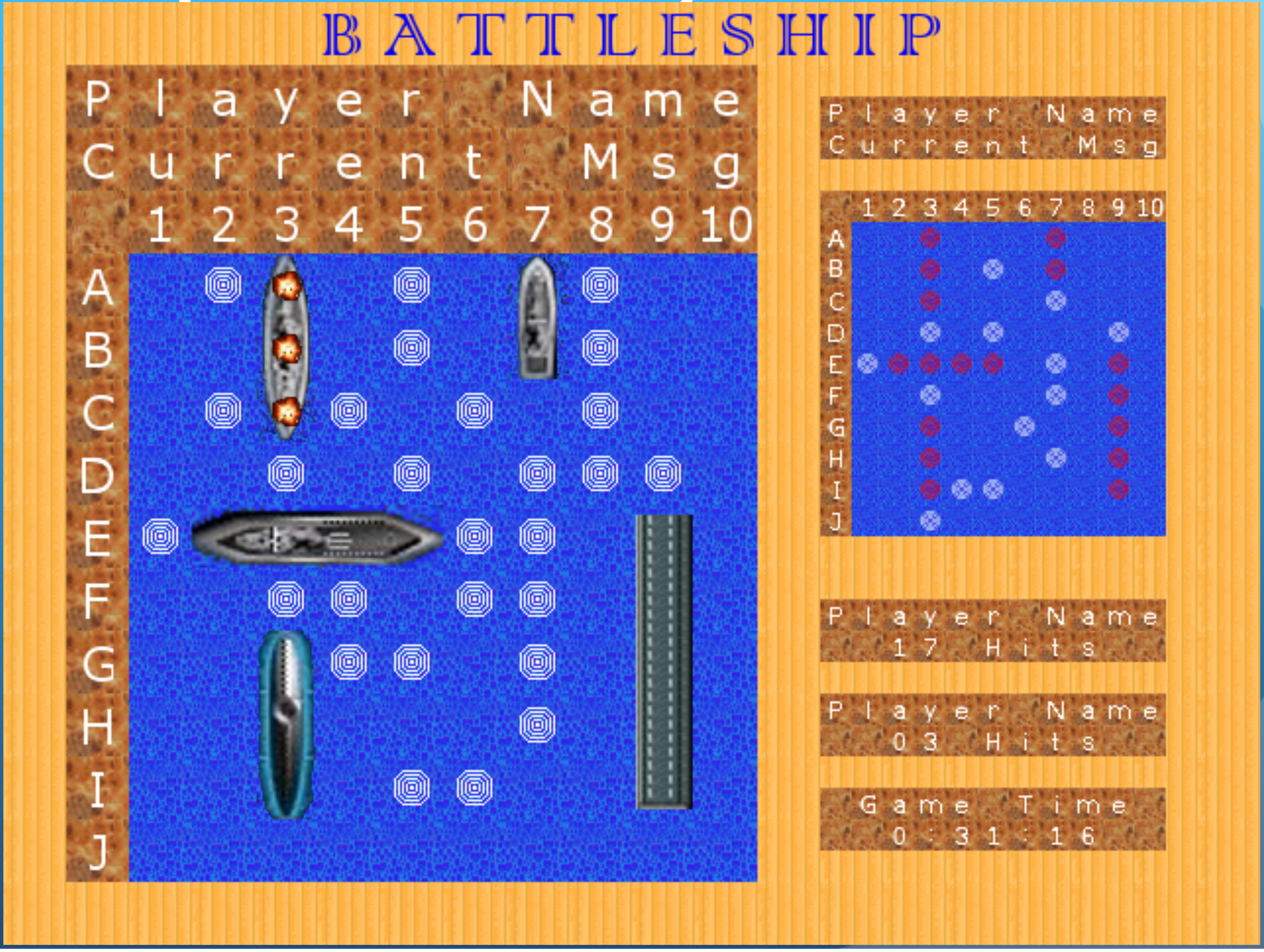
Battleship

Marc Howard, Dan Aprahamian
Apoorva Gade, Shihab Hamati

Game Concept

- Build the game Battleship on embedded hardware
- Build the game Battleship on a computer
- Get the two versions of the game to communicate via Ethernet in order to play a game

