

HARDWARE AND POWER MANAGEMENT

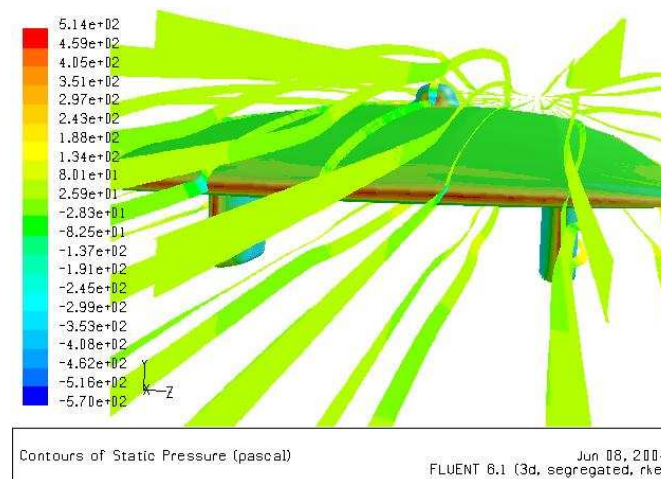
David Snowdon

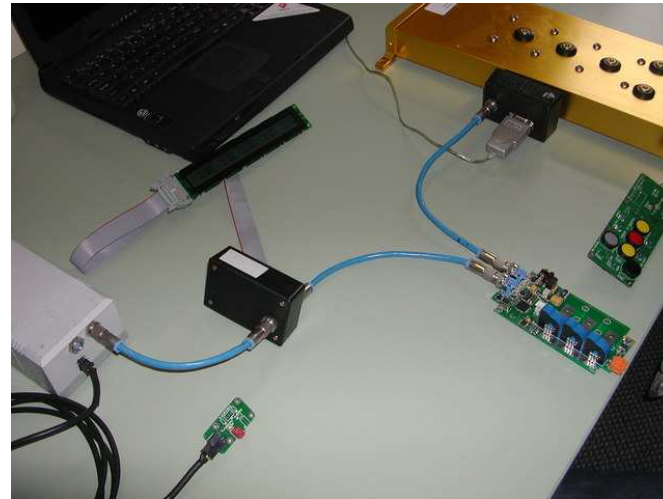
daves@cse.unsw.edu.au

- ① About me
- ② How hardware exists

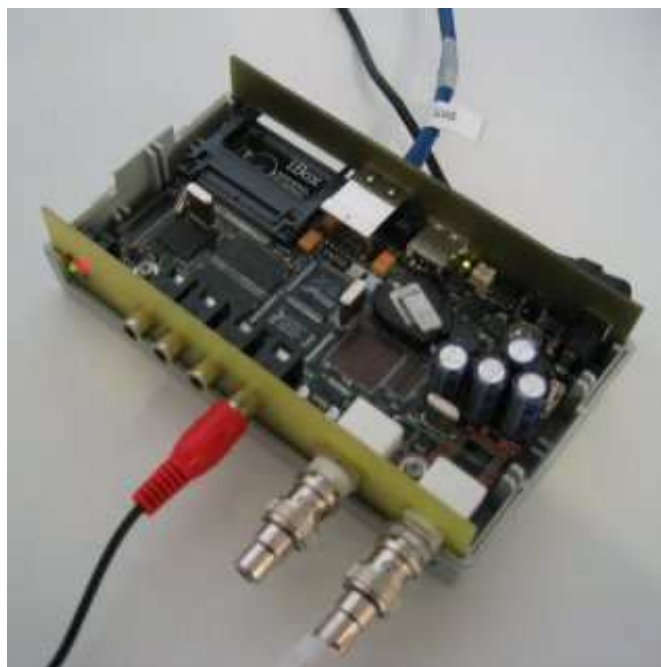
ABOUT ME

- Computer Engineering 1999–2002
- Honours: Hardware and Software Infrastructure for Sunswift II
- PhD: Operating system directed power management
- Solar cars: WSC (99, 01, 03, 05)





IBOX



- purpose: intelligent video surveillance;
- 624MHz Xscale (PXA270), 64MB RAM, 4MB Flash;
- analogue video capture, overlays, MPEG 4 encoding, audio;
- USB, CF, Ethernet;
- designed for L4/Iguana (OK inside).

HARDWARE

HARDWARE

aka: convincing yourself that it just might work

OVERVIEW

Overview:

→ why talk about hardware design?

Because:

- Hardware designs are difficult to change.
- Embedded systems require hardware-software co-design.
- Designing the hardware right make the software much nicer.
- Hardware and software engineers should work together.

OVERVIEW

Overview:

- why talk about hardware design?

Because:

- Hardware designs are difficult to change.
- Embedded systems require hardware-software co-design.
- Designing the hardware right make the software much nicer.
- Hardware and software engineers should work together.

Note: this is an informal discussion.

OVERVIEW

Case Study: the iBox:

- requirements;
- general design;
- schematics;
- layout;
- manufacture;
- board bring-up.

REQUIREMENTS

Purpose:

- processing platform for digital video surveillance;
- teaching platform for AOS.

REQUIREMENTS

Purpose:

- processing platform for digital video surveillance;
- teaching platform for AOS.

Video surveillance:

- motion detection;
- number recognition;
- face recognition.

REQUIREMENTS

Functional:

- Performance: five frames per second at 352×288 ;
- IO: Ethernet, audio, video, alarm, overlay;
- Video compression: MPEG4 at 768×576 ;
- System software update: by an engineer on-site;
- Power supply: up to 24V AC or DC;
- Power consumption: less than 5W.
- Software interfaces: video, audio, network stack, overlay generation, MPEG compression

REQUIREMENTS

Non-functional:

- Cost
- Modularity
- Size
- Development time
- Connectors: standard
- Un-trusted software.

REQUIREMENTS

Non-functional:

- Cost
- Modularity
- Size
- Development time
- Connectors: standard
- Un-trusted software.

It should cost nothing, use no power, be infinitely small, and take no time to develop. What's the problem?

REQUIREMENTS

Trade-offs:

- Cost vs. Performance
- Size vs. Performance
- Size vs. Utility
- Power vs. Performance
- Time vs. everything
- Cost vs. Quantity
- In-house vs. out-sourcing

REQUIREMENTS

Summary:

- MPEG 4 requires *serious* processing grunt;
- Motion detection requires *serious* processing grunt;

DESIGN

Strategy:

- ① come up with lots of options;
- ② pick one that best fits the requirements;
- ③ lather, rinse, repeat;
- ④ work out that you've made a mistake as early as possible.

We'll go through a few of the design decisions we made here

DESIGN

FPGA:

- *Field Programmable Gate Array*
- RAM + logic == configurable IC
- “programmed” using VHDL, Verilog, etc.
- can use soft cores

DESIGN

FPGA:

- *Field Programmable Gate Array*
- RAM + logic == configurable IC
- “programmed” using VHDL, Verilog, etc.
- can use soft cores

For:

- ✓ high performance
- ✓ low power

Against:

- ✗ difficult to implement algorithms
- ✗ some expensive soft-cores
- ✗ no L4

DESIGN

DSP:

- Digital signal processor
- VLIW processors optimised for signal processing
- e.g. TI TMS320C6416: 8 GMACs/s @ 1GHz

DESIGN

DSP:

- Digital signal processor
- VLIW processors optimised for signal processing
- e.g. TI TMS320C6416: 8 GMACs/s @ 1GHz

For:

- ✓ very high signal processing performance
- ✓ various implementations with video interfaces

Against:

- ✗ no MMU
- ✗ optimisation required to achieve performance
- ✗ expensive compilers

DESIGN

Digital signal processor + Generic processor:

- use a general-purpose processor for control
- use a DSP for signal processing algorithms
- often packaged together (e.g. TI OMAP, DaVinci)

DESIGN

Digital signal processor + Generic processor:

- use a general-purpose processor for control
- use a DSP for signal processing algorithms
- often packaged together (e.g. TI OMAP, DaVinci)

For:

- ✓ solves lots of problems with a DSP alone
- ✓ easier to handle real-time issues

Against:

- ✗ higher software complexity than a single-core solution
- ✗ need more than one DSP for our application

DESIGN

General-purpose processor:

→ optimised for a wide range of tasks

DESIGN

General-purpose processor:

→ optimised for a wide range of tasks

For:

- ✓ good support from compilers and operating systems
- ✓ MMU
- ✓ system-on-a-chip designs make implementation easier/smaller

Against:

- ✗ very high performance required == high power, large size
- ✗ not optimised for signal processing
- ✗ not capable of MPEG 4 encode

DESIGN

SMP:

→ multiple generic cores

DESIGN

SMP:

- multiple generic cores

For:

- ✓ high performance
- ✓ retains generic capabilities while providing processing power

Against:

- ✗ large size
- ✗ high power
- ✗ higher parts count

DESIGN

What next?:

- rough out the designs: identify components
- don't worry about the easy bits (PSU, connectors)
- obtain development kits and test performance
- look at how well the software fits the design

Calculate:

- cost
- parts count
- part availability
- power consumption

DESIGN

What we considered:

- ① PXA270
- ② TI DaVinci
- ③ Sierra MIPS

DESIGN

What we considered:

- ① PXA270
- ② TI DaVinci
- ③ Sierra MIPS

We chose the PXA270, combined with an external MPEG 4
codec ASIC

PXA270 based solution:

