spy cam
Design Document
Sid Misra, Amit Mehta, Ken Tang

Design Overview Schematic

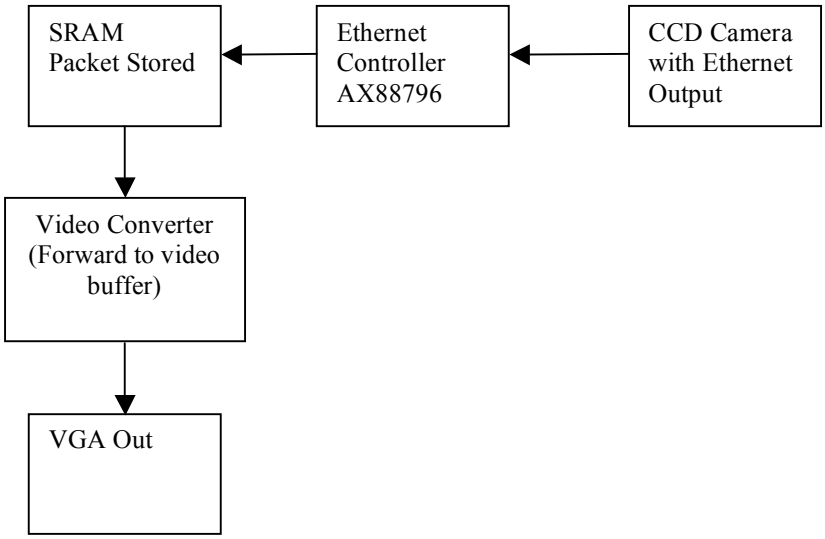

Figure 1. Design schematic

Objective

Our objective is to implement a CCD camera output using the Ethernet protocol. We will use the FPGA to decode the Ethernet packets received from a standard IP camera and present the current image on the camera via the VGA out port on the FPGA.

Details

As the data streams in from the CCD camera, it will reside temporarily in the Ethernet controller's FIFO buffer. This data is then transferred to the SRAM and then transferred to the screen buffer.

The challenging aspect of the project will be to implement a connection oriented TCP protocol to set up a connection between the FPGA and the camera NIC. This will require sending acknowledgements and initial handshake information as per the TCP protocol.

A video peripheral may be needed to convert the JPEG or MPEG4 frames to standard VGA frames. This segment of the project is pending selection of the CCD device and will be explained in detail in the project report.

Components

SRAM
Toshiba TC55V16256J

In our project, we will be using the SRAM chip on the FPGA board to store the incoming packets before they will be displayed on the screen. Unless we are bound by memory constraints we hope to implement 680 x 340 = 300k of memory for the video packets.

Memory

The SRAM consists of 512K of memory.

Ethernet Controller
AX88796

We intend to use the TCP/IP protocol instead of a simpler UDP protocol in order to retain the option of transmitting the frames wirelessly.

Ethernet Packet

We have had the opportunity to examine the Jay Cam design document available at the following URL: http://www1.cs.columbia.edu/~sedwards/classes/2004/4840/designs/jay-cam.pdf

However, our implementation intends to use the TCP protocol as against the connectionless UDP protocol. Please see figure 2 for the header information for the TCP packets.

Ethernet Control Registers

The Ethernet Controller's white paper consists a listing of the registers present on the microcontroller. The document also describes how to set these registers to instruct the Ethernet card to initialize, transmit, receive, and so on.

A listing of these registers and their functions can be found on page 32 and 33 at http://www1.cs.columbia.edu/~sedwards/classes/2006/4840/ax88796.pdf