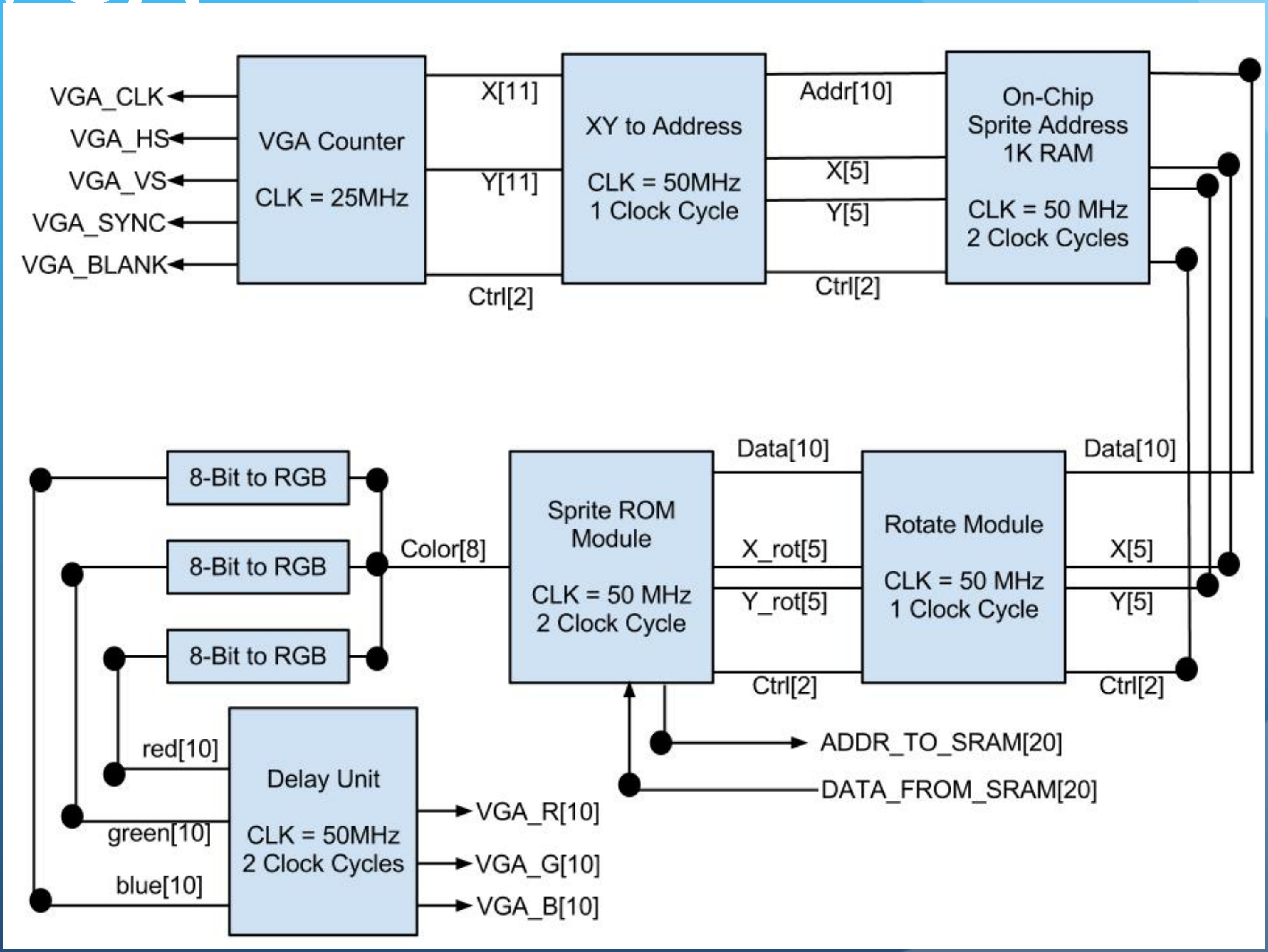
Graphical Layout



Hardware Implemented

- VGA Tile Manipulation
  - Rotation Module
  - Flip Module
  - Invert Module
- VGA Tile Display
- Tile Storage (Memory)
- PS2 Keyboard Driver
- Ethernet Driver

