January 1995

AVATAR -- ATM VideoAudio Transmit and Receive

William S. Marcus
Bell Communications

C. Brendan S. Traw
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-95-12.

AVATAR -- ATM VideoAudio Transmit and Receive

Abstract

To facilitate the transport of audio and video data across emerging Asynchronous Transfer Mode (ATM) networks, a simple, low cost, audio/video ATM appliance, the AVATAR, has been developed. This appliance is capable of handling uncompressed bidirectional audio and NTSC video connections.

The intended applications for this device include TeleMentoring (a NSF sponsored exploration of distance mentoring), video conferencing, and network monitoring. Our experience has shown that AVATAR is an effective, low cost means of providing multimedia connectivity between sites within the Aurora Gigabit testbed.

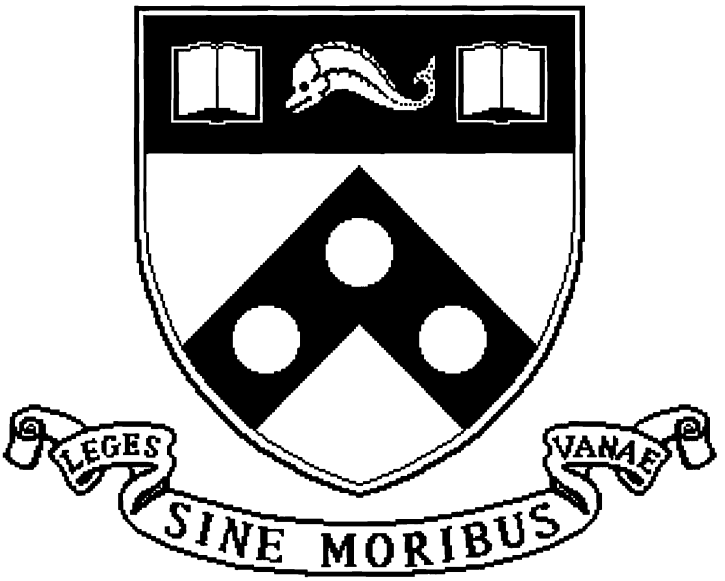
Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-95-12.

AVATAR
ATM VideoAudio Transmit and Receive

MS-CIS-95-12

W. S. Marcus
C. B. S. Traw



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

March 1995

AVATAR

ATM Video/Audio Transmit And Receive

W. S. Marcus* C. B. S. Traw†

October 24, 1994

Abstract

To facilitate the transport of audio and video data across emerging Asynchronous Transfer Mode (ATM) networks, a simple, low cost, audio/video ATM appliance, the AVATAR, has been developed. This appliance is capable of handling uncompressed bidirectional audio and NTSC video connections.

The intended applications for this device include TeleMentoring (a NSF sponsored exploration of distance mentoring), video conferencing, and network monitoring. Our experience has shown that AVATAR is an effective, low cost, means of providing multimedia connectivity between sites within the Aurora Gigabit testbed.

1 Introduction

The transport of audio and video data is one of the primary goals of the emerging gigabit per second asynchronous transfer mode[3] (ATM) networks. The Aurora gigabit testbed[5], connecting Bellcore, IBM, MIT, and UPenn has been instrumental in developing the technologies necessary to support multimedia and data networking at these bandwidths.

To support the transfer of uncompressed audio and color video across the Aurora testbed, an audio/video appliance has been developed at Bellcore and UPenn. This device digitizes and then packetizes uncompressed 64Kbps audio and 80 Mbps video streams into ATM cells. Separate network connections are used for each of the media so that they can be switched independently of

*Bell Communications Research

†University of Pennsylvania

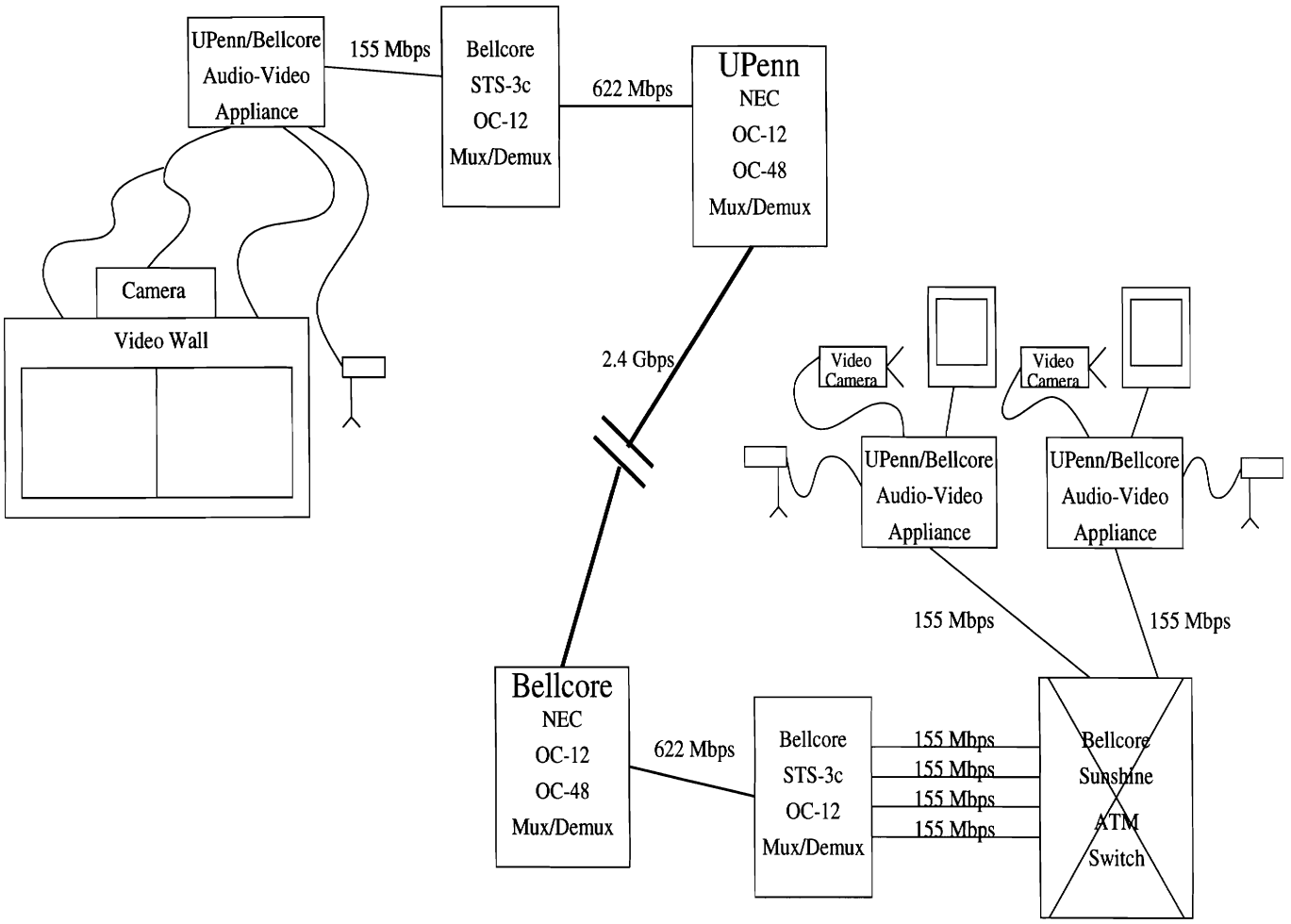


Figure 1: Network Configuration

each other. The appliance can also receive ATM cells making up these interleaved media streams, extract the data from the ATM cells, and then recreate the analog video and audio signals.

Figure 1 shows several audio/video appliance board in a possible Aurora network topology. Anticipated uses for the audio/video appliance in the type of network configurations include Telementoring[8], video conferencing, and network monitoring.

Other work within the Aurora testbed in the area of hardware support for multimedia applications includes Joel Adam's "Vidboard[1]" for MIT's VuNET desk area network and Sanjay Udani's NTSC video capture card[10] for the IBM MicroChannel bus. Of these two other efforts, the Vidboard is the most similar. Both the Vidboard and the audio/video appliance discussed in this paper are capable of sinking and sourcing audio and video data. The major differences focus on the complexity and capabilities of the designs. The Vidboard is significantly more complicated,

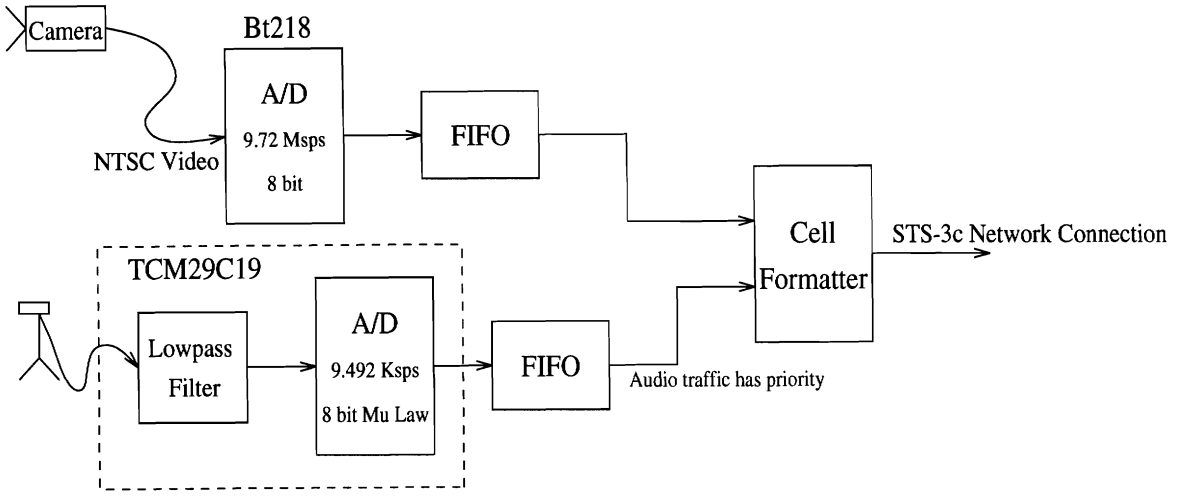


Figure 2: Audio/Video Transmitter

allowing extraction of video frames from the NTSC signal as well as the flexibility provided by a DSP in the datapath.

Devices with similar capabilities to this audio/video appliance have been developed by Cesar Johnston of Bellcore[4] and for use in AT&T's LuckyNet broadband networking testbed[7].

The remainder of this document explains the architecture of the audio/video appliance and its usage.

2 Architecture

The audio/video appliance can be separated into two major parts, the transmitter and the receiver.

2.1 Transmitter

The transmitter, as depicted in Figure 2, digitizes analog video and audio signals, encapsulates the data into ATM cells, and presents the cells to a STS-3c network connection. It is implemented using off-the-shelf technology: three Erasable Programmable Logic Devices (EPLD), two synchronous FIFOs, two A/D integrated circuits, and several linear components.

The digitization of the composite video stream is conducted via a flash A/D converter integrated within a single device, the Brooktree Bt218 at a rate of 9.72 megasamples per second. This rate is above the Nyquist rate of a NTSC video signal. In order to decouple the video source (i.e., the camera) from the Audio/Video ATM Appliance, the video input is AC coupled. This requires the A/D device to be periodically zeroed or "clamped" to force the AC coupled signal to a known

reference voltage. The clamping is performed during horizontal and vertical video synchronization intervals. In lieu of analog video frame detection of these intervals, we used the following scheme. At reset the analog input is properly biased. An EPLD monitors the digital output of the A/D device for values indicative of video blanking, an event that occurs only during synchronization. Once detected, the EPLD momentarily clamps the video input. The A/D employs raw linear coding to produce 8-bit values which are written into a decoupling FIFO. Once 48 bytes are available, the STS-3c network connection requests an ATM cell, and 48 bytes from the audio A/D are not available, a second EPLD removes 48 bytes from the FIFO, produces a hardwired ATM cell header (with VCI == 1), and presents the cell to the network.