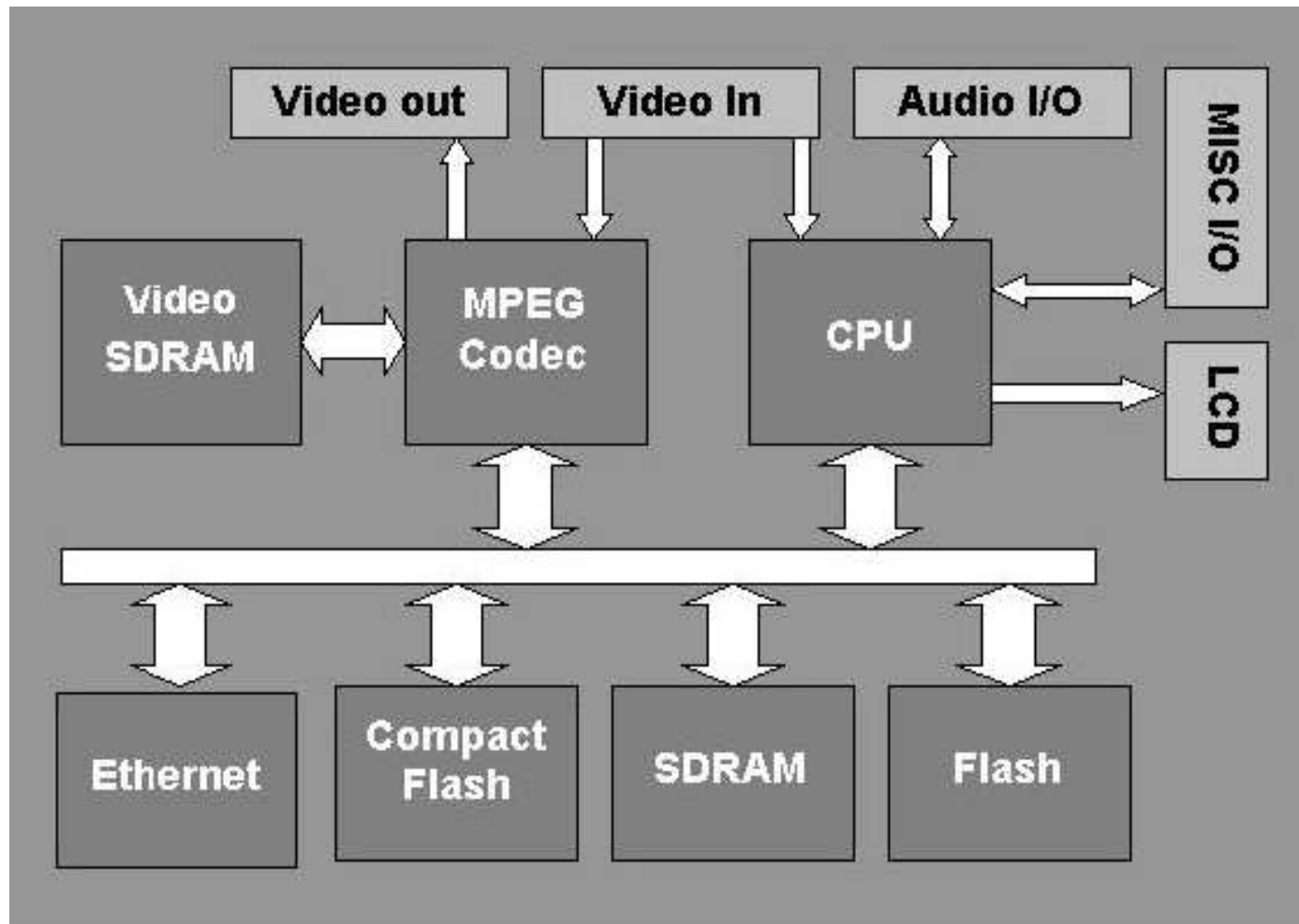
- 624MHz XScale (ARM) core — fast enough for video algorithms
- quick-capture interface for video input
- tricky circuit for overlay generation
- other useful built-in peripherals (DMA, SRAM, USB, CF, LCD)
 - ✓ L4 was already ported
 - ✓ good Linux support
 - ✓ MMX and DSP instructions
 - ✓ very low power

DESIGN

PXA270 based solution:

- 624MHz XScale (ARM) core — fast enough for video algorithms
- quick-capture interface for video input
- tricky circuit for overlay generation
- other useful built-in peripherals (DMA, SRAM, USB, CF, LCD)
- ✓ L4 was already ported
- ✓ good Linux support
- ✓ MMX and DSP instructions
- ✓ very low power
- ✗ requires some expensive PCB technologies

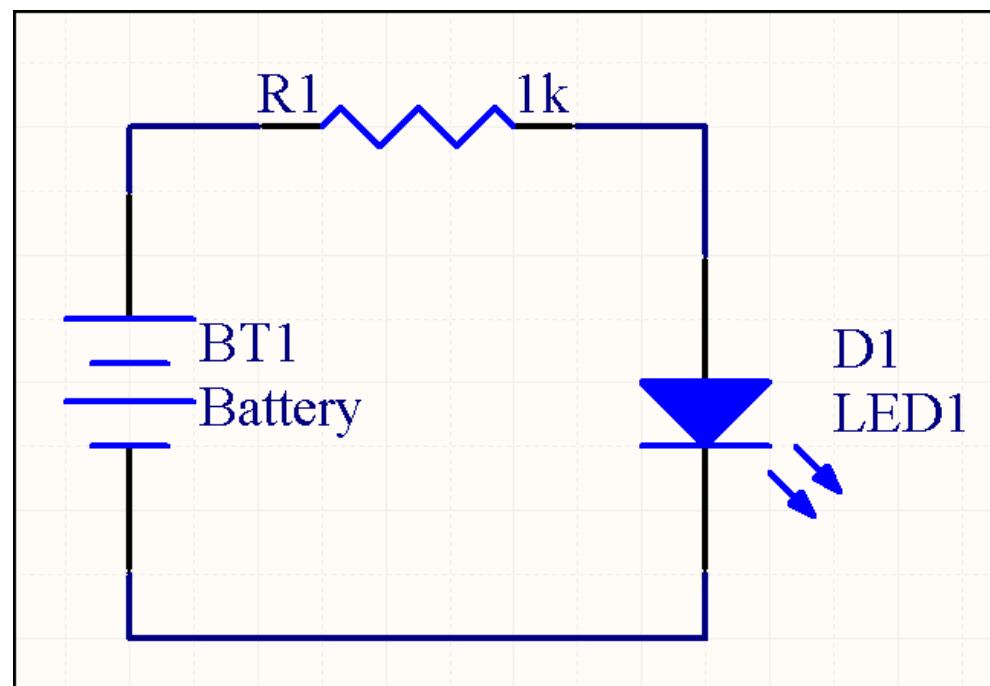
DESIGN



SCHEMATICS

Schematics:

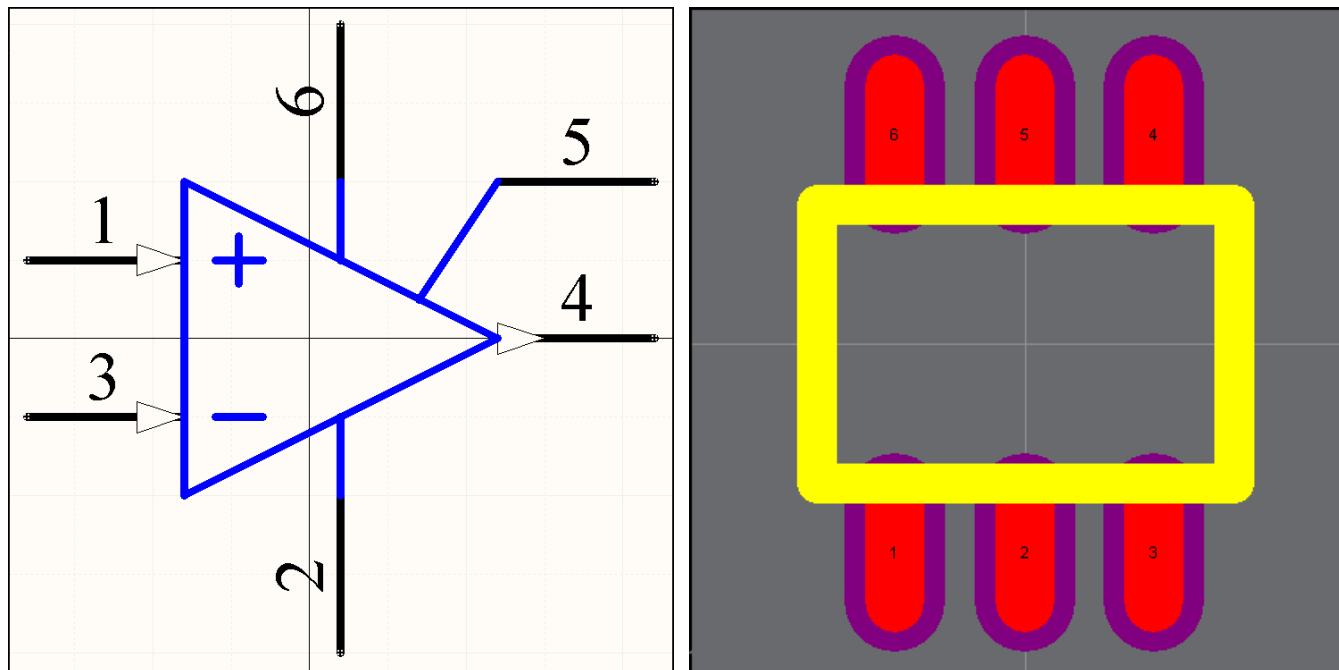
- schematics define how components connect logically
- component pins are connected to form *nets*
- hierarchical schematics are possible via *ports*



