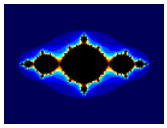
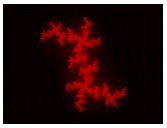


Lecture #7: Fractals!

Paul Hartke
Phartke@stanford.edu
Stanford EE183
April 24, 2002

Lab #1

- Due ***TODAY*** at midnight
  - You can demo anytime before then
- I'll be in the lab around 11pmish for demos
  - Or whenever I get back from the A's game… ☺
- Writeup due next Monday at midnight

Mandelbrot Fractal

- The Mandelbrot set is the set of points in the complex c-plane that do not go to infinity when iterating $z_{n+1} = z_n^2 + c$ starting with $z = 0$. One can avoid the use of complex numbers by using $z = x + iy$ and $c = a + ib$, and computing the orbits in the ab-plane for the 2-D mapping

$$x_{n+1} = x_n^2 - y_n^2 + a$$
$$y_{n+1} = 2x_n y_n + b$$

  with initial conditions $x = y = 0$ (or equivalently $x = a$ and $y = b$). It can be proved that the orbits are unbounded if $|z| > 2$ (i.e., $x^2 + y^2 > 4$).
  - http://www.olympus.net/personal/dewey/mandelbrot.html
  - http://www.jade-leaves.com/mandelbrot_set/index.shtml

Julia Set

- Very similar except for the next state generation except the (a,b)
  - these are constants throughout the calculation
- Sample code for matlab and perl is in
  - http://www.stanford.edu/class/ee183/fractals/
  - Note: the perl ouput looks funny since ascii character dimensions are not proportional

Lab 2

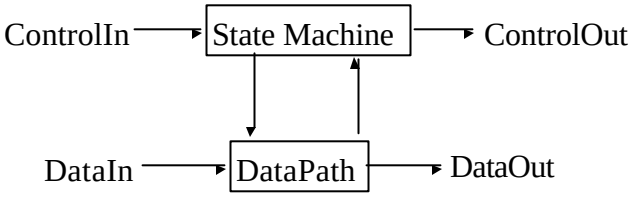
- Display the Mandelbrot Fractal on the VGA monitor
- Use the Sega Controller to select the constant for the Julia set and then draw it
- Calculate both from –2 to 2 on a 64x64 grid with 4 bit color
  - Just like the perl and matlab files
- Demonstration:
  - http://www.unca.edu/~mcmcclur/java/Julia/
- Create a Julia Animation

Game Plan

- Reuse concepts from Lab1
  - Sega Gamepad frontend
  - Dual Port BRAM to calculate into one port and VGA scans the other port
    - Only need 1 DPBRAM because we don't need any state to compute *from*
  - Cursor location to select Julia constant

Separation of Control & Datapath

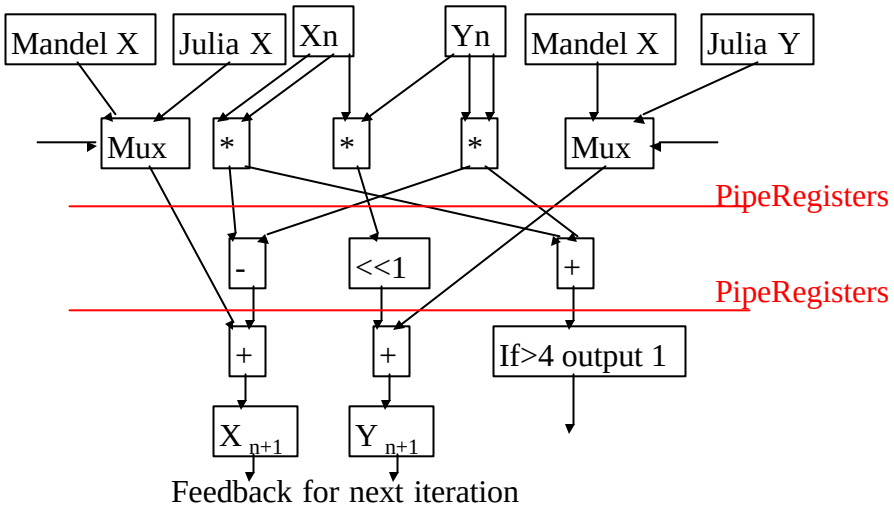
- You have already been doing this
  - Remember the tutorial



Implementation

- Control is random logic
  - Synthesize it
- Datapath is regularly structured logic
  - Array implementations or bitslice
  - Aim for maximum frequency by minimizing wire lengths
  - Most likely will not fit in a single clock cycle
    - Pipeline it!

Fractal Generation Datapath

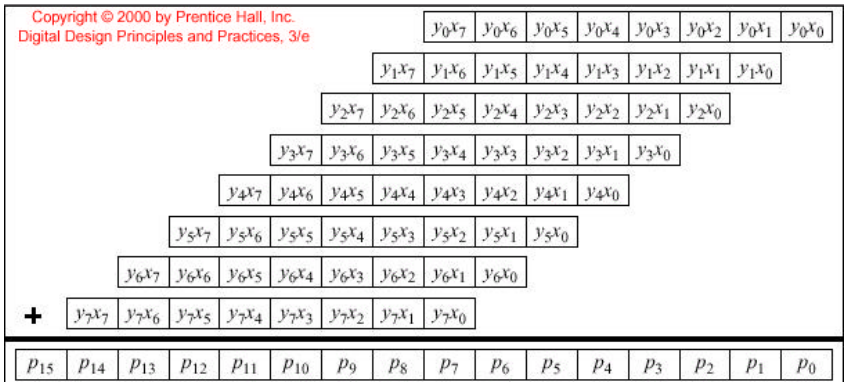


We need multipliers!

