Lecture 7: Lab 2 & Pipelining

David Black-Schaffer
davidbbs@stanford.edu
EE183 Spring 2003

Overview

- Fixed Point
  - Determine your number format from the matlab code (what's the largest number you get?)
  - Map the -2 to 2 plane to a 0 to 63 screen by extracting bits and choosing a binary point
  - Fixed point notation is just a different interpretation (same counting)
- Pipelining
  - If it won't fit in one clock cycle you have to divide it up so each stage will fit
  - The control logic must be designed with this in mind
  - Make sure you need it

Public Service Announcement

- Xilinx Programmable World
  - Tuesday, May 6th
  - http://www.xilinx.com/events/pw2003/index.htm
- Guest Lectures
  - Monday, April 28th
    Ryan Donohue on Metastability and Synchronization
  - Wednesday, May 7th
    Gary Spivey on ASIC & FPGA Design for Speed
  - The content of these lectures will be on the Quiz

Logistics

- Writeup for Lab 1 due tonight at midnight.
- If you haven't finished Lab 1 let us know what's up — getting behind now can be a real problem later!
- Lab 2 Prelab due Friday by 5pm
- Guest lecture next Monday**: SHOW UP! (please!)**

Lab 2 Requirements

- Pipelined calculation of a 64x64x4-bit fractal from -2 to 2 in the real and imaginary planes
- Switch display between Mandelbrot and Julia set
- Julia set constants chosen by the position of a blinking cursor as in lab 1
- You must have at least one of:
  - Animation around an "interesting" path for the Julia set
  - Zoom in/out capability (much cooler)
- Encouraged:
  - Color animation
  - Parallel computation

Key Concepts for Lab 2

- Data path and control path separation
  - Fixed calculation path
  - Standard FSM control
- Fixed-point math
  - Counting is the same, it's just a matter or interpretation
  - 0 to 64 counts the same as 0.00 to 4.00 in binary
- Pipelining
  - What if it doesn't all fit in one clock cycle? (20ns)
  - Split it up into chunks with **pipeline registers** between them
- Parallelism
  - How much can you calculate at the same time?
  - Conflicts in accessing shared resources? (RAM)

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., $x^2 + y^2 > 4$).

Not Really Complicated

Really just iterate over the -2 to 2 real (x) and imaginary (y) planes (i.e., the screen) repeatedly calculating:

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

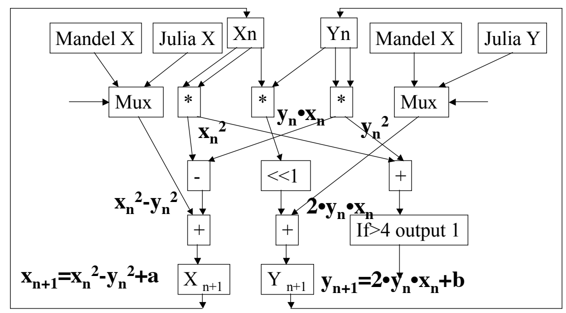
Until $x^2 + y^2 > 4$ or the number of iterations is > 64. Then **the number of iterations it took is what you display** at that location on a 64x64x4-bit display.

Complicated bits

- How do we do the multiplication?
- How do we get the numbers -2 to 2 to map to a screen 64 pixels wide? Fractions!?
- How do we zoom in?
- How do we make it run fast?

EE183 Lecture 7 - Slide 9

Data Path



Feedback for next iteration

EE183 Lecture 7 - Slide 10

Fixed Point Examples

- Twos-complement numbers just work
- It all depends on how you interpret the binary point

3.3 Notation:		**6.0 Notation:**	
000.110	+0.75	000110.	+6
101.100	-2.50	101100.	-20
110.010	-1.75	110010.	-14

- What is this? Shift binary point left 3 places? Divide by 8 when **interpreting**!

EE183 Lecture 7 - Slide 11

Fixed Point Math

- Addition/Subtraction as normal **if you use twos-complement!**
  - Any reason not to use it?
  - None that I can think of.
- Multiplication works as normal if you select the right thing in CoreGen
- 8-bit multiplier takes in two 8-bit numbers and outputs a 16-bit result
  - What do you keep?
  - How big/small is the result?

EE183 Lecture 7 - Slide 12

Fixed Point Partition?

- How big should the integer part or fractional part be?
  - As small as possible to keep the multipliers small and fast
  - Not so small that we loose precision or overflow
- Key insight:
  We stop the loop when magnitude is greater than 4
  - Use that knowledge to approximate size of intermediate operands
  - Run a matlab simulation and figure out the largest value
    - You all know matlab, right?
  - What about zooming?
    - Need more precision?
    - How much?

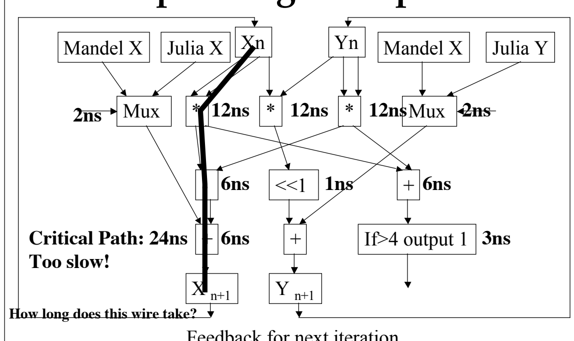
Tricky Bit

- We have a 64x64 pixel screen. We want to map this to -2 to 2. How do we do that?
- Hint:
  - Counting from 0 to 64 goes
    0000000. to **0111111.** in 7.0 notation
  - Counting from 0.00 to 4.00 goes
    000.**0000** to **011**.**1111** in 3.4 notation
  - What's the difference? **Only your interpretation of where the binary point is different.**
  - So 0 to 64 is the same as 0.00 to 4.00,
    but we want -2.00 to 2.00
  - What can you do to **easily** fix that?