The digitization of the audio signal is performed via a single IC device, the Texas Instrument TCM29C19. The device incorporates a front end anti-aliasing filter which requires only an external reference clock. The A/D employs μ-law coding to produce 8-bit values which are presented as serial output. An EPLD performs serial to byte conversion and writes the 8-bit values into a decoupling FIFO. Once 48 bytes are available and the STS-3c network connection requests an ATM cell, an EPLD removes 48 bytes from the FIFO, produces a hardwired ATM cell header (with VCI == 2), and presents the cell to the network.

When the Audio/Video Transmitter has access to the underlying global SONET physical transport layer its digital sampling clocks, for both the video and audio analog signal, are derived from the SONET STS-3c 19.44MHz clock. The Audio/Video Receiver uses the same global clock to derive its playback clocks. This clocking scheme assures no data loss due to clock mismatch; excessive jitter incurred in the ATM subnet is the only source of data loss. When the Audio/Video Appliance does not have access to the underlying global SONET physical transport layer, it uses an on board 19.44MHz oscillator to derive its sampling and playback clocks. In this mode, both network jitter and clock mismatch can cause data loss. Due to the redundancy of the "raw" video and audio digitized streams, our simple method of dealing with loss, which is described in the following section, proved more than adequate.

2.2 Receiver

The receiver, as shown in Figure 3, takes cells from interleaved audio and video connections and demultiplexes them. This operation is performed by looking at the value of the VCI in the incoming cell's header. The payload of cells belonging to the video connection (VCI == 1) are written into one FIFO while the payload of cells belonging to the audio connection (VCI == 2) are written to a separate FIFO. Cells belonging to other connections (VCI != 1 or 2) are discarded. The VCI values for audio and video data have been arbitrarily assigned. Any valid VCI value can be used for either connection, by altering the EPLDs which implement the functionality. The header cyclic redundancy check (CRC) for the ATM cell header is also ignored.

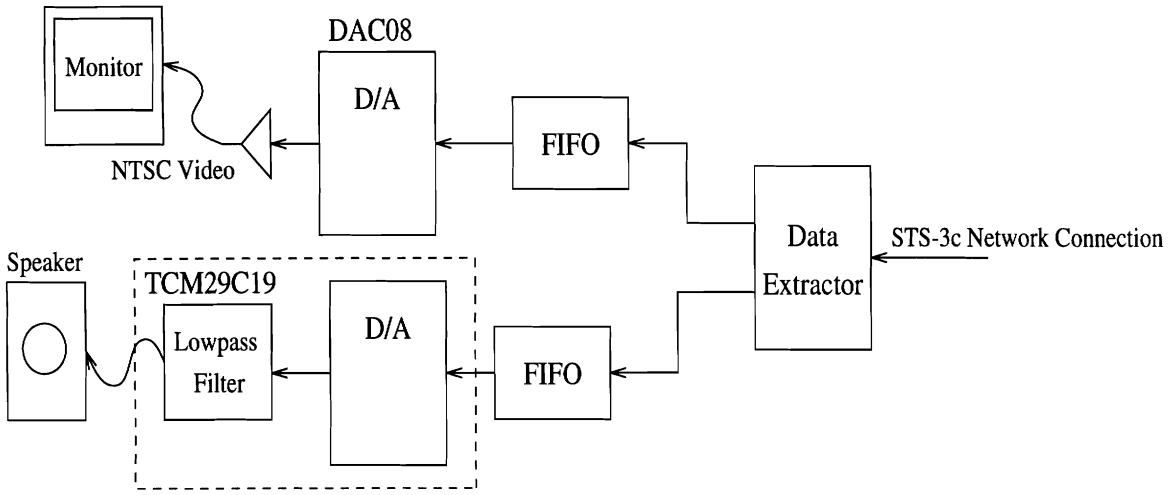


Figure 3: Audio/Video Receiver

The state of these FIFOs is monitored to maintain sufficient buffering to ensure reasonable audio and video quality. The ideal state of the receive FIFOs is half full. Being half full allows the greatest tolerance for cell jitter. If the buffer begins to over or under run, loss of data or the use of incorrect data is possible. If an over run condition is detected, no additional data is written to the FIFO until it has been drained down to the half full level. This results in the discarding of some cells received from the network. An under run condition pauses the reading of the FIFO until it has been refilled to the half full condition. Either of these corrective actions results in a momentary loss of video or audio, while the error condition is being corrected.

Audio and Video data are read from their respective FIFOs. Video data is read at 9.72 MBps (Network clock divided by 2) and converted back to a analog signal by an Analog Devices DAC008 digital to analog converter. Since the DAC008 produces a variable current as output, a resistor to ground is used to convert the current to a voltage. The value of the resistor was selected such that the full scale output of the DAC would generate a one volt signal. The resulting output voltage is buffered by a unity gain opamp with a high impedence input and capable of driving low impedence video loads.

Audio data is read from its FIFO at 9.492 KBps (Network clock divided by 2048). The byte wide data is converted to a serial bit stream shifted at 2.43 MHz (Network clock divided by 8) as required by the input of the decoder section of the TI audio CODEC. The decoder converts the μ-law encoded digital data back into an analog form and then low pass filters it to remove reconstruction artifacts before it is presented on the output pin.

2.3 Design Decisions

No data compression was utilized for several reasons. First, the inclusion of compression for the audio data would have an insignificant effect on the overall bandwidth required for a video conference since the bandwidth required by the audio channels is several orders of magnitude lower than that required for the video. Video compression would significantly reduce the bandwidth requirements, but it would require significant effort to include the necessary hardware and/or software resources needed to perform it at full NTSC quality and frame rate. Compression would also defeat one of the systems oriented goals of this appliance which is to provide a heavy application load on the network infrastructure. Finally, the lack of compression has made the video signal very tolerant of data loss and corruption, as expected since the video signal format is intended for the lossy broadcast environment.

Another controversial design decision was to digitize the raw NTSC signal instead of extracting out the video frames and just transmitting them. We had two major reasons for this design decision. The first is that the extraction of the video data from the analog signal and then recreating that signal at the remote point would have increased the complexity of the implementation. The other reason is that the analog NTSC signal contains all synchronization information and thus no synchronization support is needed by the appliance.

3 Operation

Figure 4 show the major connectors and controls for the audio/video appliance.

3.1 Analog Interface

The analog video interface consists of two BNC connectors, an input and an output. The video input is expected to be one volt peak to peak and conform to the standard NTSC video format. Although video quality degrades with cable length, video interconnection cables as long as 75 feet have been used and provide reasonable video quality.

Audio connection is provided via two quarter inch mono phone jacks. The audio signal may be up to 4 volts peak to peak. Acceptable cable lengths for audio connections vary greatly depending on the type of cable and what is driving it.

Audio or video Loopback operation may be selected for testing of boards or cabling by setting the first two DIP switches as shown in Table 1 to the proper position. Video loopback is accomplished by taking the byte wide output of the Bt218 and feeding it directly into the byte wide data input of the DAC008. Audio loopback is performed entirely within the audio CODEC chip.

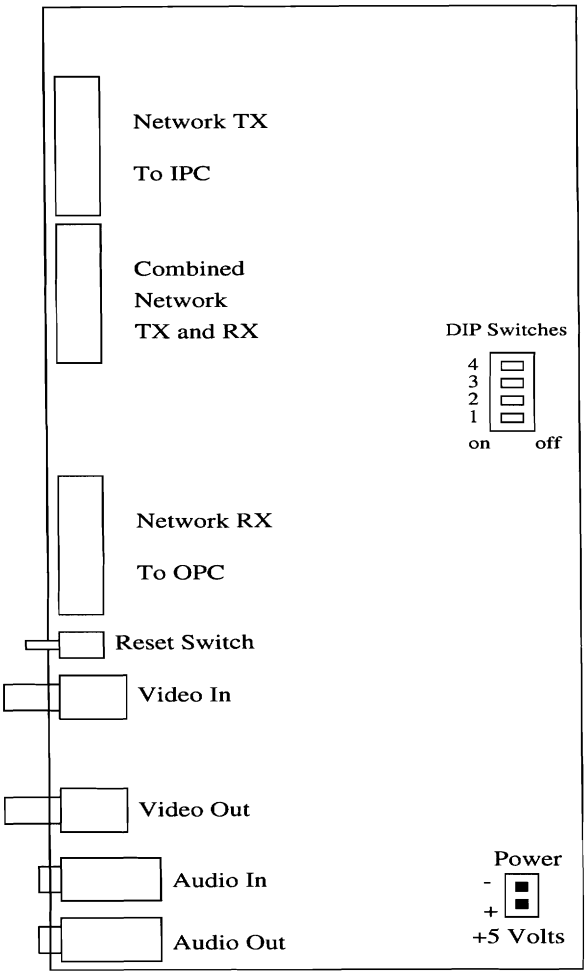


Figure 4: Connector Locations

While video or audio loopback are selected, video and audio cells will still be generated, but any audio or video cells received by the appliance will be ignored.

3.2 Network Interfaces

The remaining two DIP switches select the mode of the network interface and determines which of the 34 pin ribbon cable headers will be used to attach to the network,

The audio/video appliance is designed to attach to the network in two different modes selectable by a fourth DIP switch. When this switch is off, the appliance is a network slave device, meaning that it receives its framing and clock from the network. The mode is typically used when the appliance is attached to a physical layer interface such as a SONET multiplexer or GLINK board[9].

Switch	On function	Off function
4	Audio Loopback off	Audio Loopback on
3	Video Loopback off	Video Loopback on
2	Combined RX and TX network connectors	Separate RX and TX network connectors
1	Network master	Network slave

Table 1: DIP Switch Functions

When this switch is on, the appliance generates its own clock and framing signals. This mode is typically used when the device is attached directly to the Sunshine ATM switch [6].

The third switch selects which set of connectors are used to attach to the network. The middle 34 pin ribbon cable connector is selected by placing the third switch in the ON position. This connector has both the transmit and receive directions supported. It is typically used for connecting to physical layers such as SONET or GLINK. When this switch is off, the outer two 34 pin ribbon cable connectors are selected. The end connector has the transmit direction while the remaining connector supports the receive direction. These two connectors are typically used when connecting directly to the Sunshine ATM switch. The transmit connector goes to the switch's input port controller (IPC) while the receiver connector goes to the output port controller (OPC). The transmit and receive directions are given separate connectors to reduce cross talk between the conductors by separating each signal conductor from its neighbors by a grounded conductor. The length of the connecting cables for these connectors is limited to about 14 inches. Seperate ribbon cables are required for the TX and RX data due to the design of the Sunshine switch rack.

3.3 Other Connections and Controls

The power connector is also shown in the figure. The board requires +5 volts at about 1.5 Amps. Users should be careful when connecting the power supply as the connector is not polarized.

The final control shown in Figure 4 is the reset switch. This switch, when pressed, resets all of the digital logic including the FIFO buffers on the board.

Finally, there are several potentiometers and jumpers on the board which are not shown in Figure 4. For their functions, see the schematics in Appendix B.

4 Current Status

Two versions of this appliance have been produced. An initial, video only, version has fabricated late in February 1994. This version was placed into use for Telementoring early in March 1994. The second version which supports both audio and video was fabricated and operational in April 1994. Boards from this second run have been installed at Bellcore, MIT, and UPenn and have been used extensively.