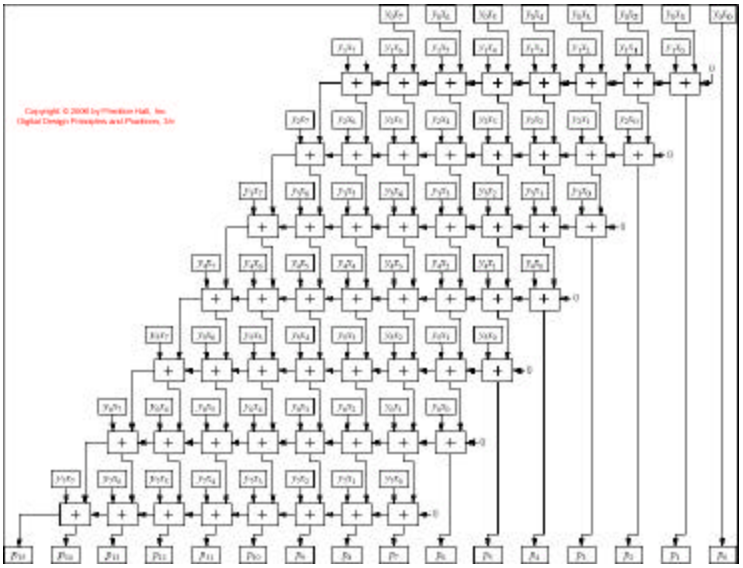
- All other elements in datapath we know how to build..
- We discussed multipliers in EE121…

Multipliers

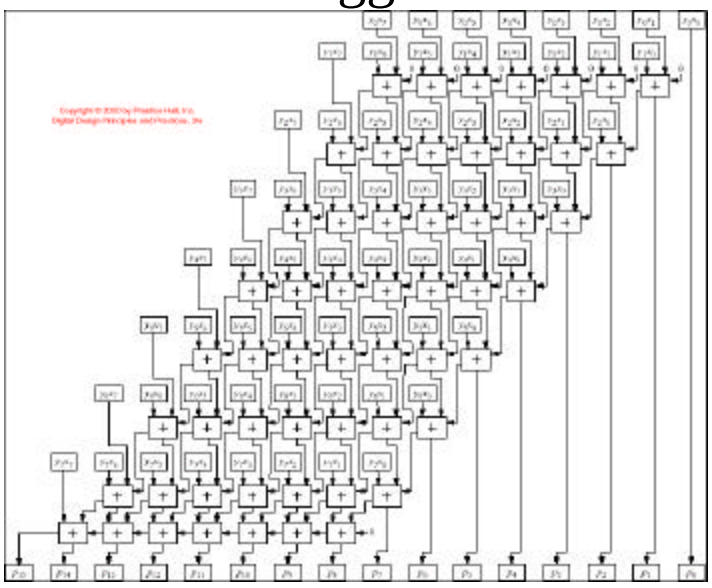
- Repeated Addition



Initial Architecture



More Aggressive



Multipliers Summary

- Lots of clever architectures out there. They all do the same thing—multiply!
- Consider routing delay in addition to logic delay.

Fractions?

- Those were all Integer Multipliers
  - Signed operands in twos complement work fine
- Our algorithm calls for fractional arithmetic
  - Normally implemented as Floating Point Math
    - Very painful
  - So use Fixed Point Math
    - Assume numbers are always in the form: X.Y where X and Y have constant width

Fixed Point Math

- Addition/Subtraction as normal
- Multiplication requires an *arithmetic* right shift after the computation
  - Divide to remove the least significant digits
    - Restore the location of the decimal point
  - Verilog >> is logical shift
    - Construct Arithmetic Shift from conditional (?:)
- Aside: Verilog 2000 has >>>/<<< as arithmetic shifts.

Fixed Point Partition?

- How big should the integer part or fractional part be?
  - Want to minimize these since multipliers grow quickly in size and latency with operand size
  - Don't want them so small that overflow of the integer part occurs (results in aliasing) or that the fractional part has large quantization error
- We stop the loop when magnitude is greater than 4
  - Use that knowledge to approximate size of intermediate operands

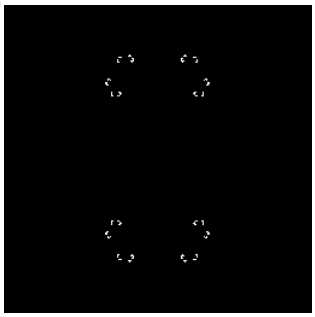
How to Pipeline?

- Create the datapath in a single module
  - Use the coregen multiplier
  - ***Register all inputs and outputs***
  - Implement and then run the static timing tool
  - Add a pipe stage
  - Iterate
- Note that the coregen multiplier has several pipeline options

Create a Julia Set Animation

- How long does each Julia image take to create?
- (64*64*64*7*1/50e6) = 0.036s
- So can calculate them in real time
- Would double buffering help?
  - Not sure…

http://homepages.enterprise.net/scruss/julia_anim.html



Julia Set Animation Constants

- Previous gif: “It's a sequence of Julia sets for the points at 5° intervals around the unit circle.”
  - Why should they be on a circle? What other trajectories would be “interesting?”
- Could calculate these values on the fly
  - Use the Sine-Cosine generator in Coregen
- Or precompute the values and store them in a ROM
- Use the CORDIC algorithm
  - http://www.opencores.org/projects/cordic/