IP Header

Bit Number

```
1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

Version	IHL	Type of Service	Total Length
Identification		Flags	Fragment Offset
Time to Live		Protocol	Header Checksum
Source Address			
Destination Address			
Options (optional)			

IP Header Contents

Version
4 IP version 4

Internet Header Length
Number of 32-bit words in IP header; minimum value = 5 (20 bytes) & maximum value = 15 (60 bytes)

Type of Service (PreDTRCx) --> Differentiated Services

Precedence (000-111)		000
D	(1 = minimize delay)	0
T	(1 = maximize throughout)	0
R	(1 = maximize reliability)	0
C	(1 = minimize cost)	1 = ECN capable
x	(reserved and set to 0)	1 = congestion experienced

Total Length
Number of bytes in packet; maximum length = 65,535

Flags (xDM)
x (reserved and set to 0)
D (1 = Don't Fragment)
M (1 = More Fragments)

Fragment Offset
Position of this fragment in the original datagram, in units of 8 bytes

Protocol

1	ICMP	17 UDP	57 SKIP
2	IGMP	47 GRE	88 EIGRP
6	TCP	50 ESP	89 OSPF
9	IGRP	51 AH	115 L2TP

Header Checksum
Covers IP header only

Addressing

NET_ID	RFC 1918 PRIVATE ADDRESSES
0-127 Class A	10.0.0.0-10.255.255.255
128-191 Class B	172.16.0.0-172.31.255.255
192-223 Class C	192.168.0.0-192.168.255.255
224-239 Class D	(multicast)
240-255 Class E	(experimental)

HOST_ID
0 Network value; broadcast (old)
255 Broadcast

Options (0-40 bytes; padded to 4-byte boundary)

0	End of Options list	68	Timestamp
1	No operation (pad)	131	Loose source route
7	Record route	137	Strict source route

TCP Header

Bit Number

```
1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

Source Port			Destination Port
Sequence Number			
Acknowledgment Number			
Offset (Header Length)	Reserved	Flags	Window
Checksum			Urgent Pointer
Options (optional)			

TCP Header Contents

Common TCP Well-Known Server Ports

7	echo	110	pop3
19	chargen	111	sunrpc
20	ftp-data	119	nntp
21	ftp-control	139	netbios-ssn
22	ssh	143	imap
23	telnet	179	bgp
25	smtp	389	ldap
53	domain	443	https (ssl)
79	finger	445	microsoft-ds
80	http	1080	socks

Offset
Number of 32-bit words in TCP header; minimum value = 5

Reserved
4 bits; set to 0
ECN bits (used when ECN employed; else 00)
CWR (1 = sender has cut congestion window in half)
ECN-Echo (1 = receiver cuts congestion window in half)

Flags (UAPRSF)
U (1 = Urgent pointer valid)
A (1 = Acknowledgement field value valid)
P (1 = Push data)
R (1 = Reset connection)
S (1 = Synchronize sequence numbers)
F (1 = no more data; Finish connection)

Checksum
Covers pseudoheader and entire TCP segment

Urgent Pointer
Points to the sequence number of the byte following urgent data.

Options

0	End of Options list	3	Window scale
1	No operation (pad)	4	Selective ACK ok
2	Maximum segment size	8	Timestamp

Figure 2. TCP and IP header information

The following registers need to be set for this protocol to work correctly.

IP Header (20 bytes)

Field	Size	Initial Value	Comment
Version	4 bits	"0b0100"	Version 4
Internet Header Length	4 bits	"0b0101"	Length = 5
Type of Service	1 byte	"0b00000000"	
Total Length	2 bytes	"0x217"	537 bytes
Identification	2 bytes	Unique Number	
Flags	3 bits	"010"	x = 0: Reserved D = 1; Don't Fragment M = 0; Less Fragments
Fragment Offset	13 bits	"0000000000000"	
Time to Live	1 byte	TBD	Must change based of error rate
Protocol	1 byte	0x06	UDP = 17, TCP = 6
Header Checksum	2 bytes	Computed	For IP header only
Source IP Address	4 bytes	TBD	Constant value
Destination IP Address	4 bytes	TBD	Set by user using C code
Data	517 bytes	Contains Video	TCP header: 4 bytes Data: 513 bytes

TCP Header (20 bytes)

Field	Size	Initial Value	Comment
Source Port	2 bytes	TBD	Pending More Info
Destination Port	2 bytes	TBD	Choose a free port on client
Sequence Number	4 bytes	ONTHEFLY	Set by C code
Acknowledgement Number	4 bytes	ONTHEFLY	Set by C code
Offset (Header Length)	6 bits	"101"	Set to 5 bytes
Reserved	4 bits	"0000"	Since not using ECN
Flags	6 bits	"000000" Adjusted on the fly	U=0 (Urgent pointer valid) A=0 (Acknowledgement field value valid) P=0 (Push data) R=0 (Reset connection) S=0 (Synchronize sequence numbers) F=0 (no more data; Finish connection)
Window	2 bytes		
Checksum	2 bytes	Computed	For entire TCP segment
Urgent Pointer	2 bytes		

Transmission Details

The receive data must be removed from the receive buffer and moved to the on-chip SRAM. This may be accomplished as in the jay cam project by setting certain control registers and performing DMA operations.