5 Known Bugs

The termination for the offboard/onboard oscillator clock is done incorrectly. There should have been a voltage divider but instead there is only R20 to ground. Unless an additional resistor is added to VCC, R20 should be unpopulated.

The silk screen for D1 is incorrect. The polarity of the device should be reversed.

Acknowledgments

The authors would like to thank Drew Moore and Sanjay Udani for their assistance and Jonathan Smith for encouraging this work.

AURORA is a joint research effort undertaken by Bell Atlantic, Bellcore, IBM Research, MIT, MCI, NYNEX, U. Arizona, and U. Penn. AURORA is sponsored as part of the NSF/DARPA Sponsored Gigabit Testbed Initiative through the Corporation for National Research Initiatives, under Cooperative Agreement number NCR-8919038. NSF and DARPA provide funds to the University participants in AURORA. Bellcore has provided support to UPenn through the DAWN project. Telementoring is supported at the University of Pennsylvania by the NSF under Agreement number CDA-92-14924. Hewlett Packard has supported this effort through the donation of laboratory test equipment to UPenn.

References

[1] Joel Adam and David Tennenhouse, "The Vidboard: A Video Capture and Processing Peripheral for a Distributed Multimedia System," ACM Multimedia, June, 1993.

[2] Altera Corporation, 1992 Data Book.

[3] CCITT Recommendation I.361, *ATM Layer Specification for B-ISDN*, 1990.

[4] H. J. Chao, C. A. Johnston, and L. S. Smoot, "A Packet Video/Audio System Using the Asynchronous Transfer Mode Technique," *IEEE Transactions on Consumer Electronics*, pp. 97-105, May, 1989.

[5] D.D. Clark, B.S. Davie, D.J. Farber, I.S. Gopal, B.K. Kadaba, W.D. Sincoskie, J.M. Smith, and D.L. Tennenhouse, "An Overview of the AURORA Gigabit Testbed," *Proceedings of the 1992 IEEE Infocom Conference*, Florence, Italy, 1992.

[6] J. Giacopelli, J. Hickey, W. Marcus, W. D. Sincoskie, and M. Littlewood, "Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture," *IEEE Journal on Selected Areas in Communications* **9**(8), pp. 1289-1298 (October, 1991).

[7] R. D. Gitlin and T. B. London, "Broadband Gigabit Research and the LuckyNet Testbed," Journal of High Speed Networks, pp. 1-47, 1992.

[8] J. M. Smith, *et al.*, "Experiences with TeleMentoring: Lab Teaching over Gigabit Networks," CIS Department, University of Pennsylvania, unpublished.

[9] Christopher J. Russo, Andrew M. Moore, C. Brendan S. Traw, and Jonathan M. Smith, "Early Experiences with ATM Over Hewlett-Packard HDMP-1000," CIS Department, University of Pennsylvania, unpublished.

[10] Sanjay Udani, "Experimental Evaluation of a Video Capture Board for Networked Workstations," Technical Report MS-CIS-93-31, CIS Department, University of Pennsylvania, 1993.

