Understanding Engineers #1

- The graduate with a Science degree asks, "Why does it work?"
- The graduate with an Engineering degree asks, "How does it work?"
- The graduate with an Accounting degree asks, "How much will it cost?"
- The graduate with an Arts degree asks, "Do you want fries with that?"

Understanding Engineers #2

- MS CS -- Soft-ware
- MS EE -- Hard-ware
- MBA -- Un-a-ware
- MFA -- No-ware

Midterm II

- THIS Thursday, 22 March (that is TWO days from today!), 2:10 -- 3:30, CS 150 Lab
- Lectures 10, 11, 12, (no lecture 13!), 14, 15, 16; Labs #4 and #5 (Debugging/Logic Analyzers) + Checkpoints #0 and #1 (SDRAM + Video Encoder)
- Don’t forget: Spring 05/Fall 05 exams are on-line!
- 5 x 10 point questions, mostly design-oriented
- Closed book, open crib sheet: PENCIL, not pen!
- Two review sessions: Tu 8 PM and W 8 PM in the lab
- NOTE: Discussion sections and lab lecture cancelled this week

Sequential Logic Implementation

- Models for representing sequential circuits
  - Mealy, Moore, and synchronous Mealy machines
  - Verilog specifications for state machines
- Finite state machine design procedure
  - Deriving state diagram from word specifications
  - Deriving state transition table
  - Determining next state and output functions
  - Implementing combinational logic

SDRAM Memory Controller

- Static RAM Technology
  - 6T Memory Cell
  - Memory Access Timing
- Dynamic RAM Technology
  - 1T Memory Cell
  - Memory Access Timing
- Theory in lecture, but practical detailed memory system organization and timing in Lab Checkpoint #0

Two-way Video Conferencing Project

- Project Concept and Background
- SDRAM Controller (Checkpoint #0)
- Video Encoder/Display System (Checkpoint #1)
Videoconferencing System Concept

Register Transfer

C → A
Sel → 0; Ld → 1
C → B
Sel → 1; Ld → 1

Clk
Sel
Ld

Register Transfer

Point-to-point connection
- Dedicated wires
- Mixes on inputs of each register

Common input from multiplexer
- Load enables for each register
- Control signals for multiplexer

Common bus with output enables
- Output enables and load enables for each register

State Machine Implementation

- Alternative controller FSM implementation approaches based on:
  - Classical Moore and Mealy machines
  - Time state: Divide and Counter
  - Jump counters
  - Microprogramming (ROM) based approaches
    - horizontal microcode
    - vertical microcode

Computer Organization

- Computer design as an application of digital logic design procedures
- Computer = processing unit + memory system
- Processing unit = control + datapath
- Control = finite state machine
- Inputs = machine instruction, datapath conditions
- Outputs = register transfer control signals, ALU operation codes
- Instruction interpretation = instruction fetch, decode, execute
- Datapath = functional units + registers
- Functional units = ALU, multipliers, dividers, etc.
- Registers = program counter, shifters, storage registers

Time State (Divide & Conquer)

Time State FSM

Most instructions follow same basic sequence
Differ only in detailed execution sequence

Time State FSM can be parameterized by opcode and AC states

Instruction State: stored in IR(15:14)

Condition State: stored in AC(15)
### Jump Counters

**Pure Jump Counter**

- Logic blocks implemented via discrete logic, PLAs, ROMs.

**Hybrid Jump Counter**

- Load inputs are function of state and FSM inputs.

### Branch Sequencers

**4 Way Branch Sequencer**

- Current State selects two inputs to form part of ROM address.
- These select one of four possible next states (and output sets).
- Every state has exactly four possible next states.

### Vertical Microprogramming

**Branch Jump**

- Compare indicated signal to 0 or 1.

**Register Transfer**

- Source, Destination, Operation.

- Vertical format: \((14 + 4) \times 64 = 1152\) ROM bits.
- Horizontal format: \((14 + 4 \times 2) \times 16 = 512\) ROM bits.

- 10 ROM Bits.
Vertical Programming

Controller Block Diagram

Design/Reverse Engineering

- Design Procedure: Specification → Abstract Design → Concrete Implementation
  - E.g., “What the state machine is supposed to do” to state diagram to jump counter implementation
- Reverse Engineering: Concrete Implementation → Abstract Design → Specification
  - E.g., Jump counter implementation to state diagram to “what the state machine is supposed to do”