Receive packet data from NIC on-chip RAM. 512 data bytes, 1 positioning byte, 20 TCP bytes, 20 IP header bytes, i.e. 553 bytes.

Command Register

The Command register is an 8-bit register with the inputs [PS1, PS0, RD2, RD1, RD0, TXP, START, STOP]. The following are some common values to which the register will be set.

To activate register and abort current processes	: [00100010] or 0x22
To perform a Remote DMA Read	: [00001010] or 0x0A
To perform a Remote DMA Write	: [00010010] or 0x12
To initiate transmission of a packet	: [00100110] or 0x26

Procedure

Establish connection with camera by exchanging packets using DMA reads and DMA writes.
Send "0x22" to the Command Register to activate the controller.
Send a Remote DMA write operation to transfer data from the FIFO to the on-chip SRAM.
Send a Transfer acknowledgement packet by conducting a remote DMA read.

OPB Bus

The bus is not going to be terribly complicated to set up because we do not have too many peripherals that are awaiting access to the bus.

- The CPU wants to read packets from the Ethernet Controller
- The CPU wants to write packets to the SRAM and Screen buffer simultaneously

Video Converter Outline

Converts JPEG or MPEG4 frames and outputs standard VGA frames that are then sent to the screen buffer and remain there until the buffer is refreshed in the next cycle.

Memory Use

The data received from a 640x480 CCD camera will be about 300K This data will fit on the SRAM and will be concurrently moved onto the screen buffer. This is the largest

resolution we will handle at about 12 fps, we will use a smaller video resolution of 160x120, i.e. 19.2 K per frame.
To store an acknowledgement or synchronization packet on the SRAM we would need: 20 bytes (IP) + 20 bytes (TCP) + 1 byte (positioning) = 41 bytes

Pin Connections for Inter-Peripheral Control

These are the pins set in the system.ucf file to set the Ethernet Controller.

```
# Clock bus
net sys_clk period = 18.000;
net pixel_clock period = 36.000;
net io_clock period = 9.000;
net ICLK period = 30.000;
net FPGA_CLK1 loc="p77"; #use 100 MHz clock (old loc="p77")

# Address bus
net PB_A<0> loc="p83";
net PB_A<1> loc="p84";
net PB_A<2> loc="p86";
net PB_A<3> loc="p87";
net PB_A<4> loc="p88";
net PB_A<5> loc="p89";
net PB_A<6> loc="p93";
net PB_A<7> loc="p94";
net PB_A<8> loc="p100";
net PB_A<9> loc="p101";
net PB_A<10> loc="p102";
net PB_A<11> loc="p109";
net PB_A<12> loc="p110";
net PB_A<13> loc="p111";
net PB_A<14> loc="p112";
net PB_A<15> loc="p113";
net PB_A<16> loc="p114";
net PB_A<17> loc="p115";
net PB_A<18> loc="p121";
net PB_A<19> loc="p122";

# Data bus
net PB_D<0> loc="p153";
net PB_D<1> loc="p145";
net PB_D<2> loc="p141";
net PB_D<3> loc="p135";
net PB_D<4> loc="p126";
net PB_D<5> loc="p120";
net PB_D<6> loc="p116";
net PB_D<7> loc="p108";
net PB_D<8> loc="p127";
net PB_D<9> loc="p129";
net PB_D<10> loc="p132";
net PB_D<11> loc="p133";
net PB_D<12> loc="p134";
net PB_D<13> loc="p136";
net PB_D<14> loc="p138";
```

```
net PB_D<15> loc="p139";

# Control signals
net PB_LB_N loc="p140";
net PB_UB_N loc="p146";
net PB_WE_N loc="p123";
net PB_OE_N loc="p125";
net RAM_CE_N loc="p147";

# Ethernet pins
net ETHERNET_CS_N loc="p82";
net ETHERNET_RDY loc="p81";
snet ETHERNET_IREQ loc="p75";
net ETHERNET_IOCS16_N loc="p74";

# Serial port mapping
net RS232_TD loc="p71";
net RS232_RD loc="p73";
#net RS232_CTS loc="p69";
#net RS232_RTS loc="p70";
```