Lecture #3: The Game of Life

Kunle Olukotun
Stanford EE183
January 13, 2003

Questions?

- Any verilog questions?
- How is the tutorial coming along?
  - Turn in copies of all verilog, copy of "verification" scripts you wrote, corresponding waveforms annotated, FGPA Editor output (use PrtSc and copy to MS Paint), the part of the implementation output that shows the speed and the amount of logic units utilized.
  - Try to print side by side and duplex to save the forests.

Today's Lecture

- I/O
  - VGA
  - Sega Gamepad
- Conway's game of life

VGA Concepts

- VGA monitors are raster devices.
  - Data is sent sequentially for each pixel in connection with the sync signals.
  - 6 Bit color: 2 bits each of reg, blue, green
- VGA_SYNC
  - It gives you a X,Y screen location and you give it a color to display.
  - It controls your logic—think event driven programming.
- Take a look at vgaflag.v
  - Read code carefully
  - Try it out
- Watch timing!!! It must meet 50MHz.
  - Strategies if you don't?

Text Character Generation ROM

- Instead of drawing directly use TCGROM
  - Can be used to create any 8x8 pattern → Sprites
- Implemented as a verilog case statement in tcgrom.

```
·module tcgrom(addr, data);
·  input [8:0] addr;
·  output [7:0] data;
·  reg [7:0] data;
·
·  // A memory is implemented
·  // using a case statement
·
·  always @(addr)
·    begin
·      case (addr)
·        9'h000 : data = 8'b00111100; // %     ****     %
·        9'h001 : data = 8'b01100110; // %    **  **    %
·        9'h002 : data = 8'b01101110; // %    ** ***    %
·        9'h003 : data = 8'b01101110; // %    ** ***    %
·        9'h004 : data = 8'b01100000; // %    **        %
·        9'h005 : data = 8'b01100010; // %    **   *    %
·        9'h006 : data = 8'b00111100; // %     ****     %
·        9'h007 : data = 8'b00000000; // %              %

·        9'h008 : data = 8'b00011000; // %      **      %
·        9'h009 : data = 8'b00111100; // %     ****     %
·        9'h00a : data = 8'b01100110; // %    **  **    %
·        9'h00b : data = 8'b01111110; // %    ******    %
·        9'h00c : data = 8'b01100110; // %    **  **    %
·        9'h00d : data = 8'b01100110; // %    **  **    %
·        9'h00e : data = 8'b01100110; // %    **  **    %
·        9'h00f : data = 8'b00000000; // %
```

Using TCGROM

```
wire   [9:0] YPos;    // [0..479]
wire   [10:0] XPos;   // [0...1287]
//Register the output of the tcgrom

tcgrom tcgrom(.addr({char_selection_q, YPos[4:2]}),.data(tcgrom_d));
always @(XPos or tcgrom_q)
begin
       case (XPos[4:2])
              3'h0 : color = tcgrom_q[7];
              3'h1 : color = tcgrom_q[6];
              3'h2 : color = tcgrom_q[5];
              3'h3 : color = tcgrom_q[4];
              3'h4 : color = tcgrom_q[3];
              3'h5 : color = tcgrom_q[2];
              3'h6 : color = tcgrom_q[1];
              3'h7 : color = tcgrom_q[0];
       endcase
end
```

- Why are low order bits of X, Y not used?

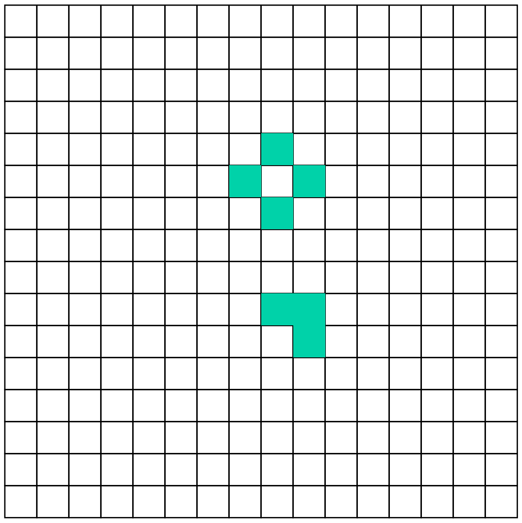
Watch Timing—20ns is not a lot!

- Note the DFFs at many stages in the sample VGA design to break timing constraints.
  - During implementation, the speed is shown.
  - Run static timing tool to see paths that are failing
    - Analyze against auto-generated design contraints (second button from left)
    - View→Floorplanner for Crossprobing
    - Click on paths to see them visually.
- Make sure to preserve hierarchy during synthesis so that more signals are presented. Explicitly instantiating DFFs helps preserve names.