Appendix A: Altera EPM5032 AHDL Code

```
% %
% Cheap video receiver %
% %
% C. Brendan S. Traw %
% 115-Nov-1993 %
% %
% %
% RX,TX Mode 0 Video Cut Through %
% RX,TX Mode 1 Clock div 8 %
% RX,TX Mode 2 Clock div 4 %
% RX,TX Mode 3 Clock div 2 %
% %
% 0002 is control VCI %
% 0001 is data VCI %
% VPI is ignored for both %
% %
% First byte of control cell format: %
% abccdeff %
% %
% a OR in default mode value for TX mode %
% b OR in value cc for TX mode %
% cc value ORed in if b is set %
% d OR in default mode value for RX mode %
% e OR in value ff for RX mode %
% ff value ORed in if e is set %
% %

DESIGN IS "VCRX"
BEGIN
    DEVICE IS "EPM5032DC-1"
    BEGIN
        default_mode0 @  28 : INPUT  ;
        default_mode1 @  27 : INPUT  ;
        _fifo_ae @  16 : INPUT  ;
        _reset   @  15 : INPUT  ;
        rx_byteclk @   2 : INPUT  ;
        rx_data_in0 @  14 : INPUT  ;
        rx_data_in1 @  13 : INPUT  ;
        rx_data_in2 @   1 : INPUT  ;
        rx_data_in3 @  11 : INPUT  ; %MC13%
        rx_data_in4 @  12 : INPUT  ; %MC15%
        rx_data_in5 @  17 : INPUT  ; %MC17%
        rx_data_in6 @  18 : INPUT  ; %MC19%
        rx_data_in7 @  19 : INPUT  ; %MC21%
        rx_hen_in @  20 : INPUT  ; %MC23%
        rx_pen_in @  23 : INPUT  ; %MC25%
        rx_ten_in @  24 : INPUT  ; %MC27%
        _cut_through_e @   3 : OUTPUT ; %MC1%
        data_enabled @   6 : OUTPUT ; %MC7%
        _fifo_oe @   4 : OUTPUT ; %MC3%
        p_fifo_ren @   5 : OUTPUT ; %MC5%
        ptx_mode0 @   9 : OUTPUT ; %MC9%
        ptx_mode1 @  10 : OUTPUT ; %MC11%
_fifo_ef @ 26 :input;
_fifo_ff @ 25 :input;
    END;
END;
```

```
subdesign vcrx
(
rx_data_in[7..0] :input;
rx_hen_in :input;
rx_pen_in :input;
rx_ten_in :input;
rx_byteclk :input;
_reset :input;
_fifo_oe :output;
data_enabled :output;
p_fifo_ren :output;
_fifo_ae :input;
_cut_through_e :output;
default_mode[1..0] :input;
ptx_mode[1..0] :output;
_fifo_ef :input;
_fifo_ff :input;
)

variable

rx_data[7..0] :dff;
rx_hen :dff;
rx_ten :dff;
rx_pen :dff;
rx_mode[1..0] :dff;
tx_mode[1..0] :dff;
header_byte[2..0] :dff;
cell_active :dff;
cell_active_data :dff;
cell_active_cont :dff;
_fifo_ren :dff;
running[1..0] :dff;
out_count[2..0] :dff;
sync_reset :dff;


begin

sync_reset.prn =_reset;
sync_reset.clk =sclk(rx_byteclk);
sync_reset =sync_reset & !_reset;

rx_data[].clrn =_reset;
rx_data[].clk =sclk(rx_byteclk);
rx_data[] =rx_data_in[];

rx_hen.clrn =_reset;
rx_hen.clk =sclk(rx_byteclk);
rx_hen =rx_hen_in;

rx_pen.clrn =_reset;
rx_pen.clk =sclk(rx_byteclk);
rx_pen =rx_pen_in;

rx_ten.clrn =_reset;
rx_ten.clk =sclk(rx_byteclk);
rx_ten =rx_ten_in;
```

```
header_byte[].clrn =_reset;
header_byte[].clk =sclk(rx_byteclk);
header_byte[] =(0 & !rx_hen & !rx_pen & !rx_ten) #
 (header_byte[]+1 & rx_hen & !rx_pen & !rx_ten) #
 (header_byte[] & rx_pen ) #
 (header_byte[] & rx_ten);

cell_active.clrn =_reset;
cell_active.clk =sclk(rx_byteclk);
cell_active =((header_byte[]==1) & (rx_data[3..0]==0) &
!rx_pen & !rx_ten) #
 ((header_byte[]==2) & (rx_data[]==0) & !rx_pen &
!rx_ten & cell_active) #
 (cell_active & rx_ten) #
 (cell_active & rx_pen);

data_enabled =cell_active_data;
cell_active_data.clrn =_reset;
cell_active_data.clk =sclk(rx_byteclk);
cell_active_data =(running[]!=3)
                                &
                                (
                                ((header_byte[]==3) & (rx_data[7..4]==1) &
!rx_pen & !rx_ten & cell_active) #
 (cell_active_data & (header_byte[]!=3)) #
 (cell_active_data & rx_pen) #
 (cell_active_data & rx_ten)
                                );

cell_active_cont.clrn =_reset;
cell_active_cont.clk =sclk(rx_byteclk);
cell_active_cont =((header_byte[]==3) & (rx_data[7..4]==2) &
!rx_pen & !rx_ten & cell_active) #
 (cell_active_cont & (header_byte[]!=3)) #
 (cell_active_cont & rx_pen) #
 (cell_active_cont & rx_ten);

ptx_mode[] =tx_mode[];
tx_mode[].clrn =_reset;
tx_mode[].clk =sclk(rx_byteclk);
tx_mode[] =(
(
(default_mode[] & rx_data[7]) #
(rx_data[5..4] & rx_data[6])
) &
(header_byte[]==5) &
cell_active_cont &
!rx_pen & !rx_ten
 ) #
 (tx_mode[] & (header_byte[]!=5)) #
 (tx_mode[] & !cell_active_cont) #
 (tx_mode[] & rx_pen) #
 (tx_mode[] & rx_ten) #
 (default_mode[] & sync_reset);

rx_mode[].clrn =_reset;
rx_mode[].clk =sclk(rx_byteclk);
rx_mode[] =(
```

```
(
(default_mode[] & rx_data[3]) #
(rx_data[1..0] & rx_data[2])
) &
(header_byte[]==5) &
cell_active_cont &
!rx_pen & !rx_ten
 ) #
 (rx_mode[] & (header_byte[]!=5)) #
 (rx_mode[] & !cell_active_cont) #
 (rx_mode[] & rx_pen) #
 (rx_mode[] & rx_ten) #
 (default_mode[] & sync_reset);

p_fifo_ren =_fifo_ren;
_fifo_ren.prn =_reset;
_fifo_ren.clk =sclk(rx_byteclk);
_fifo_ren =!(
((out_count[]==0) & running[1]) #
((rx_mode[]==2) & (out_count[]==4) & running[1]) #
  ((rx_mode[]==3) & (out_count[]==2) & running[1]) #
((rx_mode[]==3) & (out_count[]==4) & running[1]) #
((rx_mode[]==3) & (out_count[]==6) & running[1])
  );

out_count[].clrn =_reset;
out_count[].clk =sclk(rx_byteclk);
out_count[] =out_count[]+1;

        %running[]  == 0 is empty%
        %running[]  == 3 is full%
        %running[]  == 2 is play%
        %running[]  == 1 is undefined%
running[].clrn =_reset;
running[].clk =sclk(rx_byteclk);
running[] = running[] == 0 & _fifo_ae  & 2
                          #
                          running[] == 3 & !_fifo_ae  & 2
                          #
                          running[] == 3 & _fifo_ae & 3
                          #
                          running[] == 2 & _fifo_ef & _fifo_ff & 2
                          #
  running[] == 2 & !_fifo_ff & 3;

_cut_through_e =(rx_mode[]!=0);

_fifo_oe =(rx_mode[]==0);

end;
```

```
% %
% Cheap video receiver datapath %
% %
% C. Brendan S. Traw %
% 15-Nov-1993 %
% %

DESIGN IS "VDRX"
BEGIN
    DEVICE IS "EPM5032DC-1"
    BEGIN
        data_enabled @  28 : INPUT  ;
        _reset   @  27 : INPUT  ;
        rx_byteclk @   2 : INPUT  ;
        rx_data_in0 @  16 : INPUT  ;
        rx_data_in1 @  15 : INPUT  ;
        rx_data_in2 @  14 : INPUT  ;
        rx_data_in3 @  13 : INPUT  ;
        rx_data_in4 @   1 : INPUT  ;
        rx_data_in5 @   5 : INPUT  ; %MC5%
        rx_data_in6 @   6 : INPUT  ; %MC7%
        rx_data_in7 @   9 : INPUT  ; %MC9%
        rx_hen_in @  24 : INPUT  ; %MC27%
        rx_pen_in @  25 : INPUT  ; %MC29%
        rx_ten_in @  26 : INPUT  ; %MC31%
        p_fifo_wen1 @   3 : OUTPUT ; %MC1%
        pfifo_wen2 @   4 : OUTPUT ; %MC3%
        rx_data_out0 @  10 : OUTPUT ; %MC11%
        rx_data_out1 @  11 : OUTPUT ; %MC13%
        rx_data_out2 @  12 : OUTPUT ; %MC15%
        rx_data_out3 @  17 : OUTPUT ; %MC17%
        rx_data_out4 @  18 : OUTPUT ; %MC19%
        rx_data_out5 @  19 : OUTPUT ; %MC21%
        rx_data_out6 @  20 : OUTPUT ; %MC23%
        rx_data_out7 @  23 : OUTPUT ; %MC25%
    END;
END;


subdesign vdrx
(
rx_data_in[7..0] :input;
rx_data_out[7..0] :output;
rx_hen_in :input;
rx_pen_in :input;
rx_ten_in :input;
rx_byteclk :input;
_reset :input;
data_enabled :input;
p_fifo_wen1 :output;
pfifo_wen2 :output;
)

variable

rx_data[7..0] :dff;
_fifo_wen1 :dff;
fifo_wen2 :dff;
flag_count[2..0] :dff;
```

```
setup :dff;
_reset_sync :dff;

begin

_reset_sync.clk =sclk(rx_byteclk);
_reset_sync =_reset;

rx_data_out[] =rx_data[];
rx_data[].clrn =_reset;
rx_data[].clk =sclk(rx_byteclk);
rx_data[] =(rx_data_in[] & (flag_count[]==0)) #
 (h"08" & (flag_count[]==2));

p_fifo_wen1 =_fifo_wen1;
_fifo_wen1.prn =_reset;
_fifo_wen1.clk =sclk(rx_byteclk);
_fifo_wen1 =!(
(!rx_hen_in & !rx_ten_in &
!rx_pen_in & data_enabled) #
(flag_count[]!=0)
  );

pfifo_wen2 =fifo_wen2;
fifo_wen2.clrn =_reset;
fifo_wen2.clk =sclk(rx_byteclk);
fifo_wen2 =(flag_count[]==0) & _reset_sync;

flag_count[].clrn =_reset;
flag_count[].clk =sclk(rx_byteclk);
flag_count[] =(flag_count[]+1 & !setup);

setup.clrn =_reset;
setup.clk =sclk(rx_byteclk);
setup =(flag_count[]==3) #
 setup;
end;
```

```
% %

% Cheap audio rx controller  %
% %

% C. Brendan S. Traw %
% Bill Marcus %

% 30-Mar-1994 %

% %


DESIGN IS "ARX"

BEGIN

    DEVICE IS "EPM5032DC-1"

    BEGIN

    END;

END;




subdesign arx

(

pcmin :output;
p_afifo_ren :output;
pcmout :input;
clk12 :input; % about 1.5 MHz dataclock %
fs :input; % about 8KHz framestrobe %
_reset :input;
_cut_through :input;
parx_data[7..0] :input;
_afifo_pae :input;
)


variable


arx_data[7..0] :dff;

bit_cnt[2..0] :dff;
_afifo_ren :dff;
load :dff;

begin
```

```
bit_cnt[].clrn =_reset;
bit_cnt[].clk =clk12;
bit_cnt[] =(1 & fs) #
 (bit_cnt[]+1 & (bit_cnt[]>0));

p_afifo_ren =_afifo_ren;
_afifo_ren.prn =_reset;
_afifo_ren.clk =clk12;
_afifo_ren =!((bit_cnt[]==7) & !_afifo_pae);

load.clrn =_reset;
load.clk =clk12;
load =!_afifo_ren;

arx_data[].clrn =_reset;
arx_data[].clk =clk12;

pcmin =(arx_data[7] & _cut_through) #
 (pcmout & !_cut_through);

arx_data[7] =(parx_data[7] & load) #
 (arx_data[6] & (bit_cnt[]>0)) #
 (arx_data[7] & (bit_cnt[]==0));
arx_data[6] =(parx_data[6] & load) #
 (arx_data[5] & (bit_cnt[]>0)) #
 (arx_data[6] & (bit_cnt[]==0));
arx_data[5] =(parx_data[5] & load) #
 (arx_data[4] & (bit_cnt[]>0)) #
 (arx_data[5] & (bit_cnt[]==0));
arx_data[4] =(parx_data[4] & load) #
 (arx_data[3] & (bit_cnt[]>0)) #
 (arx_data[4] & (bit_cnt[]==0));
arx_data[3] =(parx_data[3] & load) #
 (arx_data[2] & (bit_cnt[]>0)) #
 (arx_data[3] & (bit_cnt[]==0));
arx_data[2] =(parx_data[2] & load) #
 (arx_data[1] & (bit_cnt[]>0)) #
 (arx_data[2] & (bit_cnt[]==0));
arx_data[1] =(parx_data[1] & load) #
 (arx_data[0] & (bit_cnt[]>0)) #
 (arx_data[1] & (bit_cnt[]==0));
arx_data[0] =(parx_data[0] & load) #
 (arx_data[0] & (bit_cnt[]==0));

end;
```

```
% %
% Cheap audio receiver datapath %
% %
% C. Brendan S. Traw %
% 15-Nov-1993 %
% %

DESIGN IS "ADRX"
BEGIN
    DEVICE IS "EPM5032DC-1"
    BEGIN
        data_enabled @  28 : INPUT  ;
        _reset   @  27 : INPUT  ;
        rx_byteclk @   2 : INPUT  ;
        rx_data_in0 @  16 : INPUT  ;
        rx_data_in1 @  15 : INPUT  ;
        rx_data_in2 @  14 : INPUT  ;
        rx_data_in3 @  13 : INPUT  ;
        rx_data_in4 @   1 : INPUT  ;
        rx_data_in5 @   5 : INPUT  ; %MC5%
        rx_data_in6 @   6 : INPUT  ; %MC7%
        rx_data_in7 @   9 : INPUT  ; %MC9%
        rx_hen_in @  24 : INPUT  ; %MC27%
        rx_pen_in @  25 : INPUT  ; %MC29%
        rx_ten_in @  26 : INPUT  ; %MC31%
        p_fifo_wen1 @   3 : OUTPUT ; %MC1%
        pfifo_wen2 @   4 : OUTPUT ; %MC3%
        rx_data_out0 @  10 : OUTPUT ; %MC11%
        rx_data_out1 @  11 : OUTPUT ; %MC13%
        rx_data_out2 @  12 : OUTPUT ; %MC15%
        rx_data_out3 @  17 : OUTPUT ; %MC17%
        rx_data_out4 @  18 : OUTPUT ; %MC19%
        rx_data_out5 @  19 : OUTPUT ; %MC21%
        rx_data_out6 @  20 : OUTPUT ; %MC23%
        rx_data_out7 @  23 : OUTPUT ; %MC25%
    END;
END;


subdesign adrx
(
rx_data_in[7..0] :input;
rx_data_out[7..0] :output;
rx_hen_in :input;
rx_pen_in :input;
rx_ten_in :input;
rx_byteclk :input;
_reset :input;
data_enabled :input;
p_fifo_wen1 :output;
pfifo_wen2 :output;
)

variable

rx_data[7..0] :dff;
_fifo_wen1 :dff;
fifo_wen2 :dff;
flag_count[2..0] :dff;
```

```
setup :dff;
_reset_sync :dff;

begin

_reset_sync.clk =sclk(rx_byteclk);
_reset_sync =_reset;

rx_data_out[] =rx_data[];
rx_data[].clrn =_reset;
rx_data[].clk =sclk(rx_byteclk);
rx_data[] =(rx_data_in[] & (flag_count[]==0)) #
 (h"48" & (flag_count[]==1));

p_fifo_wen1 =_fifo_wen1;
_fifo_wen1.prn =_reset;
_fifo_wen1.clk =sclk(rx_byteclk);
_fifo_wen1 =!(
(!rx_hen_in & !rx_ten_in &
!rx_pen_in & data_enabled) #
(flag_count[]!=0)
  );

pfifo_wen2 =fifo_wen2;
fifo_wen2.clrn =_reset;
fifo_wen2.clk =sclk(rx_byteclk);
fifo_wen2 =(flag_count[]==0) & _reset_sync;

flag_count[].clrn =_reset;
flag_count[].clk =sclk(rx_byteclk);
flag_count[] =(flag_count[]+1 & !setup);

setup.clrn =_reset;
setup.clk =sclk(rx_byteclk);
setup =(flag_count[]==3) #
 setup;
end;
```

```
%************************************************************
                        William S. Marcus
                          Brendan Traw
                            Bellcore
                          30-March-94


This PAL provides all digital data path I/O between the
TCM29C19 and the TX.  It also generates the control signals
necessary to put data into FIFO TX.  Finally it is responsible
for loading the programmable offset register for FIFO TX's /pae.

FIFO TX is a 64x9

************************************************************%

CONSTANT PAEOffset = 47;

TITLE "audtx";

DESIGN IS "audtx"
BEGIN
    DEVICE IS "EPM5032-1";
END;

SUBDESIGN audtx
(
     /rst, % active low asynchronous reset %
     clk12, % 19.44MHz/12 -- created by audtime EPLD %
     fs, % frame strobe -- created by audtime EPLD %
     pcmout % serial pcm output from TCM29C19 %
     :INPUT;


     pcmtx[7..0], % parallel output to FIFO TX %
     /ld, % load input to FIFO TX %
     /wen1 % write enable to FIFO TX %
     :OUTPUT;

)

VARIABLE
      fastck :SCLK;

      cnt[1..0] :DFF; % used to cycle through reset procedure %
      strcnt[3..0] :DFF; % used to generate /wen1 %

      p[7..0] :DFF;
      i/ld :DFF;
      i/wen1 :DFF;
BEGIN

     fastck = clk12;

     cnt[].clk  = fastck;
     cnt[].clrn = /rst;
     cnt[]      = cnt[] < 3 & cnt[]+1
                  #
                  cnt[]==3 & 3;
```

```
    /ld       = i/ld;
    i/ld.clk  = fastck;
    i/ld.clrn = /rst;
    !i/ld     = cnt[] == 2;


    pcmtx[]  = p[];
    p[].clk   = fastck;
    p[]       = (cnt[] <  3) & PAEOffset
                #
                (cnt[] == 3) & (p[6..0],pcmout);

    strcnt[].clrn = /rst;
    strcnt[].clk  = fastck;
    strcnt[]      = fs & 8
                    #
                    !fs & (strcnt[] > 7) & strcnt[]+1;


    /wen1        = i/wen1;
    i/wen1.prn  = /rst;
    i/wen1.clk  = fastck;
    !i/wen1     = cnt[]==2
                  #
                  cnt[]==3 & strcnt[]==H"f";

END;
```

```
% %
% Cheap video hengen.tdf   %
% %
% C. Brendan S. Traw %
% 19-Jan-1994 %
% %

DESIGN IS "HENGEN"
BEGIN
    DEVICE IS "EPM5032DC-1"
    BEGIN
        lof      @  26 : INPUT  ; % dmm added 01/25/94 - Loss Of Frame pin %
        byte_clk @   2 : INPUT  ;
        _null    @  28 : INPUT  ;
        _reset   @  27 : INPUT  ;
        _shared  @  16 : INPUT  ;
        prx_hen  @   3 : OUTPUT ; %MC1%
        prx_pen  @   4 : OUTPUT ; %MC3%
        prx_ten  @   5 : OUTPUT ; %MC5%
        pshrx_hen @   6 : OUTPUT ; %MC7%
        pshrx_pen @   9 : OUTPUT ; %MC9%
        pshrx_ten @  10 : OUTPUT ; %MC11%
        pshtx_hen @  11 : OUTPUT ; %MC13%
        pshtx_pen @  12 : OUTPUT ; %MC15%
        pshtx_ten @  17 : OUTPUT ; %MC17%
        ptx_hen  @  18 : OUTPUT ; %MC19%
        ptx_pen  @  19 : OUTPUT ; %MC21%
        ptx_ten  @  20 : OUTPUT ; %MC23%
        shared   @  23 : OUTPUT ; %MC25%
    END;
END;

subdesign hengen
(
    ptx_hen          :output;
    pshtx_hen        :output;
    prx_hen          :output;
    pshrx_hen        :output;
    ptx_pen          :output;
    pshtx_pen        :output;
    prx_pen          :output;
    pshrx_pen        :output;
    ptx_ten          :output;
    pshtx_ten        :output;
    prx_ten          :output;
    pshrx_ten        :output;
    lof              :input; % dmm added 01/25/94, (unused/reserved) %
    _null            :input;
    _shared          :input;
    shared           :output;
    byte_clk         :input;
    _reset     :input;
)

variable
    state[5..0]      :dff;
    hen              :dff;
    btx_hen          :tri;
```

```
    bshtx_hen       :tri;
    brx_hen         :tri;
    bshrx_hen       :tri;
    btx_pen         :tri;
    bshtx_pen       :tri;
    brx_pen         :tri;
    bshrx_pen       :tri;
    btx_ten         :tri;
    bshtx_ten       :tri;
    brx_ten         :tri;
    bshrx_ten       :tri;

begin

state[].clrn    =_reset;
state[].clk     =sclk(byte_clk);
state[]         =(state[]+1 & (state[]<52)) #
                 (0 & (state[]==52));

hen.clrn =_reset;
hen.clk =sclk(byte_clk);
hen             =(state[]>0) & (state[]<6);

ptx_hen         =btx_hen;
btx_hen.oe      =!_null;
btx_hen         =hen;

pshtx_hen       =bshtx_hen;
bshtx_hen.oe    =!_null;
bshtx_hen       =hen;

prx_hen         =brx_hen;
brx_hen.oe      =!_null;
brx_hen         =hen;

pshrx_hen       =bshrx_hen;
bshrx_hen.oe    =!_null;
bshrx_hen       =hen;

ptx_ten         =btx_ten;
btx_ten.oe      =!_null;
btx_ten         =gnd;

pshtx_ten       =bshtx_ten;
bshtx_ten.oe    =!_null;
bshtx_ten       =gnd;

prx_ten         =brx_ten;
brx_ten.oe      =!_null;
brx_ten         =gnd;

pshrx_ten       =bshrx_ten;
bshrx_ten.oe    =!_null;
bshrx_ten       =gnd;

ptx_pen         =btx_pen;
btx_pen.oe      =!_null;
btx_pen         =gnd;

pshtx_pen       =bshtx_pen;
bshtx_pen.oe    =!_null;
```

```
bshtx_pen       =gnd;

prx_pen         =brx_pen;
brx_pen.oe      =!_null;
brx_pen         =gnd;

pshrx_pen       =bshrx_pen;
bshrx_pen.oe    =!_null;
bshrx_pen       =gnd;

shared          =!_shared;

end;
```

```
%************************************************************
                        William S. Marcus
                          Brendan Traw
                            Bellcore
                          30-March-94


This EPLD genrates clk12 and fs from the 19.44MHz clock

************************************************************%


TITLE "audclk";

DESIGN IS "audclk"
BEGIN
    DEVICE IS "EPM5032-1";
END;

SUBDESIGN audclk
(

     clkin
     :INPUT;


     clk12,
     fs
     :OUTPUT;

)

VARIABLE

      cnt[3..0] :DFF;
      iclk12 :DFF;

      fcnt[7..0] :DFF;
      ifs :DFF;
BEGIN


     cnt[].clk = clkin;
     cnt[]     = (cnt[] < 11) & (cnt[]+1);

     clk12      = iclk12;
     iclk12.clk = clkin;
     iclk12     = cnt[] < 6;

     fcnt[].clk  = iclk12;
     fcnt[]      = (fcnt[] < 202) & (fcnt[]+1);

     fs          = ifs;
     ifs.clk  = iclk12;
     ifs         = (fcnt[] == 202);

END;
```

```
%************************************************************
                        William S. Marcus
                            Bellcore

              template was srcrdr pal of cheapvideo 1

                            Date: 4/7/94
                        Project: CheapVideo2


This PAL does two functions:
(1) Provides ATM cells to an Aurora/Sunshine Electrical
    STS-3c interface (as specified by GRL's OC-12 mux/demux
    board).  ATM cell bodies are found in two places:
    a. The digital video 72201 FIFO
    b. The digital audio 72201 FIFO.

    When the audio FIFO has atleast one cell body (/pae high), a cell
    body will be read out during the next ATM cell time of the STS-3c
    link.  When the video FIFO has atleast one cell body, a cell body
    will be read out only if no audio cell body is available during
    the next ATM cell time of the STS-3c link.  If no cell body is
    available in the FIFOs this PAL shall provide a payload of all 0x00.
    All ATM cell headers are provided by this PAL.  The video data
    shall have a VCI of 0x0001.  The audio data shall have a VCI of
    0x0002.  Idle cells shall have a VCI of 0x0000.  The CRC's for
    each VCI are 0x25, 0xb0, 0x55.

NOTE: The design will produce incorrect cells for the first
      several ATM cell cycles, after which everything will fall in
      to place.  This "feature" has minimal, bordering on none, harmful
      effects of the performance, but made creating the design much
      easier to create.

(2) Divide the SONET clock 2

************************************************************%

TITLE "srcrdr";

DESIGN IS "srcrdr"
BEGIN
    DEVICE IS "EPM5032-1"
    BEGIN
        bycko     @  2 : INPUT;
        /rst      @ 14 : INPUT;
        hen       @ 16 : INPUT;
        pen       @ 27 : INPUT;
        ten       @ 28 : INPUT;
        /paeV     @  1 : INPUT;
        /paeA     @  3 : INPUT;

        /fiforenV @  9 : OUTPUT;
        /fifooeV  @ 10 : OUTPUT;
        /fiforenA @  4 : OUTPUT;
        /fifooeA  @  5 : OUTPUT;
        tdo1      @ 26 : OUTPUT;
        tdo2      @ 25 : OUTPUT;
        tdo3      @ 24 : OUTPUT;
```

```
        tdo4      @ 23 : OUTPUT;
        tdo5      @ 20 : OUTPUT;
        tdo6      @ 19 : OUTPUT;
        tdo7      @ 18 : OUTPUT;
        tdo8      @ 17 : OUTPUT;
        clkdiv    @  6 : OUTPUT;
    END;
END;

SUBDESIGN srcrdr
(
     /rst, % active low asynchronous reset %
     hen, % Bellcore Framer tx_hen %
     pen, % Bellcore Framer tx_pen %
     ten, % Bellcore Framer tx_ten %
     bycko, % Bellcore Framer tx_bycko %
     /paeV, % video fifo has no available cell %
     /paeA  % audio fifo has no available cell %
     :INPUT;

     /fiforenV, % fifo /ren signal to video fifo %
     /fifooeV, % fifo /oe signal  to video fifo %
     /fiforenA, % fifo /ren signal to audio fifo %
     /fifooeA, % fifo /oe signal  to audio fifo %
     tdo[1..8], % carries all headers, and idle bodies %
     clkdiv % divided down clock  %
     :OUTPUT;

)

VARIABLE
     hend1 :DFF; % hen delayed on clock time %
     bytecnt[5..0] :DFF; % sequencer for generating control signals %
     dwnstate[1..0] :DFF;
     upstate[1..0] :DFF;
     /frenV :DFF; % fifo /REN signal to video fifo%
     /foeV :DFF; % fifo /OE signal to video fifo  %
     /frenA :DFF; % fifo /REN signal to audio fifo%
     /foeA :DFF; % fifo /OE signal to audio fifo  %
     q[7..0] :DFF;
     tbuf[7..0] :TRI;
     dvdr :DFF;

BEGIN
     % description of hend1 %
     hend1.clk  = !bycko;
     hend1      = hen;

     % description of bytecnt[] %
     bytecnt[].clk  = !bycko;
     bytecnt[].clrn = /rst;
     bytecnt[]  = (!hen # hend1) & bytecnt[]<52 & (!pen & !ten) & bytecnt[]+1
                  #
                  (!hen # hend1) & bytecnt[]<53 & (pen  # ten) & bytecnt[];

     % decription of dwnstate[] and upstate[] %
     % dwnstate[] == 0 current cell being read is an idle cell     %
     % dwnstate[] == 1 current cell being read is a video cell     %
     % dwnstate[] == 2 current cell being read is a audio cell     %
```

```
% dwnstate[] is used to gate falling edge clocked control     %
% signals (ie. /fiforenV /fiforenA).                          %
% upstate conveys the same meaning as dwnstate, but is used   %
% to gate rising edge clocked control signals (i.e. /fifooeV). %

dwnstate[].clk  = !bycko;
upstate[].clk   = !bycko;

dwnstate[].clrn = /rst;
upstate[].clrn  = /rst;

dwnstate[]      = (bytecnt[] != 50 # ten # pen) & dwnstate[]
                  #
                  (bytecnt[] == 50 & !ten & !pen) & /paeA & 2
                  #
                  (bytecnt[] == 50 & !ten & !pen) & !/paeA & /paeV & 1;

upstate[]       = (bytecnt[] == 51) & !ten & !pen & dwnstate[]
                  #
                  (bytecnt[] == 51) & (ten # pen) & upstate[]
                  #
                  bytecnt[] != 51 & upstate[];

% description of /fifoenA %
/fiforenA  = /frenA;
/frenA.clk = !bycko;
/frenA.prn = /rst;
!/frenA   = dwnstate[] == 2 & bytecnt[] > 2 & bytecnt[] < 51 & !ten & !pen;

% description of /fifooeA %
/fifooeA   = /foeA;
/foeA.clk  = bycko;
/foeA.prn  = /rst;
!/foeA     = upstate[] == 2 &  bytecnt[] >= 4 & bytecnt[] <= 51;

% description of /fifoenV %
/fiforenV  = /frenV;
/frenV.clk = !bycko;
/frenV.prn = /rst;
!/frenV   = dwnstate[] == 1 & bytecnt[] > 2 & bytecnt[] < 51 & !ten & !pen;

% description of /fifooeV %
/fifooeV   = /foeV;
/foeV.clk  = bycko;
/foeV.prn  = /rst;
!/foeV     = upstate[] == 1 &  bytecnt[] >= 4 & bytecnt[] <= 51;

% description of tdo[1..8] %
tdo[1..8] = tbuf[].out;
tbuf[].oe = /foeV & /foeA;
tbuf[].in = q[];

q[].clk  = bycko;
q[].clrn = /rst;
q[]      = upstate[] == 1 & bytecnt[] == 2 & H"10"
           #
           upstate[] == 1 & bytecnt[] == 3 & H"25"
           #
           upstate[] == 2 & bytecnt[] == 2 & H"20"
```

