Charles Hastings, Rachid Jeitani

CSEE4840 Embedded Systems

03/22/2011

# *Pong* Design Specification

# User Interaction

## Initialization

At system startup, user is prompted to select an IP address. Since no keyboard is available, mouse will be used to cycle through available addresses. A left-click selects displayed IP address. Restricting selection to 192.168.0.x is acceptable.

Once IP address is selected, rendezvous begins. See below for description of algorithm.

## Gameplay

Each player sits at a DE2 station. Station requires a VGA monitor, PS/2 mouse, and Ethernet connection through hub, switch, or crossover cable to second identical station. Both players control their respective paddles onscreen with mouse.

Master starts with ball near paddle. Player may move paddle up and down and ball will follow. On mouse click, ball is released and bounces off master player’s paddle. Normal Pong-style gameplay begins. Ball bounces off opponent paddle or sides. Opponent scores a goal when ball passes beyond player’s paddle.

## Score Display

Both local and remote score are displayed to users at the top of the screen.

## Ability Scaling

If one player is significantly ahead of the other player, the winning player’s paddle is reduced in size.

## End

Game ends at 11 points. Similar to ping pong, winner must win by 2 points. Status messages are displayed on both stations at conclusion of game.

# Display

## Screen Coordinates

The visible screen area has dimensions of 640x480 pixels. All coordinates used are relative to the visible screen area. Origin is in the upper-left corner of visible screen and is represented by coordinates (0,0).

## Screen Layers

The image displayed to the user is composed of three layers:

1) Game objects (paddles, ball)

2) Foreground bitmap

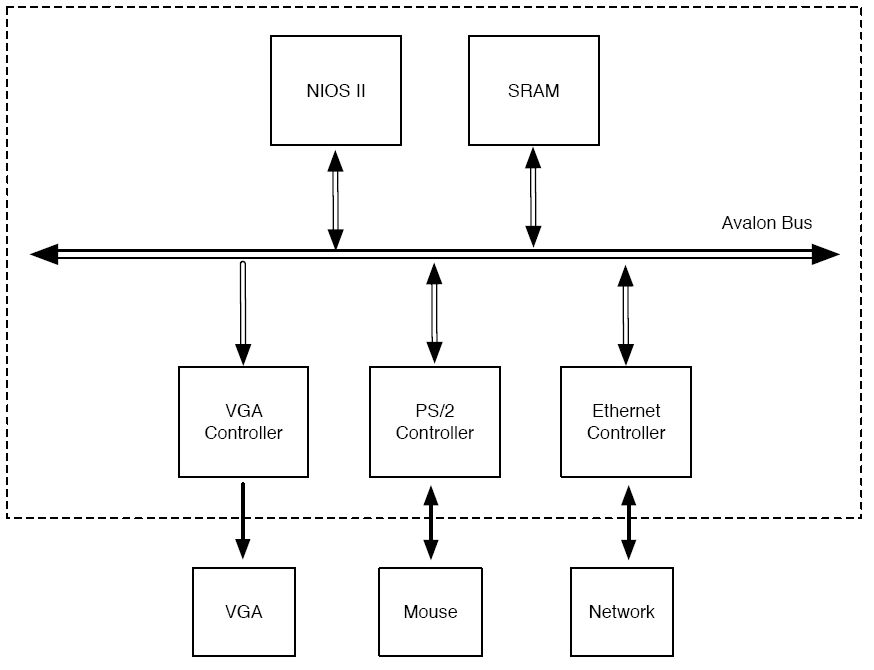
3) Background bitmap tile

The layers are listed in order. If a game object passes into the same location as an active region of foreground bitmap, the game object will take precedence. The foreground bitmap is one color, and where it is not enabled it is transparent. The background tile is an 8x8 repeated bitmap. Where not enabled the background color is black.