SCHEMATICS

Defining components:

- ① schematic component
- ② footprint



SCHEMATICS

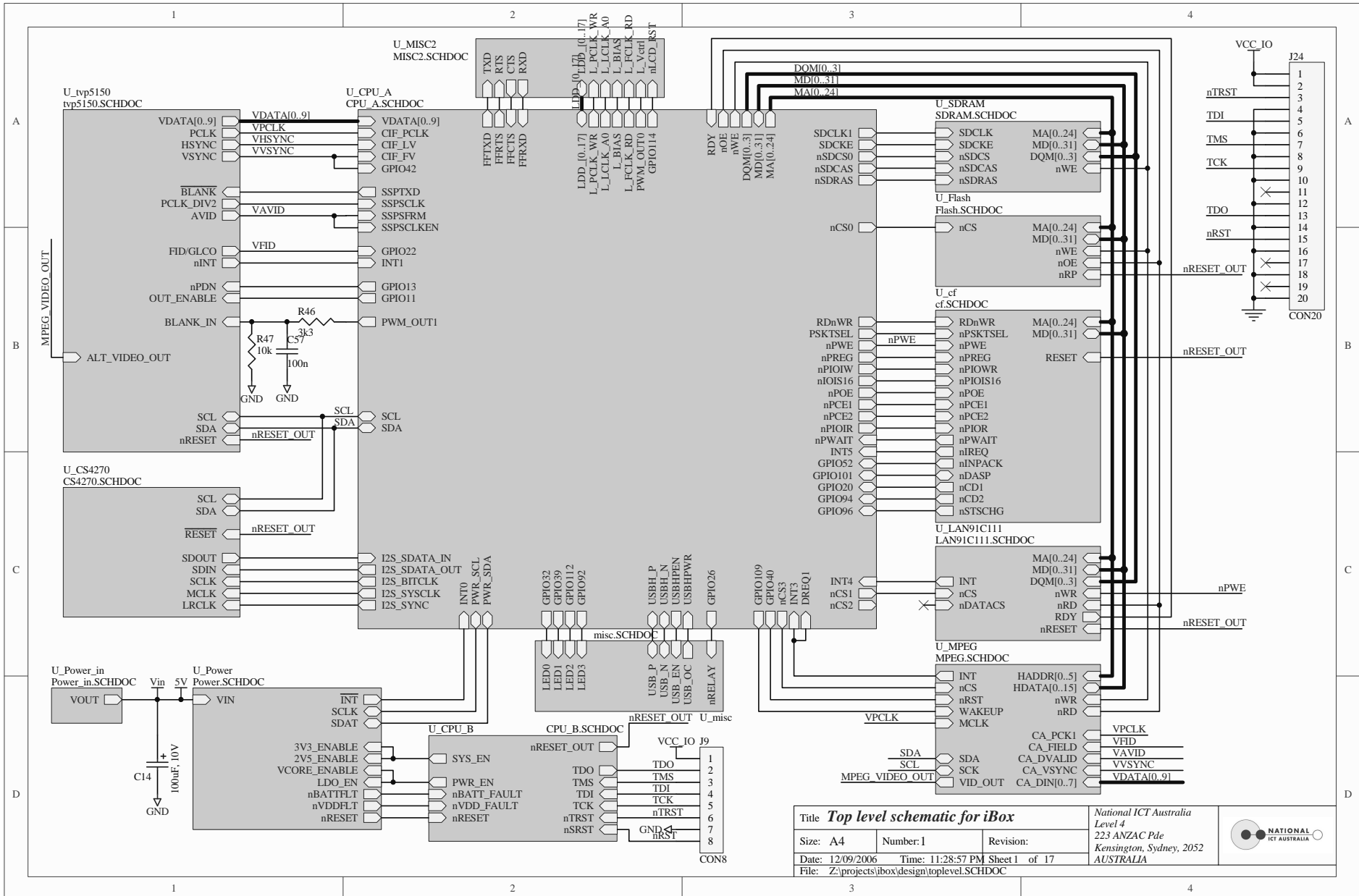
Things to think about:

- get it right first time
- decoupling – reduce power supply noise
- regulator noise
- pin assignment – shortest paths
- DMA

Hardware should be designed to make software elegant

SCHEMATICS

The iBox schematics:

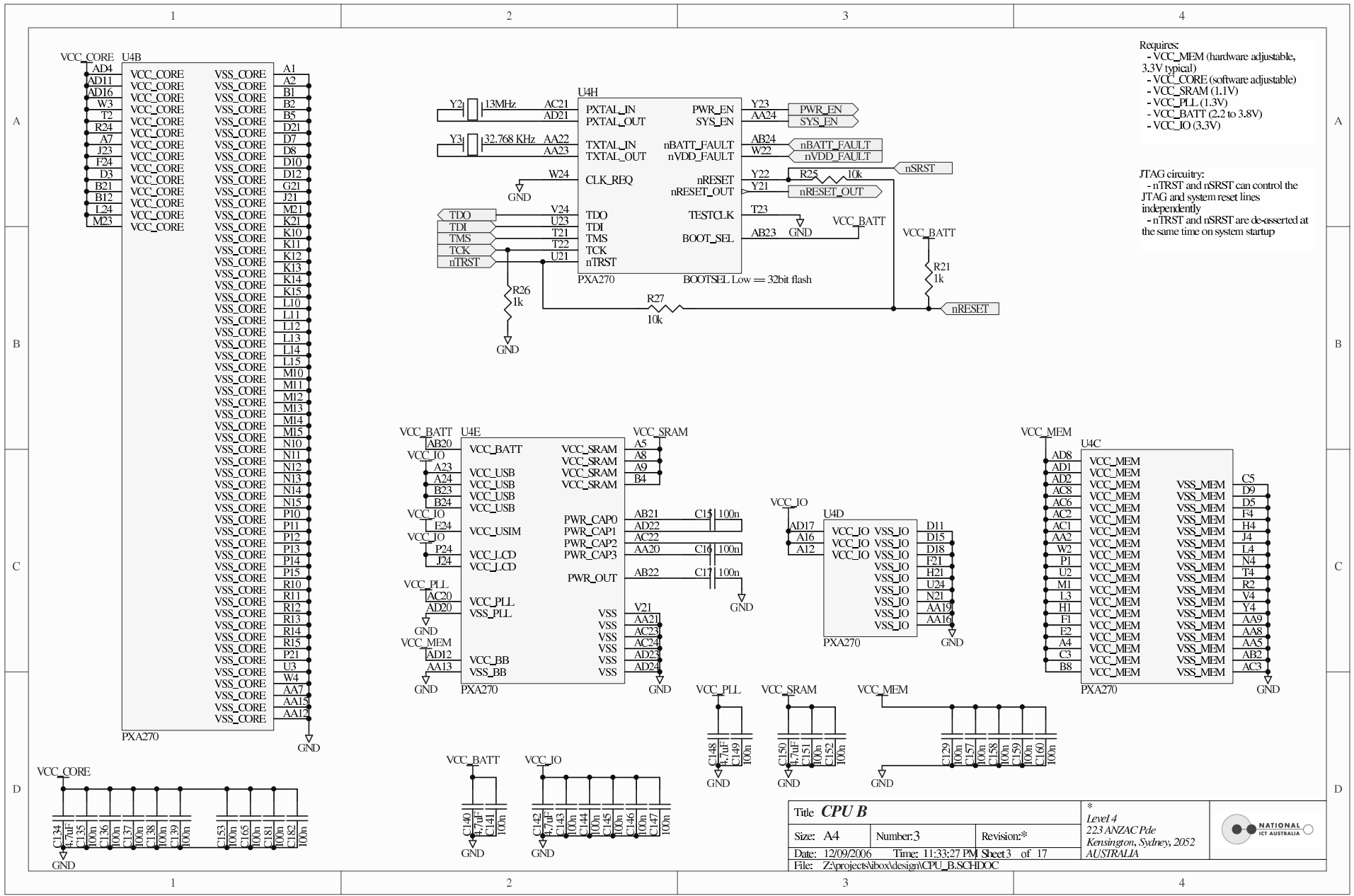


Title Top level schematic for iBox

Size: A4	Number: 1	Revision:
Date: 12/09/2006	Time: 11:28:57 PM	Sheet 1 of 17
File: Z:\projects\ibox\design\toplevel.SCHDOC		


National ICT Australia
 Level 4
 223 ANZAC Pde
 Kensington, Sydney, 2052
 AUSTRALIA