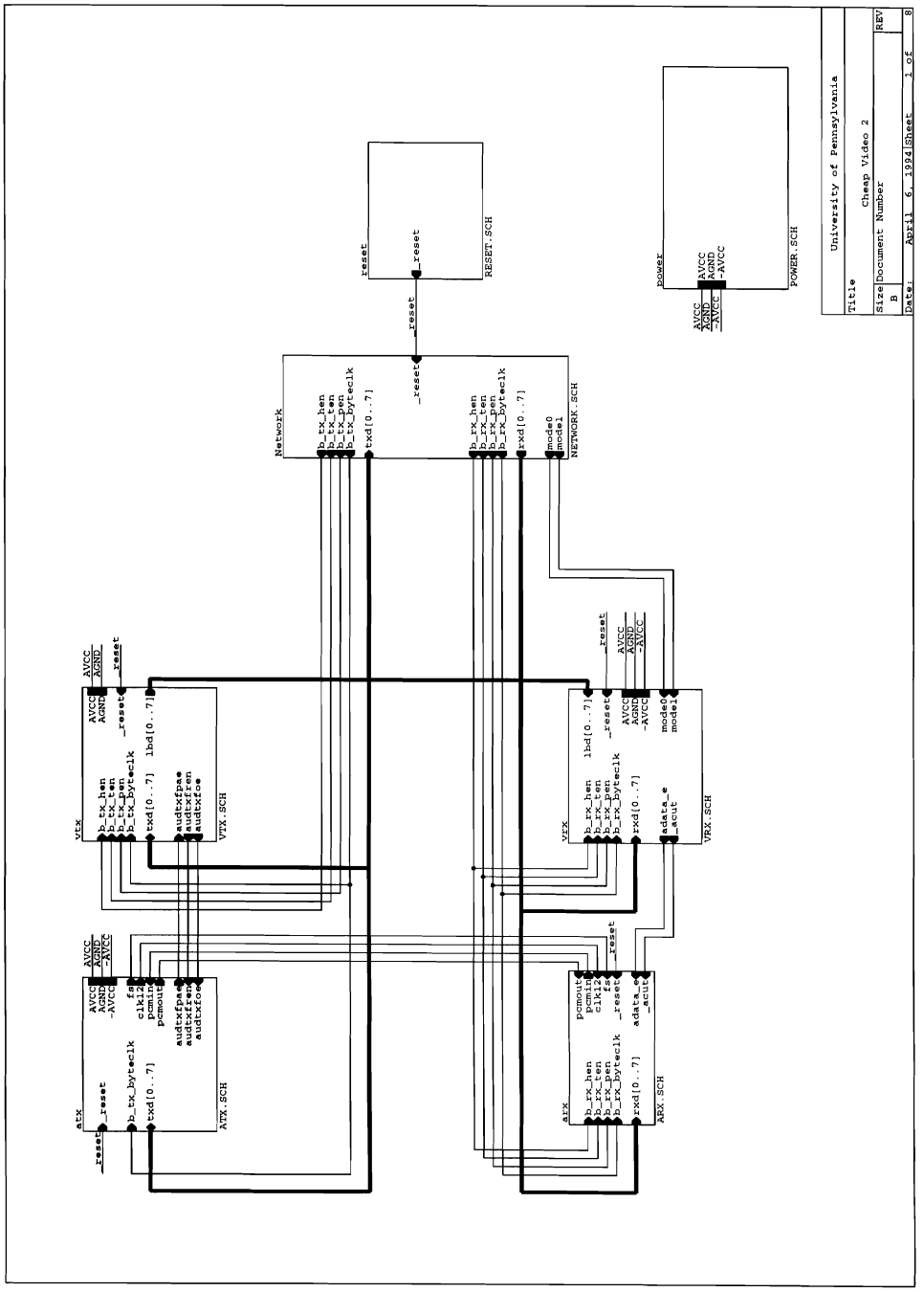
```
                #
                upstate[] == 2 & bytecnt[] == 3 & H"b5"
                #
                upstate[] == 0 & bytecnt[] == 3 & H"55";

    % description of clkdiv %
    clkdiv      = dvdr;
    dvdr.clk  = bycko;
    dvdr.clrn = /rst;
    dvdr      = !dvdr;

END;
```

