The Art of Debugging

Comp.dsp July 28 - Aug. 1 2004

Mike Rosing

BeastRider, LLC

Overview

- Introduction
- Basic Equipment
- Basic Methods
- Emulators
- Hardware - Software Boundary

Introduction

Debugging as “art form” - Instinct derived from intense practice and concentration.

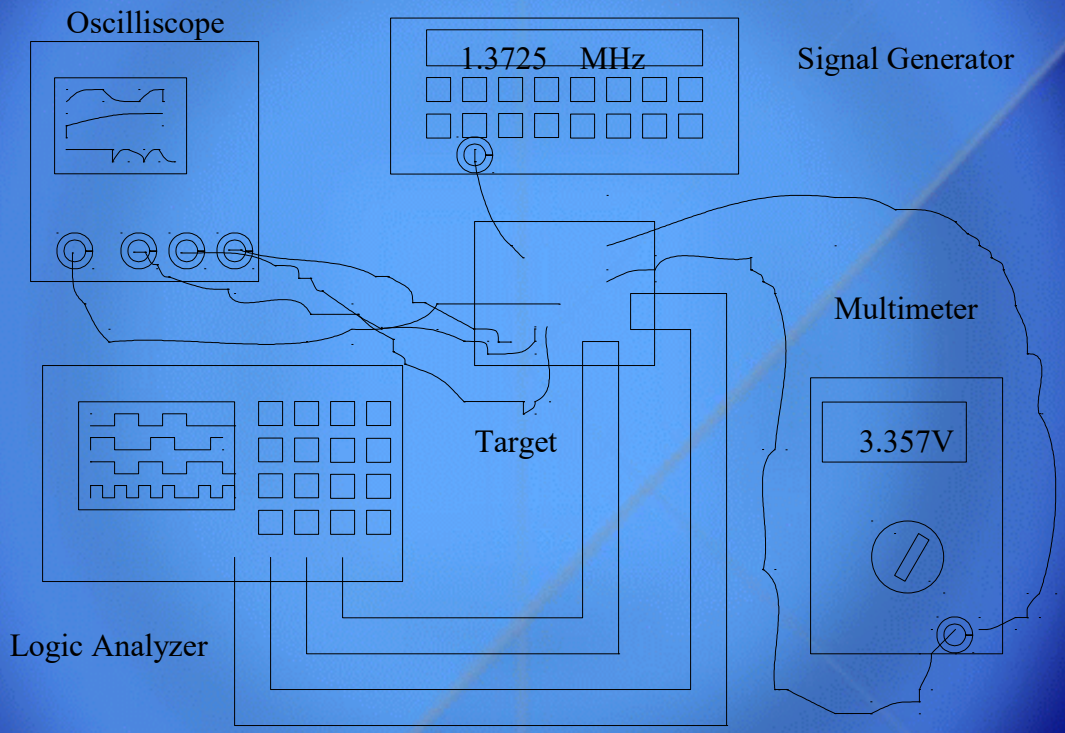
Know your equipment - meters, scopes, analyzers, emulators and generators.

Know your limits - Knowing what you don’t know can help you figure out the next most important thing to learn.

Basic Equipment

- Multimeter - DC & AC Volts & Amps, Ohms, microfarads and frequency
- Oscilloscope - up to 4 lines - essential for any real work
- Logic Analyzer - 16 to 128 or more logic lines simultaneously
- Signal Generator - tones, pulses, complex waves for simulating

Basic Tools


Oscilliscope
1.3725 MHz
Signal Generator
Multimeter
3.357V
Target
Logic Analyzer

Basic Equipment

Computer Controlled

- In Circuit Emulator – for low speed processors
- JTAG emulator – serial link for high speed processors

Basic Methods

Making Things Work

Start with hardware.

Number of logic states is 2^n. For most processors that's on the order of 2^{200}.

Software states:

If memory width is w bits, and memory depth is M words then number of software states is 2^{w*M}. For a SHARC w=48 and M = 64k, so number of states is $2^{3.1e6}$.

Use 3 lines of assembler - beat on hardware first!