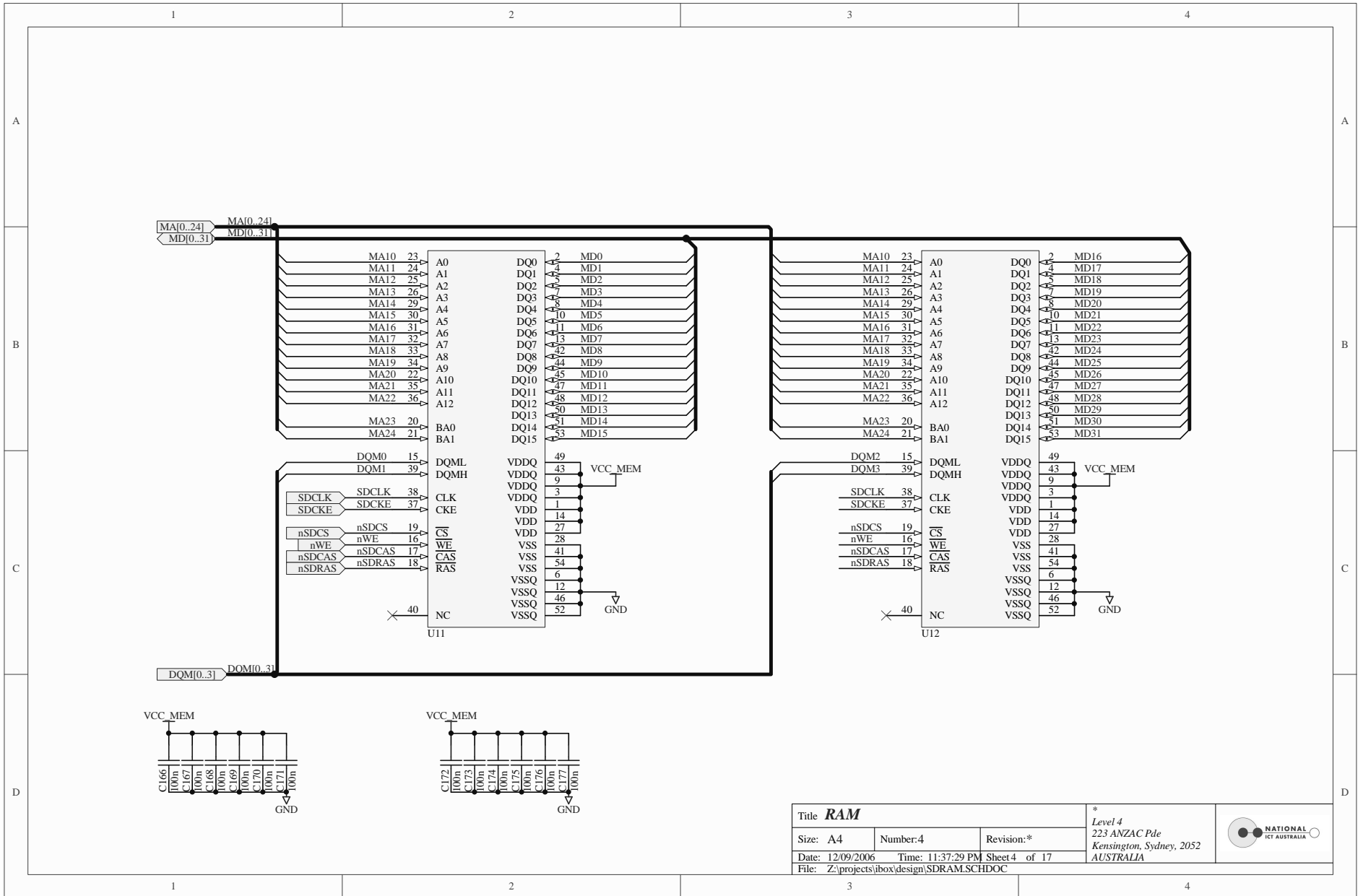




Requires:
 - VCC_MEM (hardware adjustable, 3.3V typical)
 - VCC_CORE (software adjustable)
 - VCC_SRAM (1.1V)
 - VCC_PLL (1.3V)
 - VCC_BATT (2.2 to 3.8V)
 - VCC_IO (3.3V)

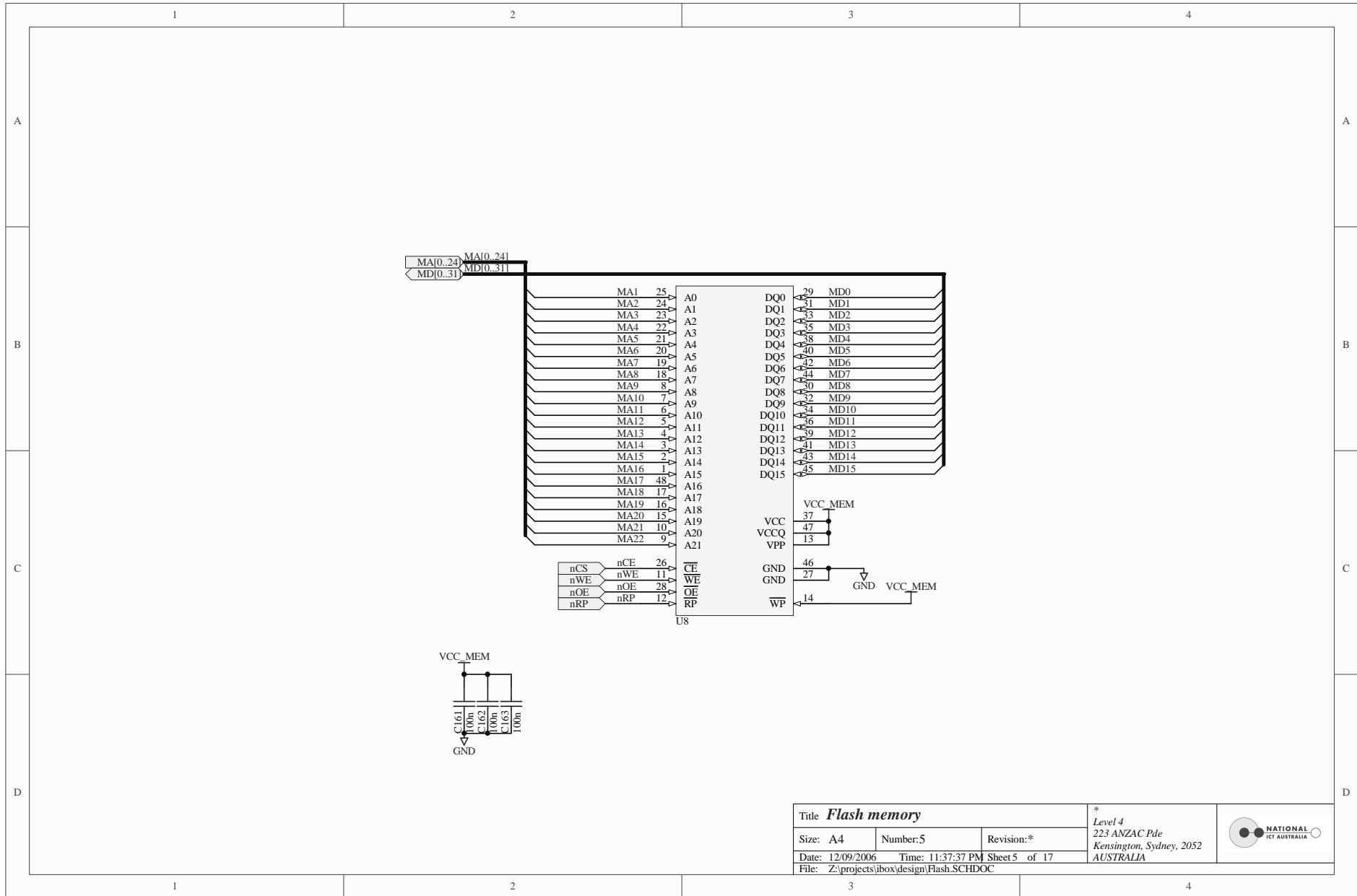
JTAG circuitry:
 - nTRST and nSRST can control the JTAG and system reset lines independently
 - nTRST and nSRST are de-asserted at the same time on system startup


Title CPU B			* Level 4
Size: A4	Number: 3	Revision: *	223 ANZAC Pde
Date: 12/09/2006	Time: 11:33:27 PM	Sheet 3 of 17	Kensington, Sydney, 2052
File: Z:\projects\ibox\design\CPU_B.SCHDOC			 NATIONAL ICT AUSTRALIA