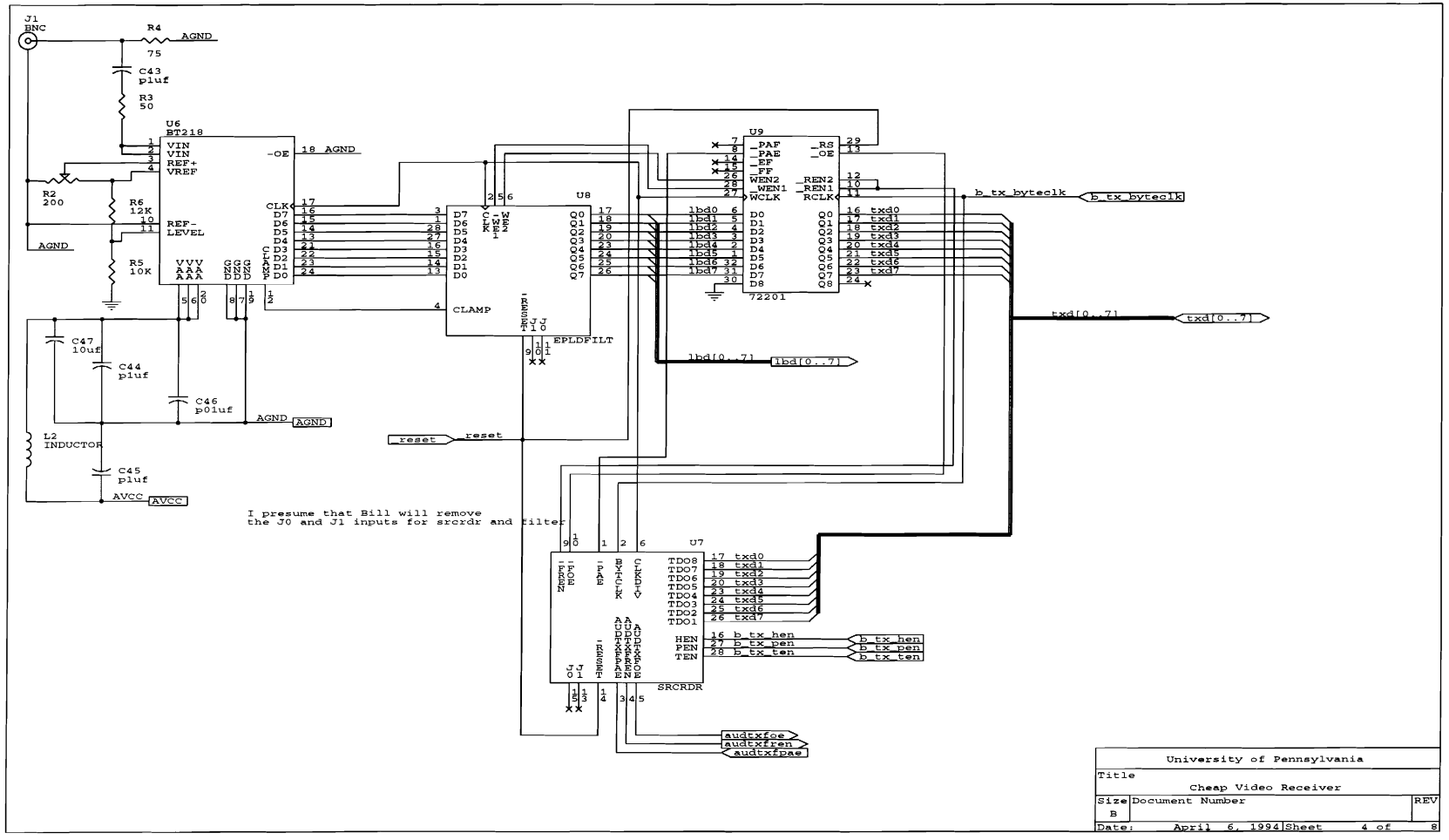
```
%************************************************************
                        William S. Marcus
                            Bellcore
                          16-November-93

       Glue Logic which resides in the data path bwt a
       Brooktree Bt218KP20 (A/D) and an IDT 72201 FIFO
       (256 x 9, with programmable flags).

FUNCTIONS:
       (1) Assert CLAMP signal to Bt218 during reset, as well
           as during horizontal synch. NOTE: horizontal synch
           pulse is 5us in duration and has a DC baised value
           of 0 Volts (blanking). While CLAMP is being asserted,
           all 0s are asserted on the q[7..0] outputs.

           I'd like to guarantee that the Bt218 is clamped
           for atleast half of the horizontal synch time, or
           2.5us.

       (2) Load FIFO's almost empty programmable flag register
           immediately following a reset.

       (3) Pass latched output of Bt218 to input of FIFO.


************************************************************%

% CONSTANTS %
CONSTANT Vthres0      = 32; % signal voltage ~= ~.128 Volts %
CONSTANT Vthres1      = 64; % signal voltage ~= ~.256 Volts %
CONSTANT PAEOffsetLSB = 48; % /PAE goes high when > 48 bytes in FIFO %
CONSTANT Clamper      = 25;

TITLE "filter";

DESIGN IS "FILTER"
BEGIN
    DEVICE IS "EPM5032-15"
    BEGIN
        clock    @  2  : INPUT  ;
        d0       @  13 : INPUT  ;
        d1       @  14 : INPUT  ;
        d2       @  15 : INPUT  ;
        d3       @  16 : INPUT  ;
        d4       @  27 : INPUT  ;
        d5       @  28 : INPUT  ;
        d6       @   1 : INPUT  ;
        d7       @   3 : INPUT  ;
        /rst     @   9 : INPUT  ;
        clamp    @   4 : OUTPUT ;
        /wen1    @   5 : OUTPUT ;
        wen2     @   6 : OUTPUT ;
        q7       @   26 : OUTPUT ;
        q6       @   25 : OUTPUT ;
        q5       @   24 : OUTPUT ;
        q4       @   23 : OUTPUT ;
        q3       @   20 : OUTPUT ;
        q2       @   19 : OUTPUT ;
```