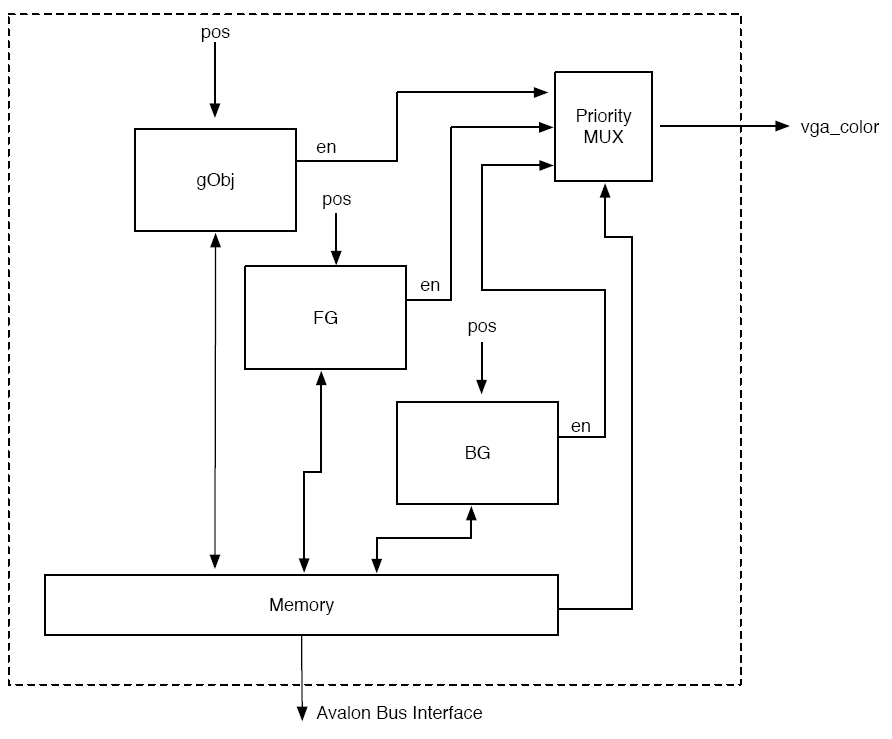
# Block Diagrams

Block diagrams are included for modules which will be implanted using a multi-level hierarchy. Single-level flat modules are not included.

## Top-Level



## VGA Controller



# Memory Maps

## VGA

|  |  |  |  |
| --- | --- | --- | --- |
| **Base Offset** | **Width** | **Name** | **Description** |
| 0x00 | 2 | BALL1X | Ball x position |
| 0x02 | 2 | BALL1Y | Ball y position |
| 0x08 | 2 | PADL1A | Paddle 1 (left) start |
| 0x0a | 2 | PADL1B | Paddle 1 (left) end |
| 0x0c | 2 | PADL2A | Paddle 2 (right) start |
| 0x0e | 2 | PADL2B | Paddle 2 (right) end |
| 0x10 | 2 | BACOLOR | Ball color (low 15 bits, 5 bits/color, R first) |
| 0x12 | 2 | PACOLOR | Paddle color |
| 0x13 | 2 | BGCOLOR | Background tile color |
| 0x16 | 2 | FGCOLOR | Foreground color |
| 0x18 | 8 | BGDATA | Background tile data (8x8 bitmap, 1 row/byte, MSB of each byte is leftmost column) |
| 0x20 | 38400 | FGDATA | 640x480 foreground bitmap (row addressed) |

## Mouse

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Width (Bits)** | **Name** | **Description** |
| 0x00 | 1 | MCTL | Mouser control register (same as first PS/2 byte) |
| 0x01 | 1 | MX | X movement |
| 0x02 | 1 | MY | Y movement |

## Ethernet Controller

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Width (Bits)** | **Name** | **Description** |

*See DM9000A datasheet*

# Communication

## Rendezvous

Master sends out a broadcast packet periodically. Slave, when ready to start a game, replies with its own status packet. Master receives reply, stops sending broadcast packets and game begins.

## OSI Protocol Layering

|  |  |
| --- | --- |
| **Layer** | **Protocol** |
| 2 | Ethernet |
| 3 | IP |
| 4 | UDP |
| 5 | Custom |

## Data Format

UDP packet payload is a fixed size and is used to convey game state information from master to slave and mouse information from slave to master.

struct ball\_t

{

u16 x

u16 y

}

struct padl\_t

{

u16 a

u16 b

}

enum msgtype\_t

{

GAME\_SYN

GAME\_ACK

STATUS\_MASTER

STATUS\_SLAVE

RESET

}

struct message\_t

{

msgtype\_t t

ball\_t ball1

padl\_t padl1

padl\_t padl2

u8 score1

u8 score2

}

# Milestone Deliverables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **M1** | **M2** | **M3** | **Final** |
| *NIOS System* |  |  |  |  |
| Skeleton SOPC project | x |  |  |  |
| SRAM peripheral | x |  |  |  |
| *VGA Peripheral* |  |  |  |  |
| Memory map header file | x |  |  |  |
| Static ball logic/display | x |  |  |  |
| Static paddle logic/display | x |  |  |  |
| Background tile logic/display |  | x |  |  |
| Foreground tile logic/display |  | x |  |  |
| *Mouse Peripheral* |  |  |  |  |
| Memory map header file | x |  |  |  |
| Skeleton module |  | x |  |  |
| Mouse position printf() |  | x |  |  |
| Paddle movement with mouse |  |  | x |  |
| *Network* |  |  |  |  |
| Full C packet structure | x |  |  |  |
| Master/slave rendezvous |  | x |  |  |
| Master/slave status sharing |  |  | x |  |
| *Gameplay* |  |  |  |  |
| Ball bounce off paddle |  | x |  |  |
| Paddle resizing |  |  | x |  |
| Scorekeeping |  |  | x |  |
| *User Interface* |  |  |  |  |
| IP address selection |  |  | x |  |
| Win/lose banners |  |  | x |  |
| Score display |  |  | x |  |
| Background selection |  |  | x |  |
| *Miscellaneous* |  |  |  |  |
| Subversion server | x |  |  |  |
| Report |  |  |  | x |