VGA


VGA_CLK
VGA_HS
VGA_VS
VGA_SYNC
VGA_BLANK
VGA Counter
CLK = 25MHz
X[11]
Y[11]
Ctrl[2]
XY to Address
CLK = 50MHz
1 Clock Cycle
Addr[10]
X[5]
Y[5]
Ctrl[2]
On-Chip
Sprite Address
1K RAM
CLK = 50 MHz
2 Clock Cycles
8-Bit to RGB
8-Bit to RGB
8-Bit to RGB
Color[8]
Sprite ROM
Module
CLK = 50 MHz
2 Clock Cycle
Data[10]
X_rot[5]
Y_rot[5]
Ctrl[2]
Rotate Module
CLK = 50 MHz
1 Clock Cycle
Data[10]
X[5]
Y[5]
Ctrl[2]
ADDR_TO_SRAM[20]
DATA_FROM_SRAM[20]
red[10]
green[10]
blue[10]
Delay Unit
CLK = 50MHz
2 Clock Cycles
VGA_R[10]
VGA_G[10]
VGA_B[10]

PS2 Keyboard

- Interfaces with the VGA
- Selects the target tile for attack using the characters

  A-J and 0-9 for the 2 dimensions
- Arrow keys to navigate around the 2X2 array of tiles
- Uses the hardware provided in lab3
- Software returns an appropriate string to the game logic software

Ethernet

- The 2 players communicate
  - through Ethernet
  - using IP/UDP protocols
  - proper checksums
- DE2 Board has IP and MAC of PC hardwired
- PC sends ARP message to correspond IP and MAC of the DE2 Board, to which DE2 responds

Ethernet (Hardships)

- Ethernet not responsive
  - Even after integrating DM9000A.vhd into project and establishing connections in top level .vhd
  - Had to allow 16 cycles of delay to set reset_n signal
- In a loop, first about 100 packets not sending
  - Had to allow a delay after initializing DM9000A controller
- Received bytes had a lot of errors
  - Reason: Ethernet clock synchronization delays
  - Using PLL instead of a logic code to create 25 MHz clock from 50 MHz

Game Programming

- Implemented in Python on computer
- Implemented identically in C on embedded system
- Only changes came in the form of wrapper functions to interact with hardware, which had identical headers on both systems.
- Used Tkinter, PIL, and Socket libraries for Python
- Embedded system goes first. Computer goes second.

Game Logic

1. Get name from user (PS2 Input)
2. Have user place ships
3. Exchange names with opponent (Ethernet)
4. Take a turn – Select a square and fire a shot (Ethernet)
5. Wait for a shot from opponent and respond (Ethernet)
6. Repeat 4 and 5 until one player reaches 17 Hits
7. Ask if player wants to play again. If so go to 2, if not go to 1.

Problems

- VGA
    - On-Chip RAM too small for tile image data - Used SRAM instead
        - Required using SDRAM for program; much more work.
    - Slight image shift depending on monitor used (negligible)
- PS2 Keyboard
    - Repeated signals from keys – solved in software
- Ethernet
    - ARP requests – solved by having board send ARP response
    - Garbage packets from switch - Filtered out in software
    - Switch often lags sending packets (Big problem!)
    - Some packets consistently get a byte garbled (Big problem!)

Accomplishments

- Successfully implemented and integrated all hardware components
- Made game run perfectly as long as Ethernet is not involved (Game startup and ship placement)
- Can demonstrate Ethernet capability with one round of combat by pushing a packet through multiple times (game is robust enough to wait until it receives the right type of packet before continuing).

Demonstration

Questions?