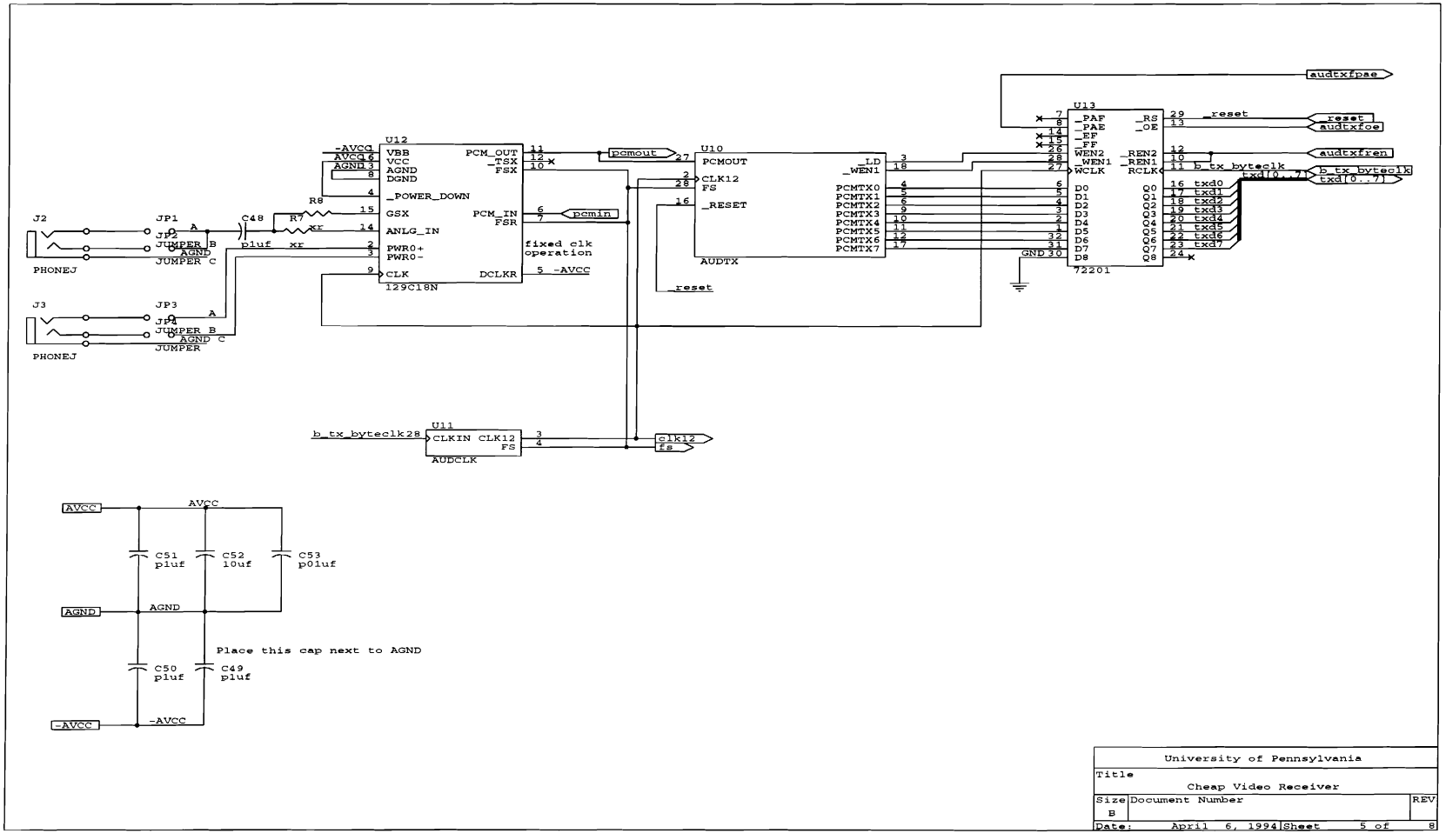
```
        q1        @   18 : OUTPUT ;
        q0        @   17 : OUTPUT ;
    END;
END;

SUBDESIGN filter
(
     clock, % STS-3c byck0 divided by n (n=2,4,8) %
     d[7..0], % data from bt218 %
     /rst               % board level reset signal %
     :INPUT;

     /wen1, % to FIFO %
     wen2,  % to FIFO %
     clamp,             % to bt218 %
     q[7..0] % to FIFO %
     :OUTPUT;

)

VARIABLE
     i/wen1 :DFF; % see /wen1 %
     iwen2 :DFF;   % see wen2 %
     iclamp :DFF;   % see clamp %
     iq[7..0] :DFF;   % see q[7..0] %

     rcnt[2..0] :DFF;   % counter as sequencer for loading
  FIFO's almost empty flag. %
     gclock :SCLK;  % global clock buffer %
     schmitt[1..0] :DFF; % used as a schmitt trigger for clamp %
     d1schmitt1 :DFF;   % shcmitt[1] delayed one cycle time %
     vshot :DFF;   % one shot when video signal makes the
  transition into synch range %
     ccnt[6..0] :DFF; % sequencer which drives clamp circuitry %

BEGIN

     % use global clock distribution %
     gclock = clock;

     % description of rcnt[] %
     rcnt[].clk  = gclock;
     rcnt[].clrn = /rst;
     rcnt[]      = (rcnt[] < 7)  & rcnt[]+1
                   #
                   (rcnt[] == 7) & 7;

     % description of wen2 %
     wen2       = iwen2;
     iwen2.clk  = gclock;
     iwen2.clrn = /rst;
     iwen2      = (rcnt[] != 2);

     % description of /wen1 %
     /wen1      = i/wen1;
     i/wen1.clk = gclock;
     i/wen1.prn = /rst;
     i/wen1     = rcnt[]!=7 & rcnt[]!=2;
```

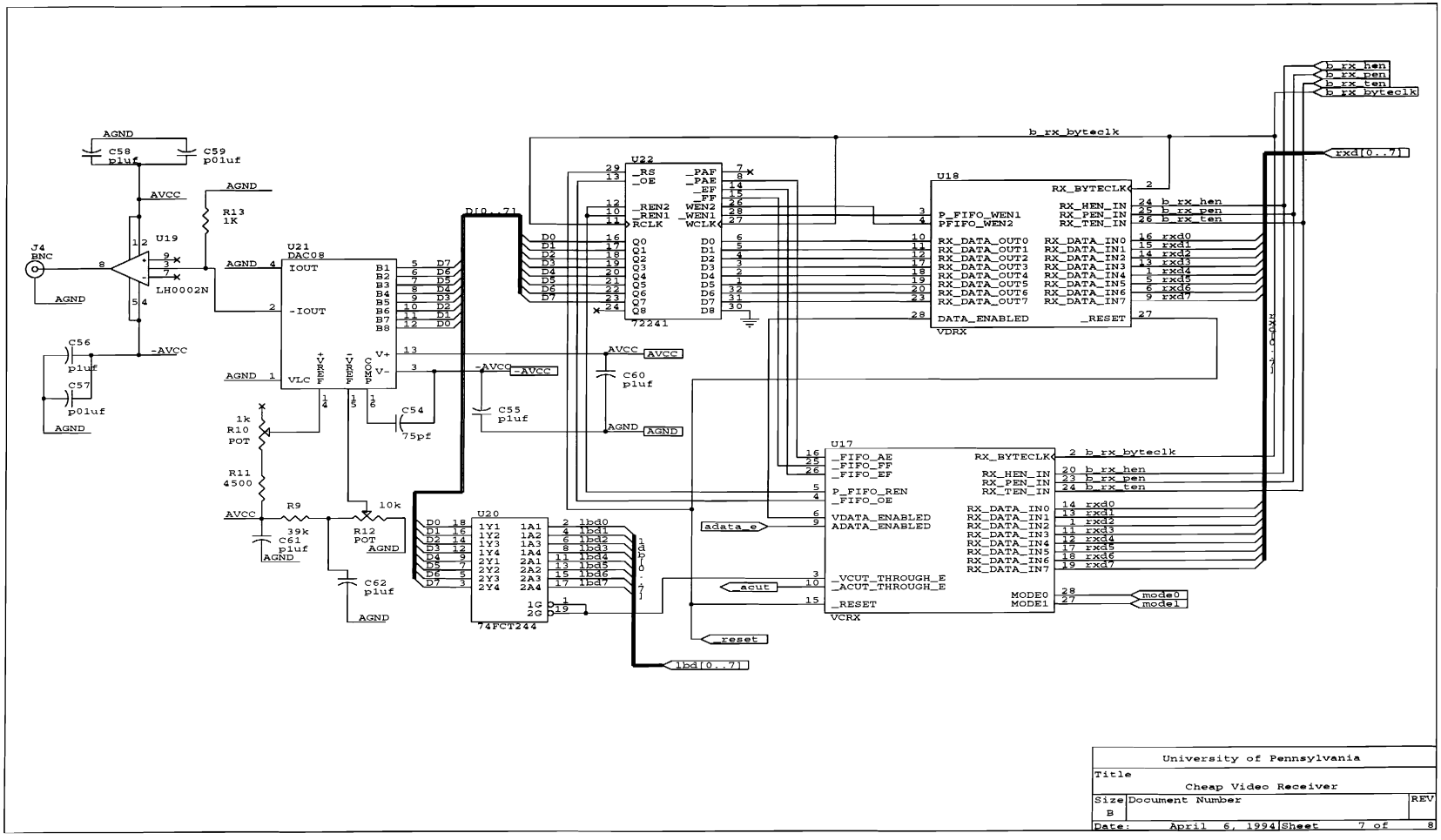
```
    % description of schmitt[] and d1schmitt1 %
    % -- it models a schmitt trigger to detect when video
      input has dropped into the synch range.
      schmitt[]  == 0  (initiation in progress)
      schmitt[]  == 1  (above synch)
      schmitt[]  == 3  (below synch)
    ---%
    schmitt[].clk  = gclock;
    schmitt[].clrn = /rst;
    schmitt[]      = schmitt[] == 0 & rcnt[] == 7 & 1
                     #
                     schmitt[] == 1 & d[] >= Vthres0 & 1
                     #
                     schmitt[] == 1 & d[] <  Vthres0 & 3
                     #
                     schmitt[] == 3 & d[] > Vthres1 & 1
                     #
                     schmitt[] == 3 & d[] <=  Vthres1 & 3 ;

    d1schmitt1.clk  = gclock;
    d1schmitt1.clrn = /rst;
    d1schmitt1      = schmitt[1];

    % description of vshot %
    vshot.clk  = gclock;
    vshot.clrn = /rst;
    vshot       = !d1schmitt1 & schmitt[1];

    % description of ccnt[5..0] %
    ccnt[].clk  = gclock;
    ccnt[].prn  = /rst;
    ccnt[]      = ccnt[] == !0 & !vshot &  !0
                  #
                  ccnt[] == !0 & vshot & 0
                  #
                  ccnt[] == Clamper & !0
                  #
                  (ccnt[] != !0 & ccnt[]!=Clamper) & ccnt[]+1;

    % description of q[7..0] %
    q[]      = iq[];
    iq[].clk = gclock;
    iq[]     = (rcnt[] == 2) &  PAEOffsetLSB
               #
               (rcnt[] != 2) & d[];

    % description of clamp (and zero) %
    % I think that while clamp is held high d[] will be zeroed %
    % if this is not the case then q[]'s description needs to  %
    % to be modified to assure that zeros are written to the fifo %
    clamp       = iclamp;
    iclamp.clk  = gclock;
    iclamp.prn = /rst;
    iclamp      = ccnt[] != !0;

END;
```

