Embedded System Debugging and Diagnostics

EE4380 Fall 2001

Class 11

Pari vallal Kannan

Center for Integrated Circuits and Systems

University of Texas at Dallas

Basics

- What to do when your embedded system design is not working as expected ?
- There is no cosmic conspiracy against you !
  - Most problems have simple causes and solutions
- Know the basics about test equipment
- Know the electrical properties of the components in your system
  - Absolute maximum ratings of currents and voltages
  - Maximum frequency of operation
  - ESD

Types of problems

- Always present (reproducible)
  - Dead board
  - No activity of a certain subsystem (display, keypad, etc)
- Erratic (sometimes irreproducible)
  - Happens occasionally
  - Usually difficult to debug and demands the most skill
- Hardware problems
- Software prolems

Test Equipment

- DMM, logic probe
  - Continuity tests, voltage measurements
  - Most trouble-shooting and fixing happens with these
  - Cheap, highly portable, versatile
- CRO
  - For checking fast changing signals
  - Quick check on activity of important pins on clocks, processors, memory, decoders, chip_selects etc
- Logic Analyzer
  - For detailed analysis of a number of signals simultaneously over a time-span
  - For debugging data accesses, DMAs, interrupts, I^2C, etc
  - Identify and trouble-shoot timing violations (setup and hold times) etc.
  - High $$$

Problem Domains in Emb. Systems

- Power related
  - Insufficient drive, switching induced noise, brown outs
  - No power at all. Disconnected or dry solder joints on Vcc, GND pins
- Design related (schematic)
  - Bad netlist, wrong connections, wrong pin-out
- Design related (logic and timing)
  - Poor logic design (state machines, decoders, address generators, PLDs, etc)
  - Timing violations (setup and hold times, memory accesses, DMAs, interrupts, etc)
- Manufacturing related
  - General open circuits and shorts
  - Dry solder, bad wire-wrap,stray metal pieces, bent pins etc
- Random

Trouble-shooting – Broad Outline

- Analytical thinking
  - Think about possible “reasons” that can cause the problem seen
  - Check the input to the system – your design and its schematics (remember GIGO)
- Problem domain identification
- Eliminate hardware issues. Then it’s the software’s turn

Hardware Debugging

- Check power and gnd connectivity on all Ics and discretes
- Check clock connectivity
- Check if the processor is out of RESET
- Check for activity on signals known for continuous activity
  - 8051 – ALE, PSEN, AD[], A[]
  - Memory – address lines and data lines (LSBs are best)
  - All ICs – Chip selects, WR and RD
- Check voltage levels on all transceivers
  - RS232 interfaces (MAX232s), transducers etc
- Check HOST settings
  - Port settings (COM, LPT), etc

Hardware Debugging - Elimination

- Power OK ?
- Clock OK ?
- Processor OK ?
  - ALE, PSEN, RD, WR etc
- Memory OK ?
  - Address and data lines, Chip-selects
- Peripherals OK ?
- Not done yet ?
  - Timing analysis (data sheets, LA etc)
  - Analyze data transfers (LA)

Real world scenario 1

- Pentium Motherboard
- Symptoms
  - Motherboard does not boot
  - CPU fan ON
  - Everything else fine (VGA, memory, etc)
- Diagnosis
  - Turn on the board and feel the CPU
  - CPU cold : maybe no power ? But Fan is on. But fan takes 12V. CPU uses 5V. Maybe 5V supply is dead
  - Check 5V pin on the PS. Dead.
  - Use another PS and turn ON the board
  - Board boots fine !

Scenario 2

- 8051 board not communicating with Host
- Symptoms
  - 8051 board powers up.
  - No communication with host. Host settings fine
- Diagnosis
  - Check power for all ICs. Passed
  - Check ALE and PSEN activity. Passed (8051 fine and out of RESET)
  - Check voltage levels on MAX232 pins. Wrong voltage on Rx pin.
  - Check serial cable connectivity. Monitor Rx on CRO while punching keys on the Host terminal. No activity. Possibly, OC on the Rx pin/header.
  - Replace serial port header. Communication successful.

Software Issues

- Embedded systems are highly affected by the software ! (duh)
- Software based diagnostics running on the system are indispensable
- Bad code
  - Wrong programming of control registers
  - Wrong assembly. Corrupt PROMs
  - Code in infinite loop with no termination condition
- Address issues
  - Mismatch between hardware's memory map and programmer's memory map
  - Timing violations in code for DMAs, LCDs, ports etc

In system Diagnostics

- Very simple code snippets included in the PROM (along with the system's main code)
- Simple tests
  - Light up a status LED
  - Display register contents on a 7seg LED display
  - Put out a string on serial port
- Detailed tests
  - Memory test routines (bit walk, AA55 tests, etc)
  - Peripheral tests
    - Tests for keypad, LCD, ADC/DAC, etc
  - Host communication tests
  - Register and stack read write tests
  - DMA and interrupt tests
- Effective debugging involves using diagnostics and hardware tests