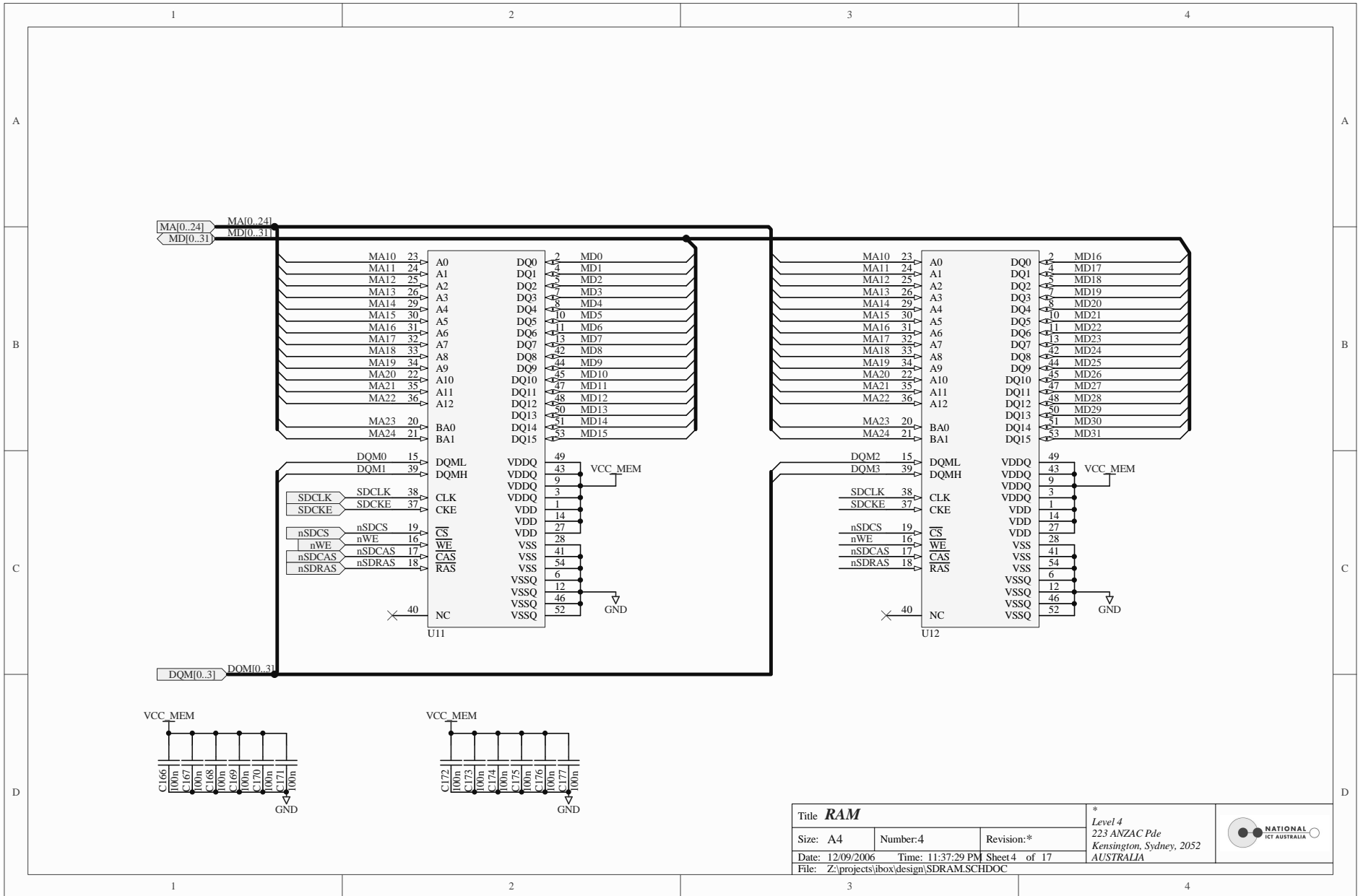


Title RAM			* Level 4 223 ANZAC Pde Kensington, Sydney, 2052 AUSTRALIA
Size: A4	Number: 4	Revision: *	
Date: 12/09/2006	Time: 11:37:29 PM	Sheet 4 of 17	
File: Z:\projects\ibox\design\SDRAM.SCHDOC			



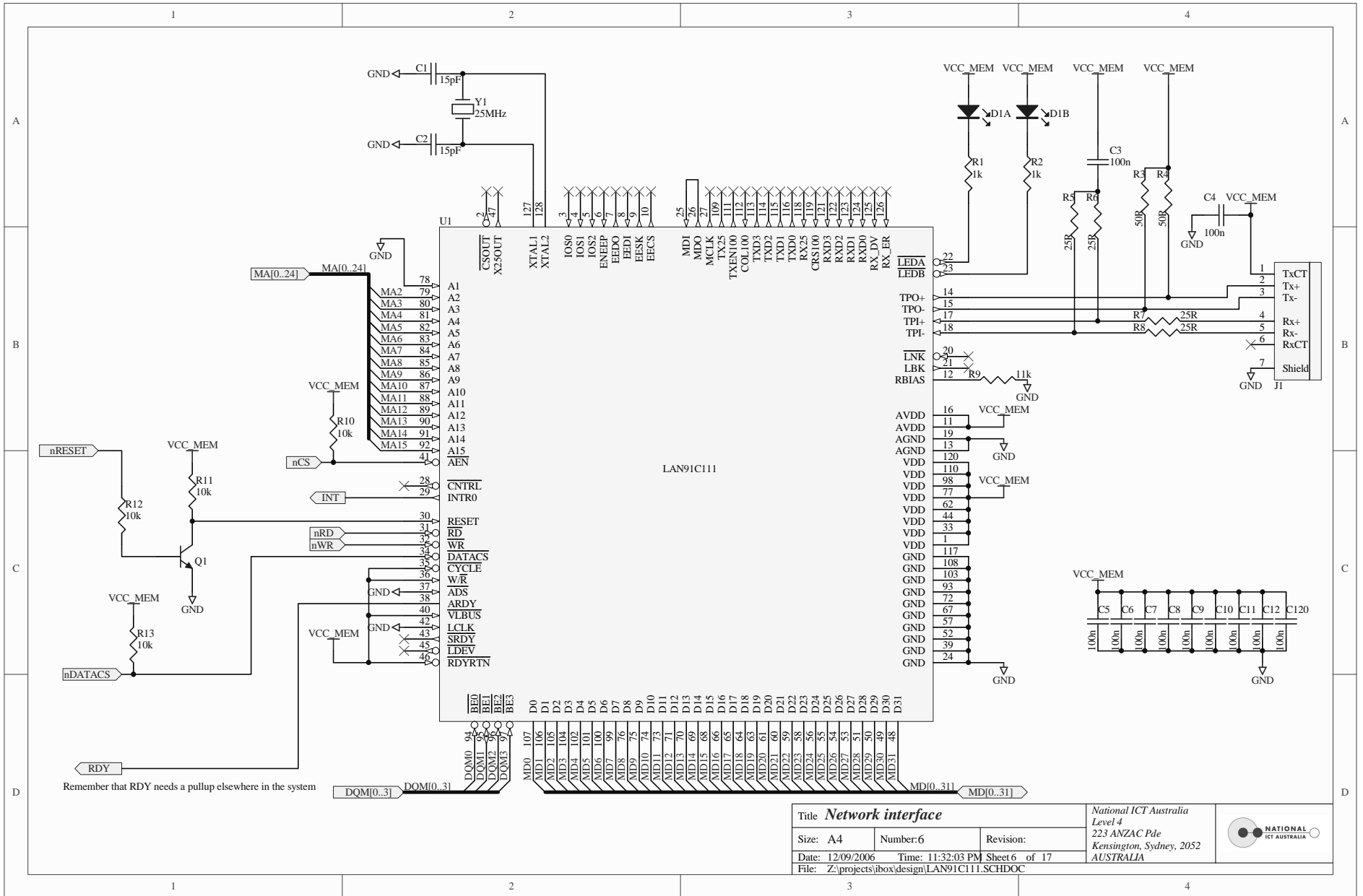


Title Flash memory			* Level 4 223 ANZAC Pde Kensington, Sydney, 2052 AUSTRALIA
Size: A4	Number: 5	Revision: *	
Date: 12/09/2006	Time: 11:37:37 PM Sheet 5 of 17		
File: Z:\projects\ibox\design\Flash.SCHDOC			



