Lecture 1: Introduction

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

Overview

- Introduction to EE183
  - A continuation of EE121
- Course Goals
  - Larger, faster, more scalable system design
- Course Logistics
  - 4 Labs, 1 Quiz. All labs **must** be completed.
  - Late days cost 10%.
- FSM Introduction
  - Basic review of FSMs

What is EE183?

- EE183 is a continuation of EE121
- More advanced system-level design and integration
- "How do you do a million-gate design?"
  - Good methodologies to handle the complexity
  - Focus on **scalability** and **speed**
  - An EE121 Final Project every 2 weeks

Course Goals

- After EE183 you will be able to…
  - design and optimize a data path,
  - integrate multiple independent FSMs,
  - work with structural verilog,
  - analyze system timing based on logic and routing delays,
  - and get an entry-level chip design job.

What are we going to do?

- More practice with digital logic design
  - Architecture - hierarchical design/decomposition
  - Simulation - what key parts should we simulate?
  - Integration - "the other 90% of the project"
  - Optimization - critical path analysis
- If you do not know FSM design like the back of your hand, this is the time to learn it! (And you will.)

The Labs

- Tutorial: **Basic FSMs and I/O**
- Lab 1: **The Game of Life**
  - VGA output/gamepad editor
  - Memory access, double-buffering
- Lab 2: **Fractals**
  - VGA Mandelbrot/Julia sets
  - Fixed-point math, data path optimization
- Lab 3: **Pipelined CPU**
  - Memory-mapped VGA output
  - Pipelining, forwarding (talk to me if you haven't taken 182)
- Lab 4: **Enhanced CPU/Your choice**
  - DES, filtering, caching, superscalar, what would you like to do?

Note

This is a **design course**. You will be given suggestions and (possibly incomplete) hints as to how you **might** complete the labs, but we expect you to investigate other possibilities and chose the best one.

There are wrong ways to do these labs, but it's up to you to find the best ways.

Basic EE121 Topics

- Boolean Algebra
  - K-maps, SOP, POS, hazards
  - Tools do most of this for you
- Combinational Logic Blocks
  - Decoders, encoders, comparators, adders, multipliers
  - Dividers? Only by a power of two. Why?
- Sequential Circuit Blocks
  - Counters, shift registers, LFSRs, edge detectors
  - Synchronous Why is a ripple counter bad?

Differences from EE121

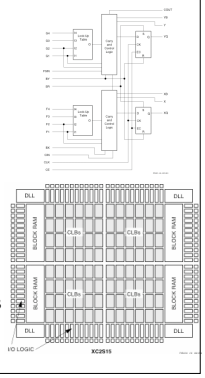
- Larger, Faster Systems
- No Schematic Entry
  - Verilog hardware description language
    - Easier to learn than VHDL (more C-like)
    - Other Stanford classes use it
    - More popular in the Valley
  - Let the tools do a lot of the dirty work
  - But, **ALWAYS** know what logic you are synthesizing. This is key!

Focus: Speed & Scalability

- FPGA Structure
  - Logic is cheap (consider 4 input XOR in VLSI) but wires are very expensive and slow
  - FPGA design speed: routing delay vs. logic delay
- Finite State Machine (FSM) Design
  - Timing and encoding
  - Integration with other FSMs
- Control/Data Path Separation
  - Pipelining
  - Parallel execution
  - Caching, communication, shared resources

FPGA Structure

- Each SLICE
  - 2 Configurable Logic Block (CLB)
    - 4 input LUT, carry chain logic
    - Can be configured as 16-bit RAM
- Routing
  - Local - within SLICEs - very fast
  - 24 lines to each adjacent CLB - fast
  - 96 staggered lines to CLBs 6 away
  - 12 global chip wide/high lines
  - 4 Dedicated clock distribution lines

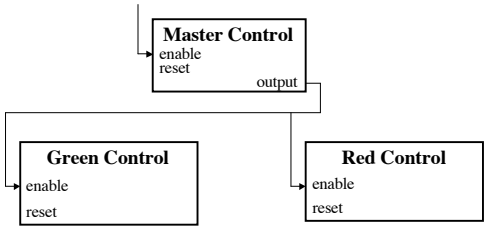


FSM Design - Encoding

State	Assignment 1 S_1S_0	Assignment 2 $S_3S_2S_1S_0$
STATE 0 **A=0**	**00**	**0001**
STATE 1 **A=1**	**01**	**0010**
STATE2 **A=1**	**10**	**0100**
STATE3 **A=0**	**11**	**1000**

- Assignment 1: $A = (S_0+S_1)\bullet(!S_0\bullet!S_1) = (S_0 \text{ XOR } S_1)$
- Assignment 2: $A = S_0+S_1$ Faster, but 4 FFs

FSM Design - Integration



- Include synchronous reset/enables on all FSMs

Course Logistics

- One amazing TA:
  - Joel Coburn (jcoburn@stanford.edu) (In Italy at the moment)
- Webpage www.stanford.edu/class/ee183
  - Read all of the "Tao of EE183" section for getting started information and write-up requirements
  - You **must** subscribe to the ee183 mailing list ASAP (email majordomo@lists, "subscribe ee183")
- 24/7 Lab access to Packard 129
  - Can work from home (who wants to?)
  - We will share the lab with 108b

Important

- **Get ID Card Access & Lab Account**
  - Write your ID#, email, and phone # on the list
  - See me after class for a computer account
- **Start on the Tutorial ASAP**
  - Downloads on the handouts page
  - Due in 1 week
  - 6-8 hours to complete
  - Covers all the basics for EE183

Grading

- Four Labs and one "Quiz"
  - Each lab is half getting it to work and half report
    - 2 points for pre-lab, 18 points for write-up, 20 points for demo
    - Pre-lab due a week before demo, write-up due Monday after demo or 24 hours later if late
  - Quiz will be simple if you have attended the lectures
- Late labs will really hurt your grade
  - It is very important to stay on track. There will be **no** free late days or extensions. The penalty for late demos or write-ups is **10% per calendar day late**. Labs **must** be completed even for 0 points to pass the course.

Any Questions?

- Logistics?
- Content?
- Grading?
- Expectations?

Feel free to email me or talk to me in lab (Packard 129) from 10-11 today.

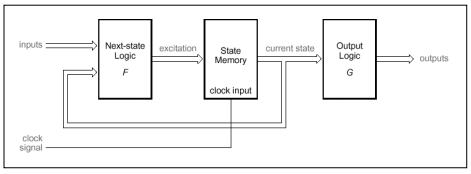
FSM Overview

A Finite State Machine is simply a **state register** that holds the current state and some **combinational logic** which calculates the next state and outputs based on the current state and the inputs.

- A feedback system which updates on each clock

FSMs

- Moore: output based only on current state
- Mealy: output based on current state and inputs
- Affects the number of states and the speed



State Machine Design

1. Determine inputs/outputs
2. Determine machine states
   (Mealy vs. Moore)
3. Determine machine flow
   (state/bubble diagram)
4. State assignment
   (affects speed. How?)
5. Implementation
   (Verilog case statement)

Lecture 1 Key Points

- EE183 is a **fun/hard lab-based course** which will teach you a lot about **digital system design**
- 4 Labs, half credit for implementation and half for write-up. **Late days really hurt**
- **FSMs are crucial** to this course and not really that complicated

- Logistics
  - Get lab access/computer account
  - Subscribe to mailing list and **start the tutorial**
  - See you Friday