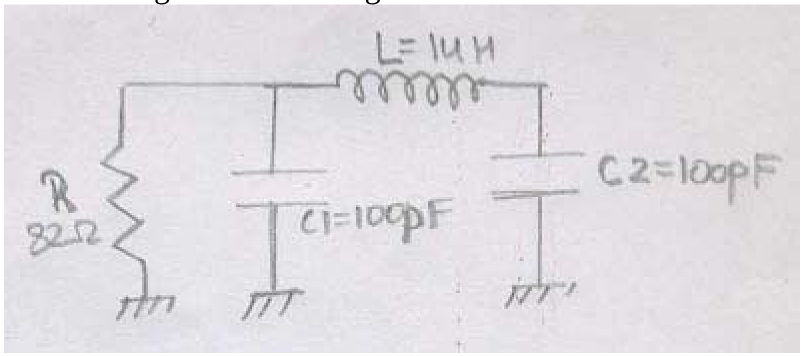

Figure 16: Circuit Diagram of Filter to be attached in Figure 14.

4. Current Status:

We have been successful in configuring the USB Wi-Fi adapters and have set up a wireless network. The adapters have been configured for an Ad-hoc (Peer-to-Peer) network. The adapters communicate only when the SSID of the transmitter and the receiver are the same so we gave a common SSID "LOCAL" to both of them. The transmitting PC is configured with the IP 10.10.10.1 and the receiving PC is configured with the IP 10.10.10.2 with the common gateway 10.10.10.10 and netmask 255.255.0.0. We were also able to transfer files from one PC to another using FTP clients through this network. The network looks as shown in figure (2) below:

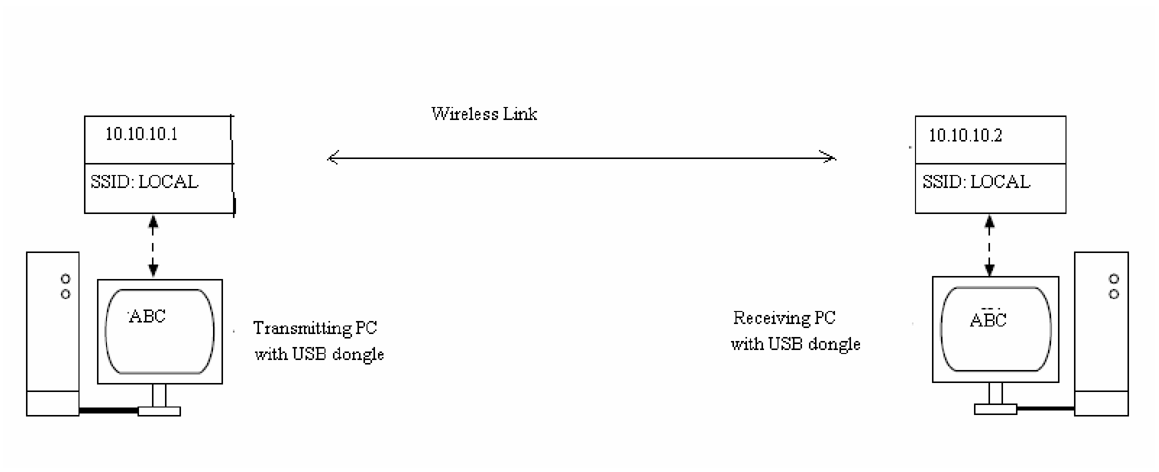

Figure 4: Wireless network

We have installed the VNC server on the Transmitting PC and have an equivalent client on the receiving PC. Now when we run the VNC server on the transmitting PC we can communicate to it from the receiver using its IP 10.10.10.2. When the IP is entered on the receiving side we can export the display of the transmitter on the monitor of the receiving PC. The display is the real time image and can show all the motions on the transmitting screen.

5. Conclusion

1. Wi-Fi Adapters were successfully configured in peer-to-peer mode and a stable wireless network was established
2. Data transfer was achieved using this wireless network.
3. VNC server and client software were configured to be used in wireless network and successful real time image transfer from server to client was achieved.
4. A paper design of an interface with an Ethernet input and a VGA output was made.

References

[1] “Data sheet for SAA7104H (Digital Video Decoder Encoder) Koninklijke Philips Electronics N.V. http://www.semiconductors.philips.com/pip/SAA7104H.html, Koninklijke Philips Electronics N.V. Accessed in March 2006

[2] “Linux Driver For X-Micro WLAN 11b USB Adapter”, http://linuxlc100020.sourceforge.net ,Andre Eberra. Accessed in March 2006

[3] “Download TightVNC ,” http://prdownloads.sourceforge.net/vnc-tight/tightvnc-1.2.9_winsrc.zip?download , SourceForge. Accessed in March 2006

[4] “TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor,“ Texas Instrumentation, http://focus.ti.com/docs/prod/folders/print/tms320dm642.html Accessed in April 2006.