Sega Genesis Gamepad Controller

- Simple interface for human input
  - Functions are multiplexed over the same pins so you need to demux them
  - See EE121 for the full scoop
- How to design digital debouncer?
  - Ignore all edges after the first for a certain period of time.
    - Counter that is enabled on first edge and only accepts inputs after counting for some time.
    - Use slow clock free running counter of suitable size. One slow clock counter per design

Lab #1: Conway's Game of Life



- Cellular Automata emulate laws of nature
- Design will stress
  - Input and output
  - Basic digital design techniques => Cooperating FSMs

Lab #1: Conway's Game of Life

- The Rules
  - For a space that is 'populated':
    - Each cell with one or no neighbors dies, as if by loneliness.
    - Each cell with four or more neighbors dies, as if by overpopulation.
    - Each cell with two or three neighbors survives.
  - For a space that is 'empty' or 'unpopulated'
    - Each cell with three neighbors becomes populated.
- To simplify things, game board wraps around
- Examples:
- http://cgi.student.nada.kth.se/cgi-bin/d95-aeh/get/lifeeng#life
- http://www.bitstorm.org/gameoflife/

Lab #1: Conway’s Game of Life

- The Rules
  - For a space that is 'populated':
    - Each cell with one or no neighbors dies, as if by loneliness.
    - Each cell with four or more neighbors dies, as if by overpopulation.
    - Each cell with two or three neighbors survives.
  - For a space that is 'empty' or 'unpopulated'
    - Each cell with three neighbors becomes populated.
- To simplify things, game board wraps around
- Examples:
- http://cgi.student.nada.kth.se/cgi-bin/d95-aeh/get/lifeeng#life
- http://www.bitstorm.org/gameoflife/

Block RAM (BRAM) stores game state

- Each BRAM is 4Kbits
  - So we can have a 64x64x1 game board
  - Use Coregen and create it as 4Kx1 and use the concatenation of the row and column as the index
  - addr[0:11] = {row[0:5], coll[0:5]};
- Need two of them.
  - One to store current state and one to store next state into.
  - Can’t update in place: need data from all 8 neighbors but rows above have already been processed

Next State FSM

- Next State FSM iterates over the Game board querying each of the 8 neighbors and computes the
- Isn't this terribly inefficient
  - Each full update is ~ 20ns*8*64*64 = 0.6ms
  - So 1500 updates a second
  - We probably want to slow this down.

How do we display?

- When it is refreshing, the VGA needs a data element every clock
  - We could coordinate the game state updates to occur around the VGA screen refreshes
  - Why work that hard? ☺
- Instead use a dual port memory and use one port for the game state updates and one for display.
  - Dual port memories are already implemented in BRAMs so we might as well use it.

Double Buffering

- We need to have the VGA logic switch between the two BRAMs depending on which one is the current complete game state.
  - Standard technique in many areas but specifically graphics.

How do we actually generate the image?

- Split the screen up into 64x64 grid
  - Fine if it is in top left corner
- Could directly draw to the screen
- Could use TCGROM
  - Maybe change one of the characters to a solid block and then an outline block so there is a grid?

Game State Initialization

- To liven things up, use the gamepad to create some specific shape to start with.
- Use the Sega Gamepad to move the mouse around
  - Show current position by flashing the location
  - Key press toggles the state of that location
- To clear, we will reset power and download the *.bit file

Game Play

- “B” Button starts the the iteration
  - Do approximately one iteration a second
- Before pressing “B”, the gamepad moves around a cursor and the “C” button toggles the state at the location of the cursor.
- The buttons chosen so we don’t have to de-multiplex the game-pad.

Optional Fun Things

- Display the number of iterations
- Capability to clear the screen
  - Instead of the cheesy (but perfectly fine) board reset
- Capability to start with a random game board
  - LFSR seeded by counter from powerup
- Fastforward
  - Have the next N game states be computed in rapid succession
    - Perhaps use a third BRAM

Known Interesting Initial States

- Some starting states are more interesting than others
  - Have the initialization of the BRAM be one of them.
  - Have multiple ones and switch between them
- Use Memutils.zip to generate the BRAMs init files.
  - http://groups.yahoo.com/group/xsboard-users/files/

Background Image

- When the game state location is off show a background image
  - Have another 64x64 BRAM storing the image and index it the same way as the game board. If the location is vacant then display whatever is in the background image.
  - we only have 10 4kbit BRAMs

Design Structure

- Hierarchy of cooperating FSMs
  - Master control FSM
  - Gamepad FSM
  - VGA FSMS
  - Board update FSM
  - Game state FSM
- Datapath elements
  - BRAM
  - Counters
  - Registers
- Make VGA and gamepad modules resuable