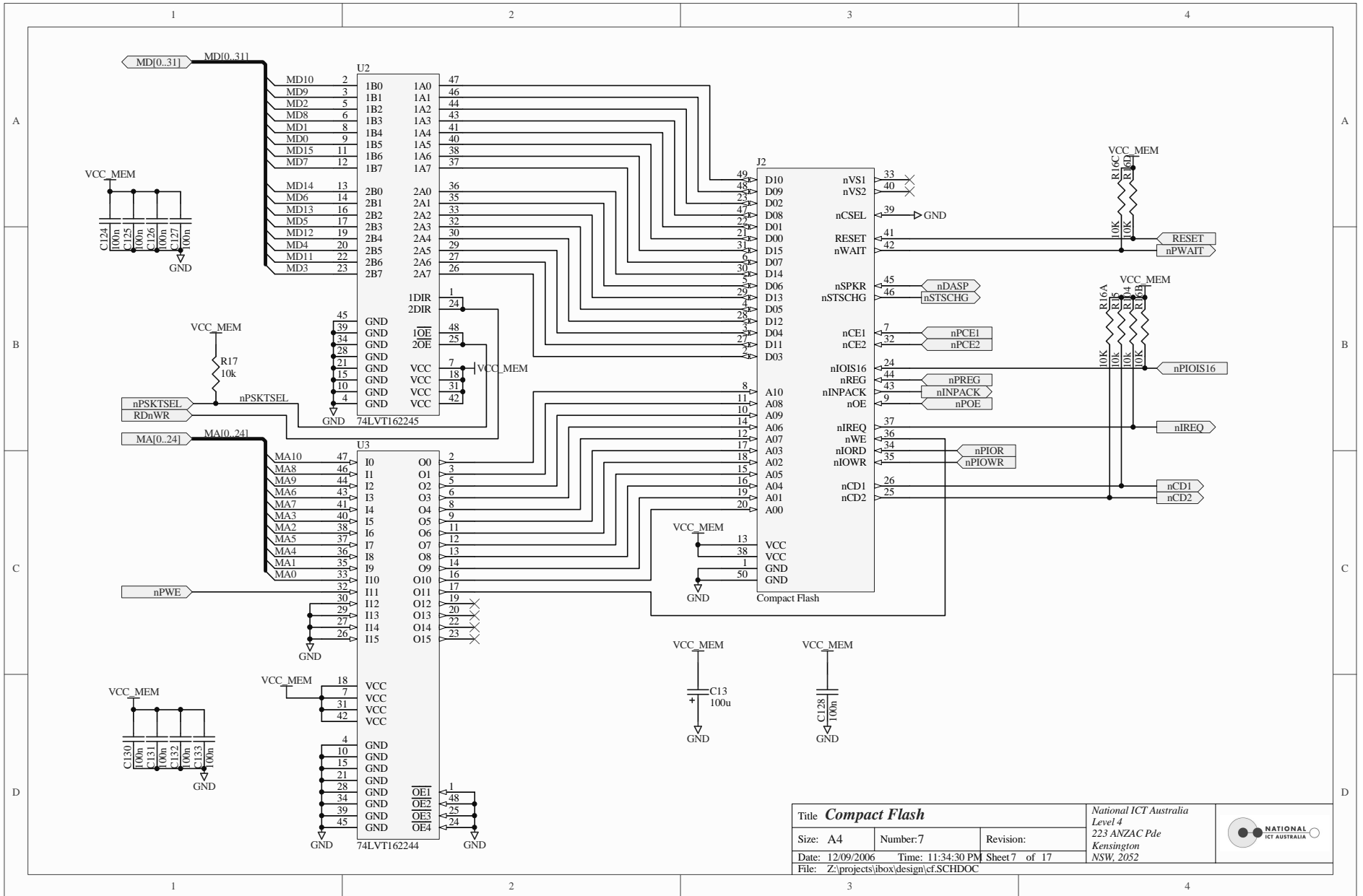
Title RAM			* Level 4 223 ANZAC Pde Kensington, Sydney, 2052 AUSTRALIA
Size: A4	Number: 4	Revision: *	
Date: 12/09/2006	Time: 11:37:29 PM	Sheet 4 of 17	
File: Z:\projects\ibox\design\SDRAM.SCHDOC			





Remember that RDY needs a pullup elsewhere in the system

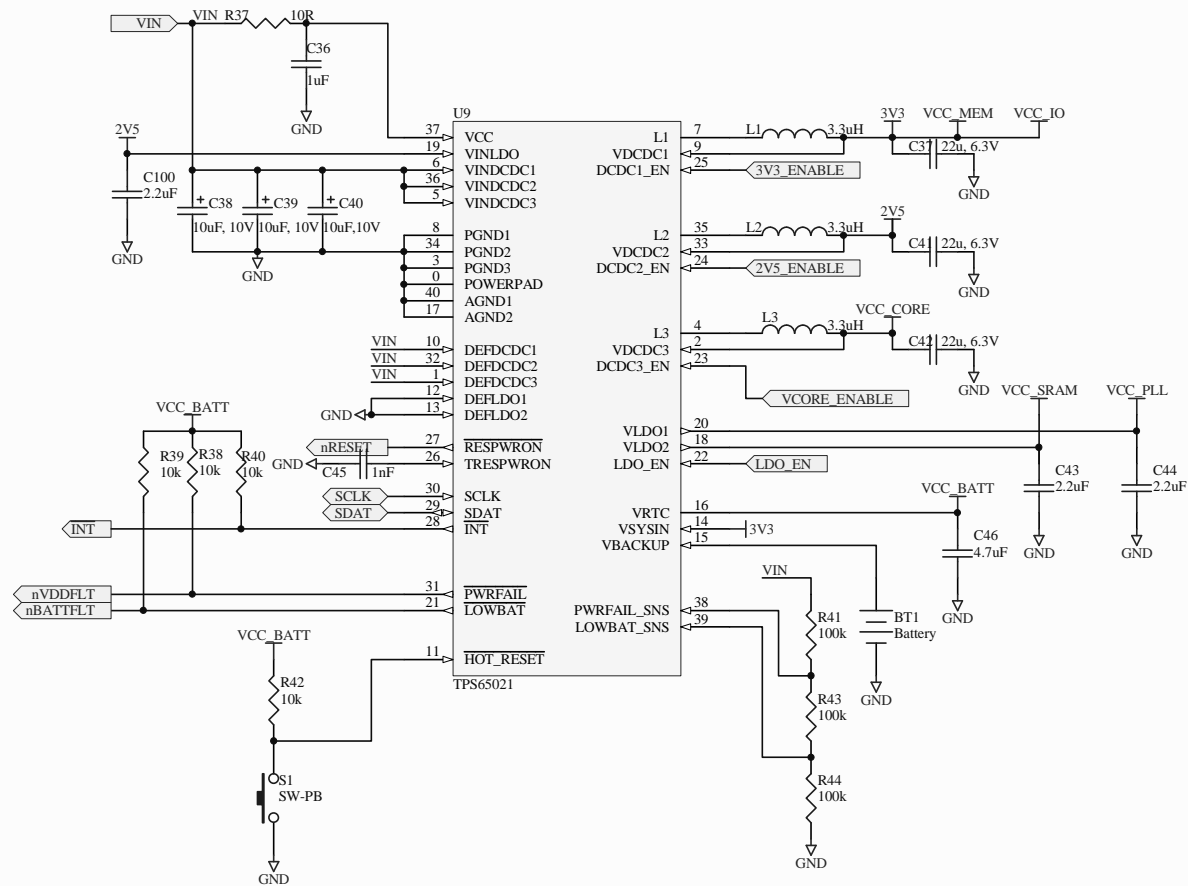
Title Network interface			National ICT Australia Level 4 223 ANZAC Pde Kensington, Sydney, 2052 AUSTRALIA	
Size: A4	Number: 6	Revision:		
Date: 12/09/2006	Time: 11:32:03 PM	Sheet 6 of 17		
File: Z:\projects\ibox\design\LAN91C11.LSCHDOC				



Title Compact Flash		
Size: A4	Number: 7	Revision:
Date: 12/09/2006	Time: 11:34:30 PM	Sheet 7 of 17
File: Z:\projects\ibox\design\cf.SCHDOC		

National ICT Australia
 Level 4
 223 ANZAC Pde
 Kensington
 NSW, 2052





Title **Power supplies**

Size: A4 Number:9 Revision:
 Date: 12/09/2006 Time: 11:34:39 PM Sheet 9 of 17
 File: Z:\projects\ibox\design\Power.SCHDOC

National ICT Australia
 Level 4
 223 ANZAC Pde
 Kensington
 NSW, 2052



1

2

3

4

A

A

B

B

C

C

D

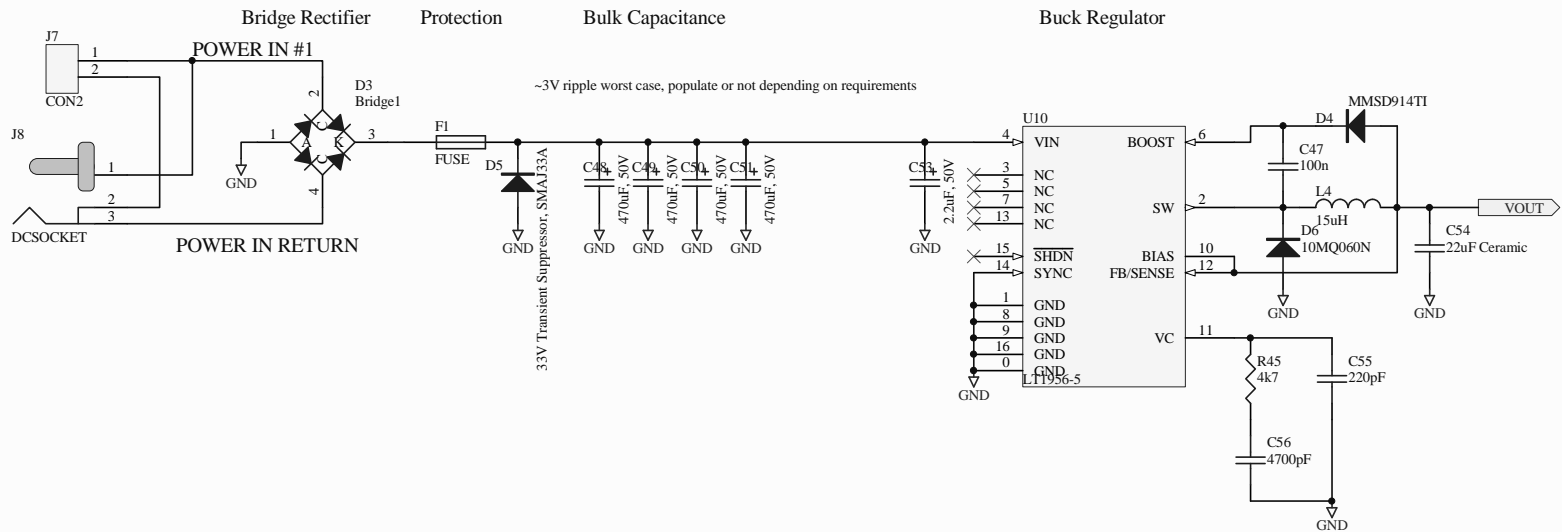
D

1

2

3

4



Title **Power input and conditioning**

Size: A4 Number: 10 Revision:
 Date: 12/09/2006 Time: 11:34:46 PM, Sheet 10 of 17
 File: Z:\projects\ibox\design\Power_in.SCHDOC