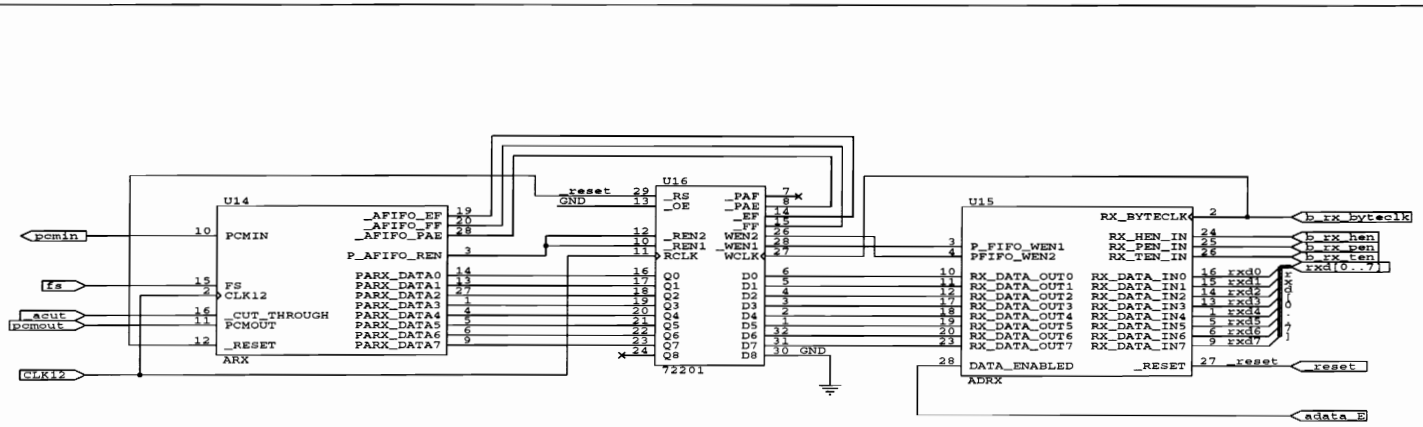
Appendix A: Schematic Diagram



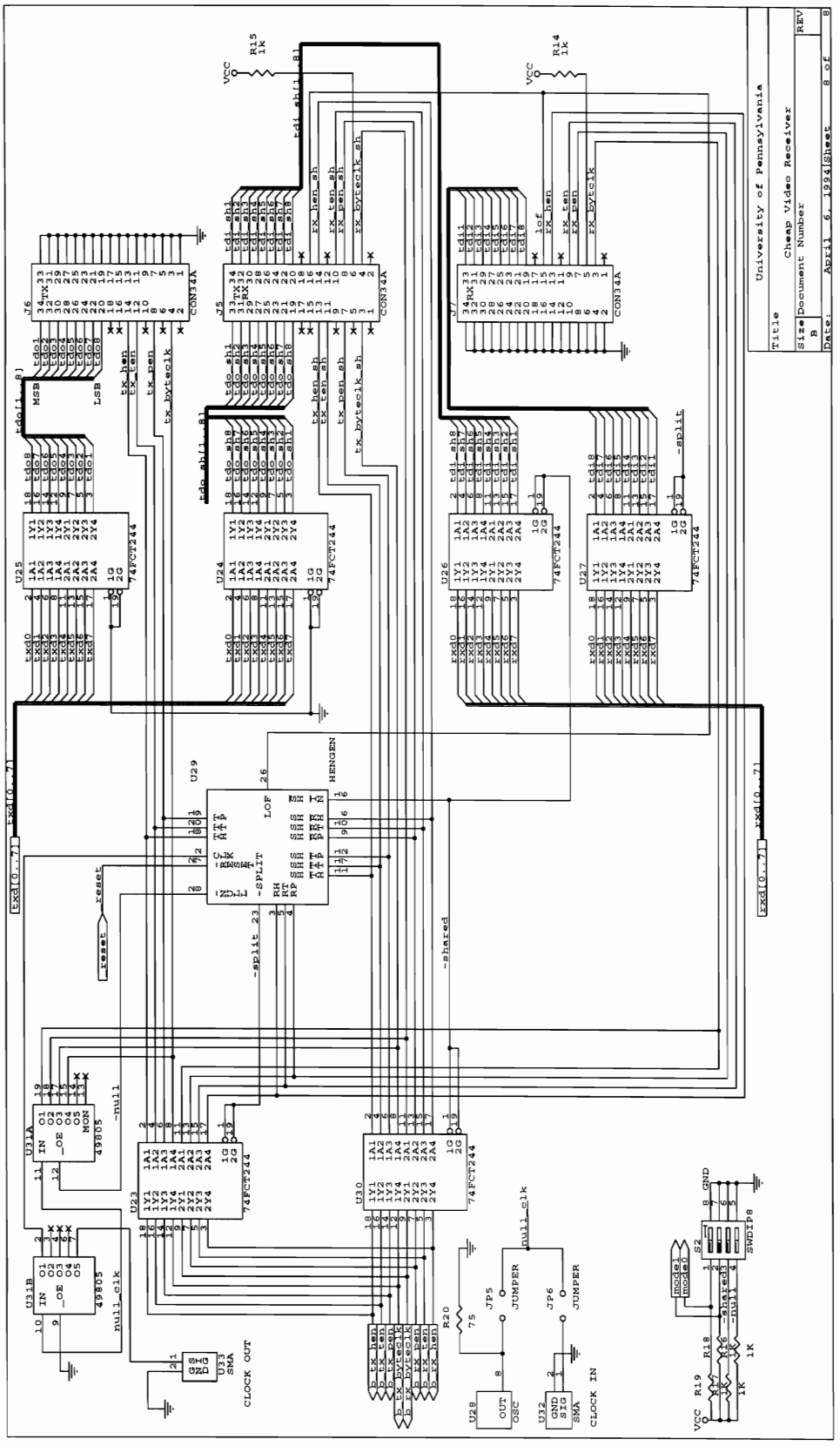

J1
BNC
R4
AGND
75
C43
pluf
R3
50
U6
BT218
VIN
VIN
REF+
VREF
-OE
18 AGND
R2
200
R6
12K
REF-
LEVEL
AGND
R5
10K
VVV
AAA
AAA
GGG
NNN
DDD
CLK
D7
D6
D5
D4
D3
D2
D1
D0
C47
10uf
C44
p1uf
C46
p01uf
AGND
L2
INDUCTOR
C45
pluf
AVCC
U8
CLAMP
EPLDFILT
U9
_PAF
_PAE
_EF
_FF
_WEN2
_WEN1
WCLK
_RS
_OE
_REN2
_REN1
RCLK
72201
b_tx_byteclk
txd[0..7]
lbd[0..7]
reset
I presume that Bill will remove
the J0 and J1 inputs for srcrdr and filter
U7
SRCRDR
b_tx_hen
b_tx_pen
b_tx_ten
audtxfoe
audtxfren
audtxfpae
University of Pennsylvania
Title
Cheap Video Receiver
Size
B
Document Number
REV
Date: April 6, 1994
Sheet 4 of 8


U13
_PAF
_PAE
_EP
_FF
WEN2
_WEN1
WCLK
_RS
_OE
_REN2
_REN1
RCLK
D0
D1
D2
D3
D4
D5
D6
D7
D8
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
72201
_reset
audtxfpae
audtxfoe
audtxfren
b_tx_byteclk
txd[0..7]
txd0
txd1
txd2
txd3
txd4
txd5
txd6
txd7
U12
VBB
VCC
AGND
DGND
_POWER_DOWN
GSX
ANLG_IN
PWR0+
PWR0-
CLK
PCM_OUT
_TSX
FSX
PCM_IN
FSR
DCLKR
129C18N
fixed clk operation
5 -AVCC
-AVCC
AVCC
pcmout
pcmin
U10
PCMOUT
CLK12
FS
_RESET
_LD
_WEN1
PCMTX0
PCMTX1
PCMTX2
PCMTX3
PCMTX4
PCMTX5
PCMTX6
PCMTX7
GND
AUDTX
J2
J3
PHONEJ
JP1
JP2
JP3
JP4
JUMPER A
JUMPER B
JUMPER C
AGND
R8
R7
C48
p1uf
xr
U11
b_tx_byteclk28
CLKIN
CLK12
FS
AUDCLK
clk12
fs
C51 pluf
C52 10uf
C53 p01uf
C50 pluf
C49 pluf
Place this cap next to AGND
University of Pennsylvania
Title
Cheap Video Receiver
Size B
Document Number
REV
Date: April 6, 1994
Sheet 5 of 8

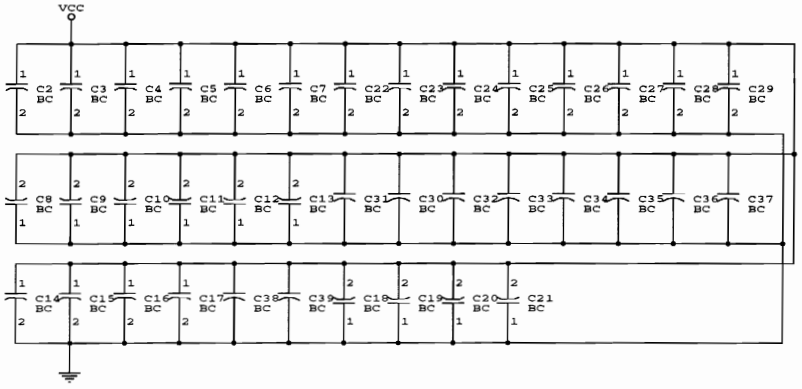

AGND
C58 pluf
C59 p01uf
AVCC
R13 1K
U19
LH0002N
J4 BNC
-AVCC
C56 pluf
C57 p01uf
U21 DAC08
IOUT
-IOUT
VLC
V+
V-
C54 75pf
C55 pluf
C60 pluf
1k R10 POT
R11 4500
R9 39k
10k
R12 POT
C61 pluf
C62 pluf
D[0..7]
U22 72241
_RS
_OE
_REN2
_REN1
RCLK
_WEN2
_WEN1
WCLK
_PAF
_PAE
_EF
_FF
U18 VDRX
RX_BYTECLK
P_FIFO_WEN1
PFIFO_WEN2
RX_DATA_OUT0
RX_DATA_OUT1
RX_DATA_OUT2
RX_DATA_OUT3
RX_DATA_OUT4
RX_DATA_OUT5
RX_DATA_OUT6
RX_DATA_OUT7
DATA_ENABLED
RX_HEN_IN
RX_PEN_IN
RX_TEN_IN
RX_DATA_IN0
RX_DATA_IN1
RX_DATA_IN2
RX_DATA_IN3
RX_DATA_IN4
RX_DATA_IN5
RX_DATA_IN6
RX_DATA_IN7
_RESET
b_rx_byteclk
b_rx_hen
b_rx_pen
b_rx_ten
rxd[0..7]
U17 VCRX
_FIFO_AE
_FIFO_PF
_FIFO_EF
P_FIFO_REN
_FIFO_OE
VDATA_ENABLED
ADATA_ENABLED
_VCUT_THROUGH_E
_ACUT_THROUGH_E
MODE0
MODE1
adata_e
_acut
_reset
U20 74FCT244
1G
2G
lbd[0..7]
mode0
mode1
University of Pennsylvania
Title
Cheap Video Receiver
Size B
Document Number
REV
Date: April 6, 1994
Sheet 7 of 8

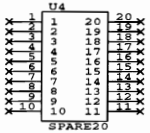


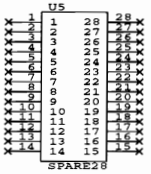
University of Pennsylvania		
Title: Cheap Video Receiver		
Size: B	Document Number	REV
Date: April 6, 1994	Sheet 6 of 8	

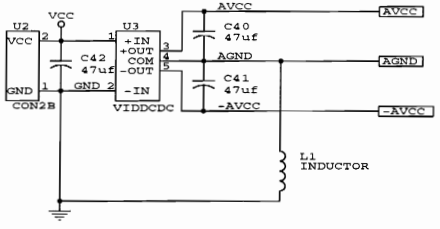

University of Pennsylvania
Title
Cheap Video Receiver
Size
B
Document Number
REV
Date: April 6, 1994 Sheet 8 of 8
VCC
R15
1k
R14
1k
J6
J5
J7
CON34A
tdi_ch1_8
tdo_ch1_8
ex_hen_sh
ex_ten_sh
ex_pen_sh
ex_bytecl_sh
tx_hen
tx_ten
tx_pen
tx_bytecl
MSB
LSB
U25
U24
U26
U27
74FCT244
txd[0..7]
rxd[0..7]
U29
HENGEN
LOF
-split
-reset
-shared
U31A
U31B
49805
-null
null_clk
U23
U30
JP5
JP6
JUMPER
R20
75
U28
OSC
U32
SMA
CLOCK IN
CLOCK OUT
SW1
SWDIP8
R18
R16
R17
1K
VCC
R19

University of Pennsylvania		
Title: Cheap Video Receiver		
Size: A	Document Number	REV
Date: April 4, 1994	Sheet 2 of 8	









University of Pennsylvania		
Title: Cheap Video Receiver		
Size: B	Document Number	REV
Date: April 6, 1994	Sheet 3 of 8	