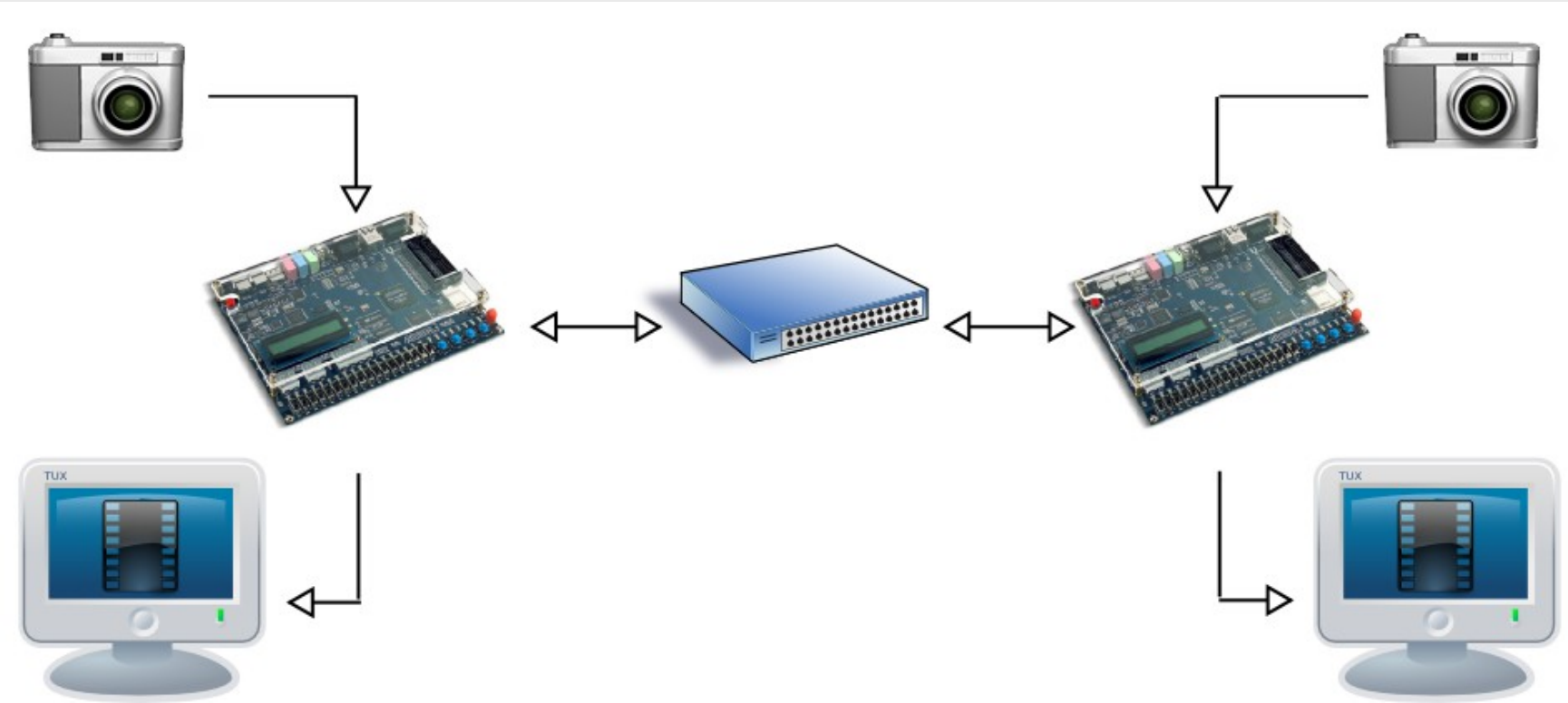
CSEE 4840: Embedded Systems Spring 2009

Video Conference System

Manish Sinha
Srikanth Vemula

Project Overview

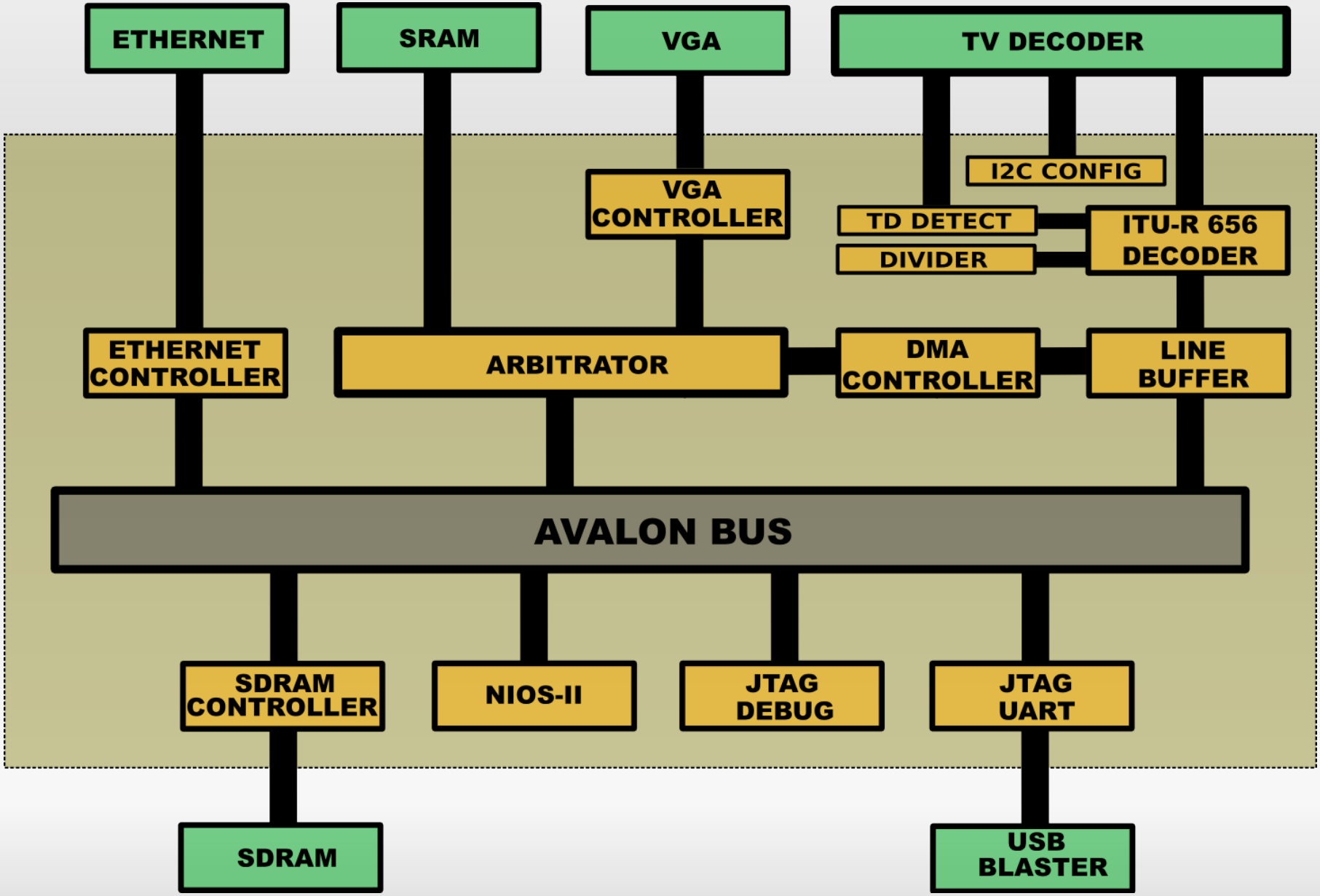


- Top frame of screen will contain the local video
- Bottom frame will contain the network video

Objectives

- Design a video conferencing system by using both hardware and software.
- Display smooth, real-time video for the local video
- Minimize frame loss on network video stream so that it is viewable.
- Building a stable system that uses multiple peripherals