Pipelining

- What do we do if the whole data path doesn't fit in 20ns?

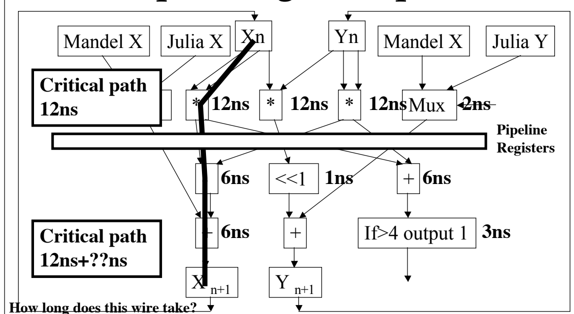
Pipelining Example 1



Pipelining

- What do we do if the whole data path doesn't fit in 20ns?
- Split it up into smaller chunks with registers between them so our **register-to-register** time fits in 20ns.
- Each chunk **does less** but **finishes faster**
- Gets our clock speed up, but takes more clocks (remember the P4 vs. P3 example)

Pipelining Example 2

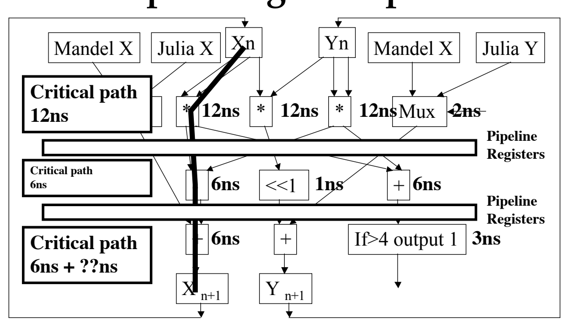


Feedback for next iteration

What next?

- It's still too slow…
- Add more pipeline stages!
- Where?
  - Where ever the critical path is > one clock cycle
  - However, try to keep each stage the same length

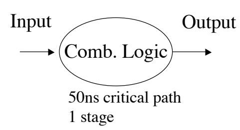
Pipelining Example 3



Feedback for next iteration

Latency and Throughput

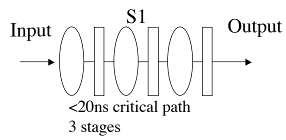
Not Pipelined



Time	**Input**	**Output**
t=0	A1	
t=50ns	A2	A1
t=100ns	A3	A2
t=**150ns**		A3

One result every 20ns after a 2 cycle delay.

Pipelined



Time	**Input**	**S1**	**Output**
t=0	A1		
t=20ns	A2	A1	
t=40ns	A3	A2	A1
t=60ns		A3	A2
t=**80ns**			A3

Key points on Pipelining

- Insert the next data item into the datapath before the previous one has finished
- PipeRegisters keep the computation separate
- Increases utilization for operators
- What is the effect of the algorithm feeding back on itself?
  - Do all iterations have the same number of iterations?
    - How to manage this in Lab 1?
    - More complicated control logic?

Issues with Pipelining

- Throughput
  - It now takes *n* cycles to get a result
  - Can we put in *n* calculations at once?
    - Conflicts? Forwarding? Lab 2 has conflicts…
  - Latency vs. Throughput — **you must understand the needs of your algorithm!**
- Difficulty
  - Non-trivial to implement
  - Make sure you need it!
  - For lab 2, do you need it?

Multipliers

- CoreGen gives you several pipelining options
- Which is best?
  - Depends on your design
- How fast are they?
  - Depends on the size
- Look at the spec sheets or run the timing tools.
  - Remember that routing delay will depend on your final design!

Pipelining Summary

- Make each stage shorter to get a higher clock speed…

but do less in each stage…

so, we need to put multiple calculations through at the same time to get higher performance out of it…

more complicated control and…

data hazards!

EE183 Lecture 7 - Slide 25

Parallelism

- Divide up the problem into multiple problems that can be solved simultaneously
- If they are identical then just instantiate multiple copies of the hardware
- Easy, if there are no resource conflicts

EE183 Lecture 7 - Slide 26

Resource Conflicts

- For Lab 2, multiple calculation units will need to write back to the same RAM.
- When they need to write back at the same time what do you do?
  - Priority scheme: delay one? Which?
  - Avoid starvation. (Round-robin, token)
- Do we care for lab 2?
  - How often will they be competing?
  - **Know your algorithm**. Simulation.

EE183 Lecture 7 - Slide 27

Lecture 6 Key Points

- Fixed-point numbers are the same as regular twos-complement numbers except for how you interpret the placement of the binary point.
- Pipelining increases the clock speed but decreases the amount of work per clock
- Parallelism is easy except for resource conflicts
- Logistics
  - **Lab 1 Writeup due tonight at midnight URL to Joel**
  - Visiting lecturer next Monday – contents will be on the quiz

EE183 Lecture 7 - Slide 28