Basic Methods

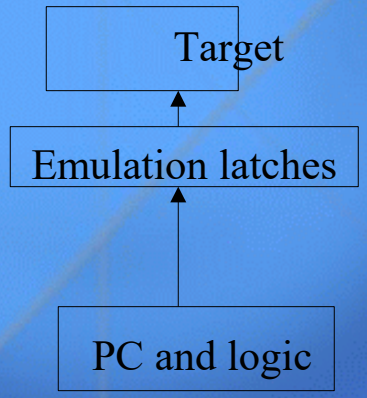
Essense of Attack

O’scope and analyzer - is logic input correct?
Are analog levels as expected?
Are test signals in same ranges as real signals?
Is output logic reasonable?
Use emulators to jam hardware into fixed states, measure DC system first.
Use very small software routines to ensure hardware AC states are reasonable.

Emulators

Why is it called an emulator?

For low speed processors, connector filled chip socket and logic circuits emulated what processor did.



Emulator

Real Emulator

Instead of processor - “target” is a socket in the prototype. Socket is connected to logic gates.
Simulation of internal states takes place in a PC.
PC sets latches (registers and gates) so pins in socket act like real processor.
Does not exactly duplicate timing, but usually is pretty close.

Emulators

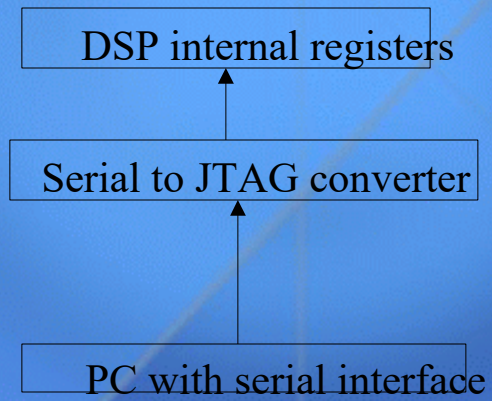
JTAG Version

Joint Test Access Group 1149.1 (IEEE)
Hardware test initially - put all pins on a chip into specific states.
Allows custom commands - used by many manufacturers to set internal states.
Expanded to include debugging capability, read all internal registers, memory and take control of processor.

Emulators

JTAG version

At high speed latches could not keep up. Comparitors in each chip with serial line interface replaced emulation and allow full speed testing.



Emulators

What really happens

The JTAG circuits are independent of the processor.

When a match occurs in one comparitor, it generates an interrupt which halts the processor.

The PC probes the JTAG circuit for the halted state because it is an independent view port into the processor.

Once the processor halts, the JTAG port can change internal states.

Breakpoints

Address Example

```
Jump someplace (db);  <------ compare matches here
                r0 = r2 + r3;
              r1 = dm( i0, m1);
                        :
                        :
            someplace:
                  call dangit;

                                        .
                                        .
                                        .
                                        .
dangit:  r4 = r1 - r0;    <------- processor PC halts here
```

Breakpoints

Hardware Types

- Instruction
- Program Memory read / write
- Data Memory read / write
- External port read / write

Breakpoints

Software

Fixed location of nops - flushes pipeline.
Real instruction replaced with call to fixed location.
Hardware break set to fixed location.
Debugger software displays correct code from list after poping program counter stack.

Advantage - get as many instruction breaks as you want

Disadvantage - uses up program counter stack space.

Hardware-Software Debug

Iteration

Sub component interaction can be found by adding or subtracting one component at a time.
System level interactions with specific inputs are hard to simulate - real world tests need to be captured and used with sub component elimination.
Not just trial and error - each system level bug needs to be isolated by careful observation of each trial.

Hardware-Software

Binary Search

Pick specific blocks and wipe them all out - does bug go away or just morph?
Put half of all block back in - observe behavior and decide if anything has changed.
Proceed to find blocks that cause interaction problems by replacing functions with nops - same number of cycles to keep timing but nothing done.
Instinct helps a lot - but so does using as many signals as possible.

Hardware-Software

Bit Flipping

Have several lines (if possible, sometimes one will do) which flip when you enter specific routines.

Use logic analyzer or o'scope to learn where problems happen, or when specific events (like interrupts) occur.

Combined with binary search and iteration, both timing problems and module interaction can be observed.

Using overall picture, JTAG interface becomes more powerful and pin pointed.

Summary

Do hardware first
Do software in subsections second
Do system level combinations iteratively
Know your tools, know your target, know your limits
and every project you work on will be successful.