Architectural Design: Block Level Diagram



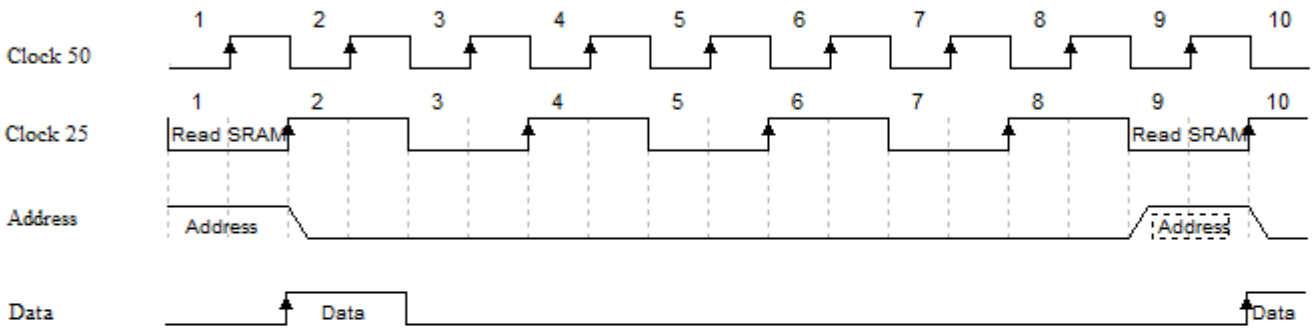
Architectural Design: Linebuffer

- ITU-656 decoder adopted from Terasic reference code.
  - Modified to output only luminance and 4-bits per pixel.
- Linebuffer captures every other pixel of every other line
  - Stored in block RAM
  - 320x240 resolution, 4-bits per pixel = 37.5 KB frame size → quick to transmit over ethernet & small enough to allocate in block RAM. Also, 4-bit pixels give acceptable quality.

Architectural Design: Arbitrator

- There are three resources contending for the SRAM. Their access is prioritized in this order.
  1) VGA: The VGA controller operates on a 25 Mhz pixel clock. It reads the SRAM on every fourth 25 Mhz clock tick because each word in the SRAM contains enough data for four pixels.
  2) Ethernet: The NIOS receives packets, from the Ethernet chip, containing 8 lines. This packet, containing the network video, is sent to the SRAM through the arbitrator.
  3) DMA Controller: The DMA controller writes local video to the SRAM through the arbitrator.
- Why this order?
  - VGA needs pixels or else screen output will become blotted/malformed.
  - Incoming ethernet data is few and infrequent relative to the DMA controller. Switching the ethernet & DMA controller order enables the DMA to take over the SRAM resulting in a poor network video framerate.

Architectural Design: Arbitrator



Architectural Design: DMA Controller

- Motive: decouple the local video from the network video as much as possible (not entirely decoupled because everything has to go through the arbitrator at the end!)
- The DMA Controller moves data from the linebuffer (stored in block RAM) to the arbitrator.
- This yielded much better results than having the NIOS do it. Now the NIOS solely concentrates on ethernet activities.

Architectural Design: DM9000A & NIOS-II Processor

- We optimized the ethernet drivers from lab 2 for speed.
  - Main technique: decrease the delays, use asm(”nop”) instead of usleep(). Increased maximum transmit rate from 80 kb/s to 1.2 mb/s
  - Receive packet routine entirely re-written
- The NIOS reads eight lines from the linebuffer (via the avalon bus) and then ships a UDP packet out to the ethernet. NIOS code and data are stored on the SDRAM. Both are running at 100 Mhz.
- In addition to the data of each line, the linenumer and field are transmitted as well (since both are needed to index into the SRAM)

Experiences & Issues

- Timing diagrams are critical. Knowing what is happening at the cycle level and doing a proper timing analysis are essential to uncovering potential problems.
- Simulations helpful when possible
  - We simulated our SRAM address calculations to ensure their correctness.
- Different clock domains can lead to many problems. Decoupling the 27Mhz, 50Mhz, and 100Mhz clock domains in our system took some careful effort.
- HW/SW tradeoff: time in exchange for speed & precision.

Lessons Learned

- Start the project early.
- Spend as much time as possible during the design phase so problems can be uncovered then rather than later on.
- Do not proceed with implementing the system until a comprehensive timing analysis has been done.
- Use the simplicty of SOPC builder to your advantage; connect components to the avalon bus and use the NIOS to debug hardware.
- Use simulations when possible to unit test components.
- Distribute the work into chunks that can be worked on in parallel.