The Game of Life

Steven Chen, Juan Gutierrez, Vincenzo Zarrillo
{stc2104, jmg2048, vaz2001}@columbia.edu
Embedded Systems Design, CS 4840
May 8, 2007
Overview of the System

■ Based on initial coordinates, outputs to VGA the game visualization (the ‘board’)
  □ White indicates ‘life’ and blue indicates ‘death’
■ Each organism modeled as one pixel on a 256x256 pixel board
■ Generations occur roughly every second
■ Hardware used to update each generation of the game
■ Software (C program) used to pass to hardware the initial conditions of the board
Architectural Design

- VGA Monitor
- VGA Video Port
- FPGA
- vga_raster
- Nios Processor
- Avalon bus
vga_raster Component Design

From Avalon Bus

To VGA Video Port

swap = 0

swap = 1
Overall System

- Nios sends initial coordinates to the ‘Load’ RAM through the Avalon bus
- ‘Load’ RAM contents loaded into RAM 1 (‘current’)
- VGA reads from ‘current’ while updater also reads from ‘current’ and writes to RAM 2 (‘next’)
- ‘next’ and ‘current’ are then swapped
Internal Representation of Game Board

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each ‘cell’ holds 32 bits

- 8 cells X 32 bits = 256 bits total per row
- 8 cells X 256 rows = 2048 \(2^{11}\) cells total in board
Dual-Port RAM - Current

From Updater
- address_a (11 bits)
- data_a (32 bits)
- wren_a (1 bit)

From VGA
- address_b (32 bits)
- data_b (1 bit)
- wren_b (1 bit)

To updater
- q_a (32 bits)
- q_b (32 bits)

To VGA
- clock

Game of Life
Game Logic Implementation

countNeighbors: 0011

From RAM 1
- dataOut1
- dataOut2
- sr_pos
- dataOut3

To RAM 2
- outRegister

34-bit registers
- dataOut1: 1 0 1 ...
- dataOut2: 0 0 0 ...
- dataOut3: 0 0 1 ...

32-bit register
- outRegister: 0 0 ...

currentPositionInOutput

Game of Life
VGA Implementation

- Reads bit by bit and colors pixel accordingly
- After reading bottom right end of the board, updater turns on
Nios Implementation

- Writes 32 bits to each location in RAM
- Random set of numbers or hard-coded set of numbers as initial conditions
Implementation Experiences and Issues

- Necessity of Precise Timing
- Difficulties in deciding on best and easiest implementation of game logic
  - Shift registers, components, etc.
- Writing Initial Conditions into the program
  - Issues with addressing
- Reading from a file in C
Primary Roles

- Steve
  - Updater Implementation
  - Design Document, Final Report, Presentation
- Juan
  - Updater/VGA/Nios Implementation
  - System Integration
- Vinny
  - VGA/Nios Implementation
  - System Integration
- Everyone
  - Design, Debugging, Troubleshooting
Lessons Learned

- Timing Diagrams – Draw them first!
- Test every potential thing that could go wrong as soon as you can.
- The simulator is your best friend
- Think Hardware, not Software
- It’s never too early to start
The Game of Life

Steven Chen, Juan Gutierrez, Vincenzo Zarrillo
{stc2104, jmg2048, vaz2001}@columbia.edu

Embedded Systems Design, CS 4840

May 8, 2007