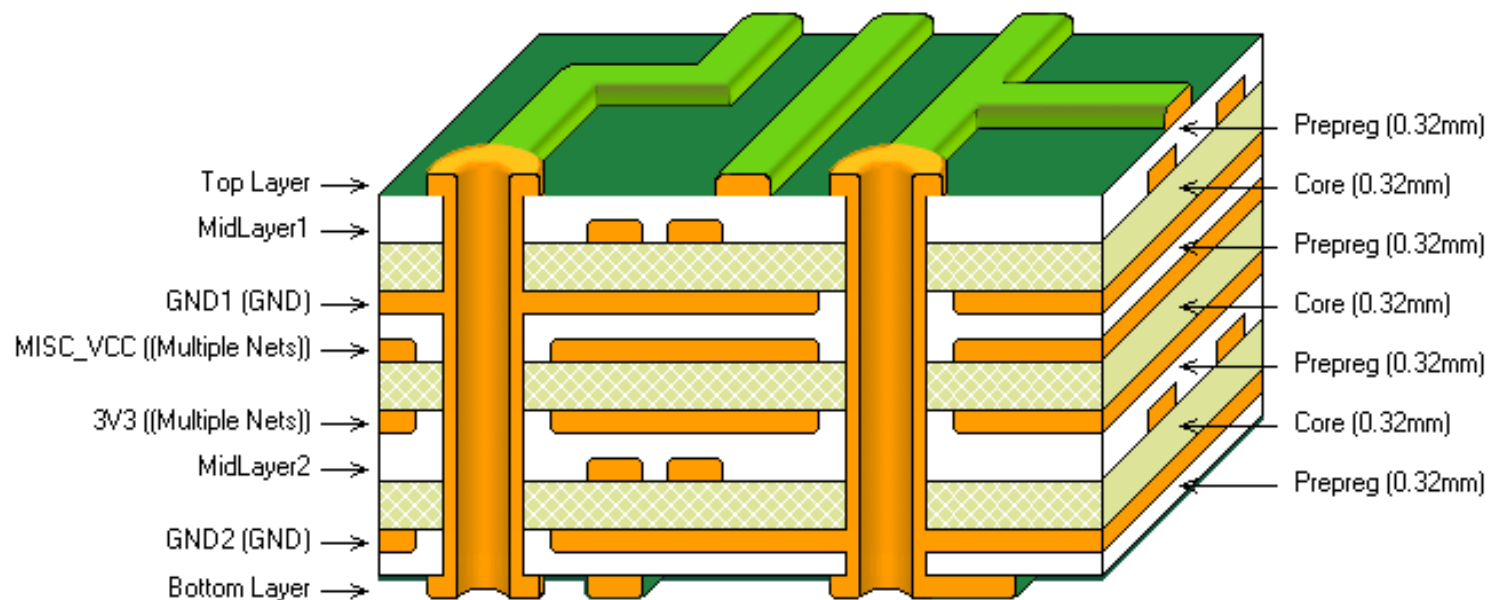
National ICT Australia
 Level 4
 223 ANZAC Pde
 Kensington
 NSW, 2052



LAYOUT

Layout technology and terminology:

- design a *printed circuit board* (PCB)
- layers of fibreglass and copper
- the iBox uses an 8 layer board (8 copper layers)
- *vias* are holes used to connect between layers

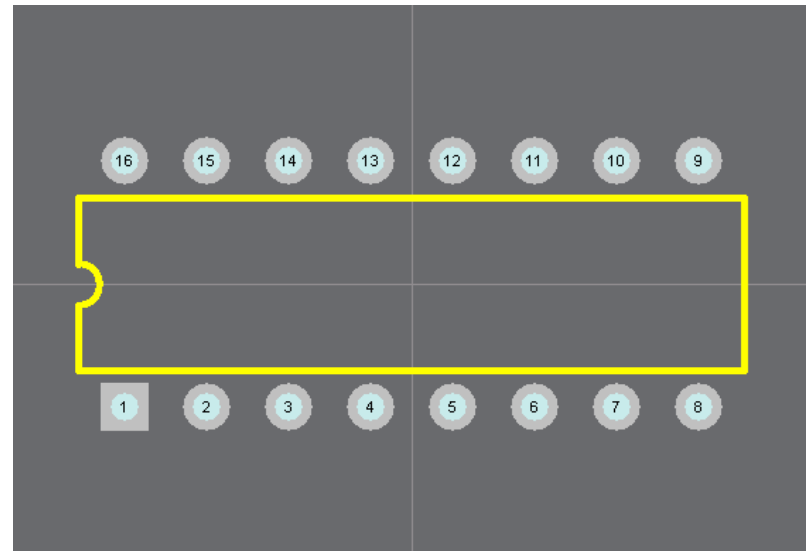
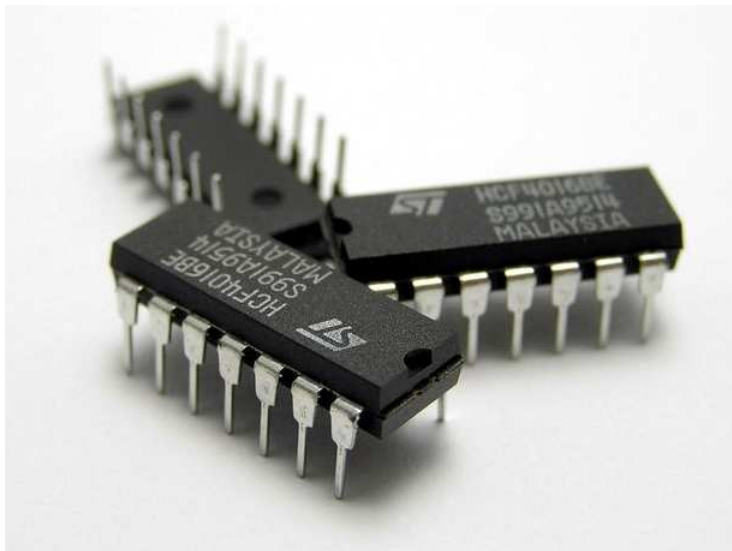


LAYOUT

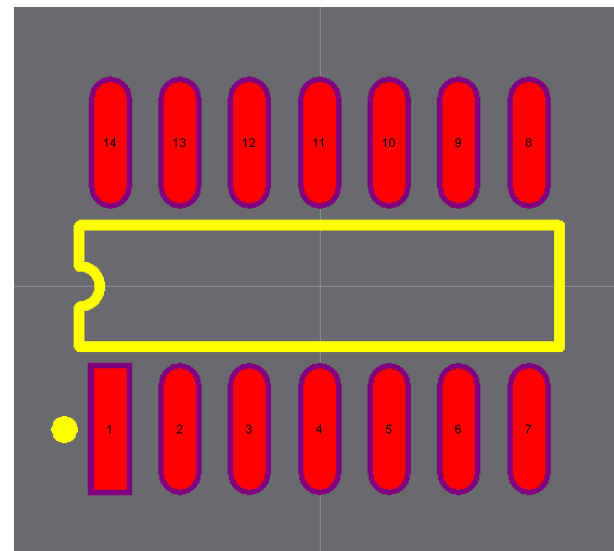
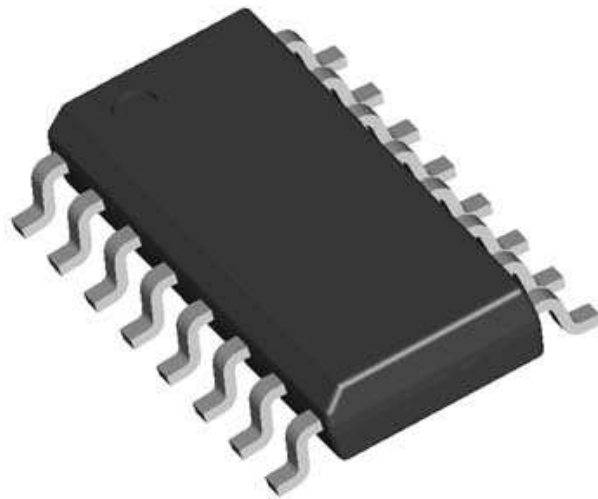
Component footprints:

- through-hole
- surface-mount

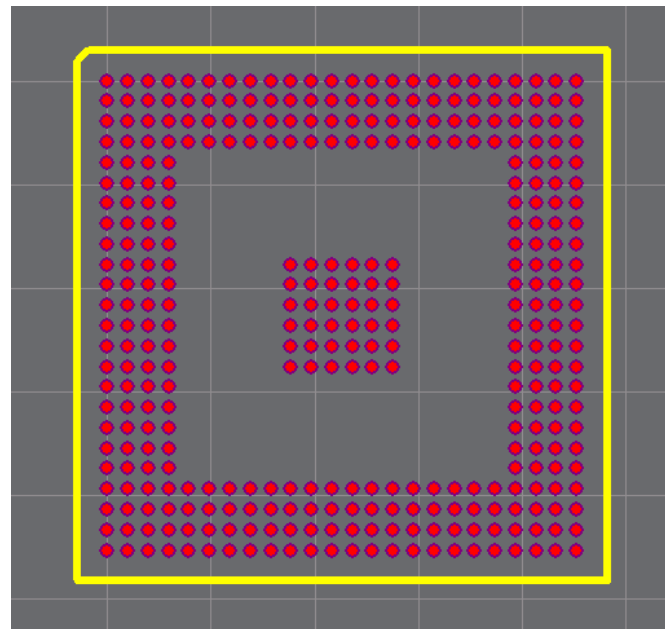
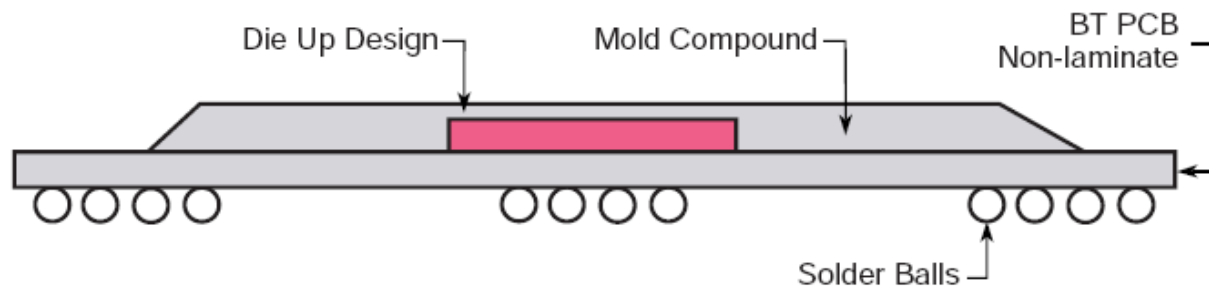
LAYOUT



LAYOUT



LAYOUT



LAYOUT

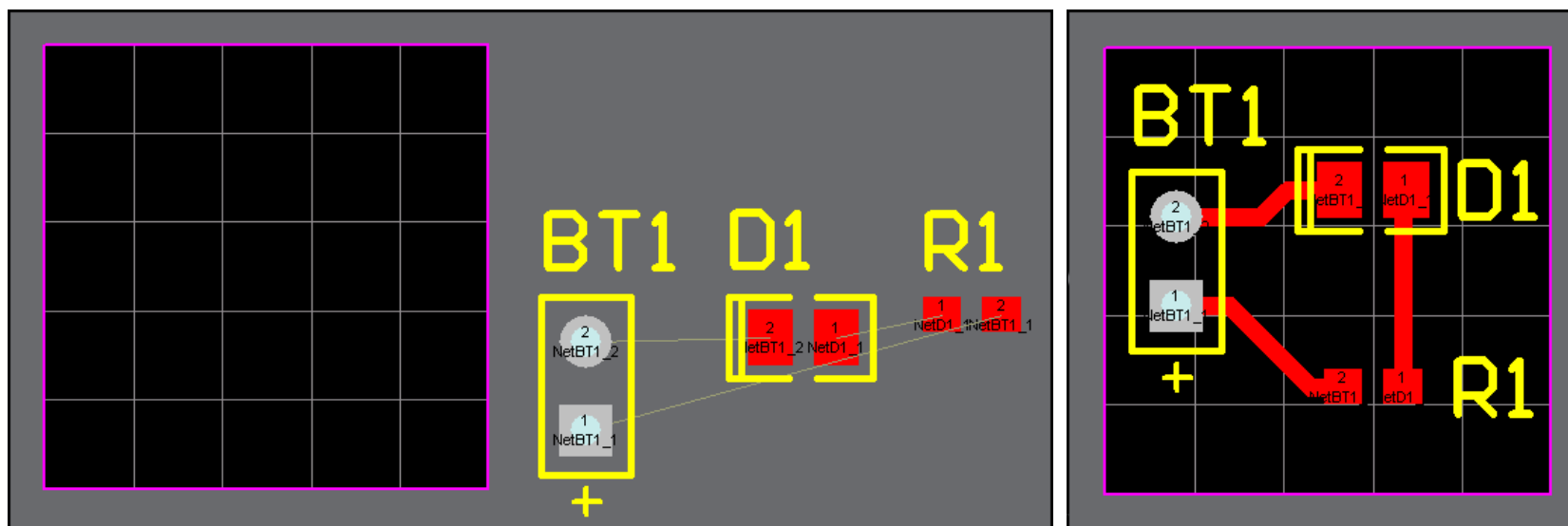
Trade-offs:

- number of layers vs. size
- component size vs. manufacturability

LAYOUT

Routing:

- the physical pins are connected with traces of copper
- the design software only allows connections defined by the schematic
- the *artwork* is used by a manufacturer to build a PCB



LAYOUT

Things to think about:

- Auto-routing vs. manual routing
- boxes and enclosures
- connector locations
- pin assignment
- size of the traces
- isolating noise
- signal integrity
- simulation

LAYOUT

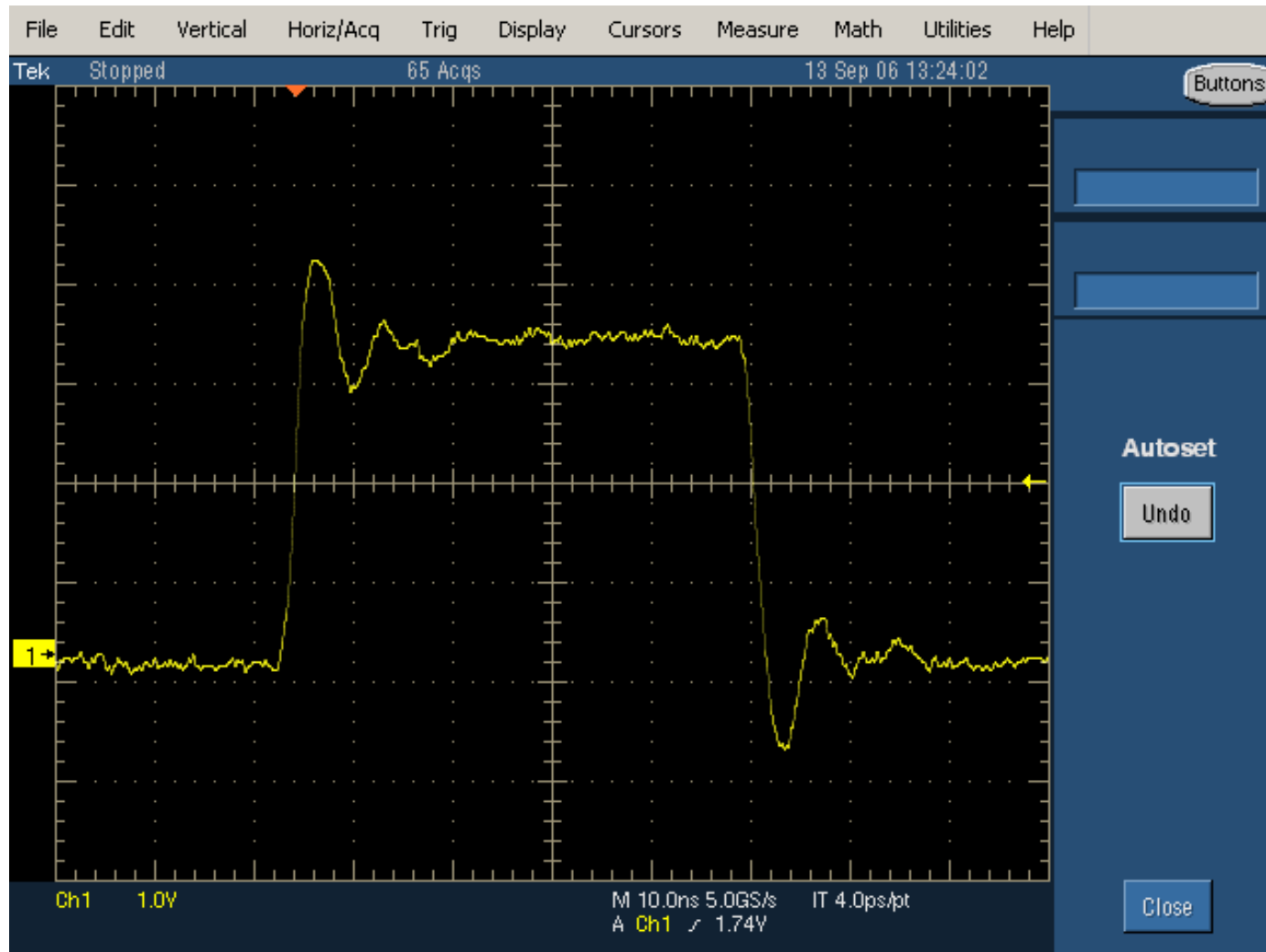
Signal integrity: Digital systems are still analogue!

- Above 100MHz, edges on digital signals tend to have high-frequency components
- transmission line effects
- inductance and capacitance in the traces/packages
- noise, cross-talk, EMI problems

So:

- keep stub lengths short
- use termination
- think about return paths
- use simulation

LAYOUT

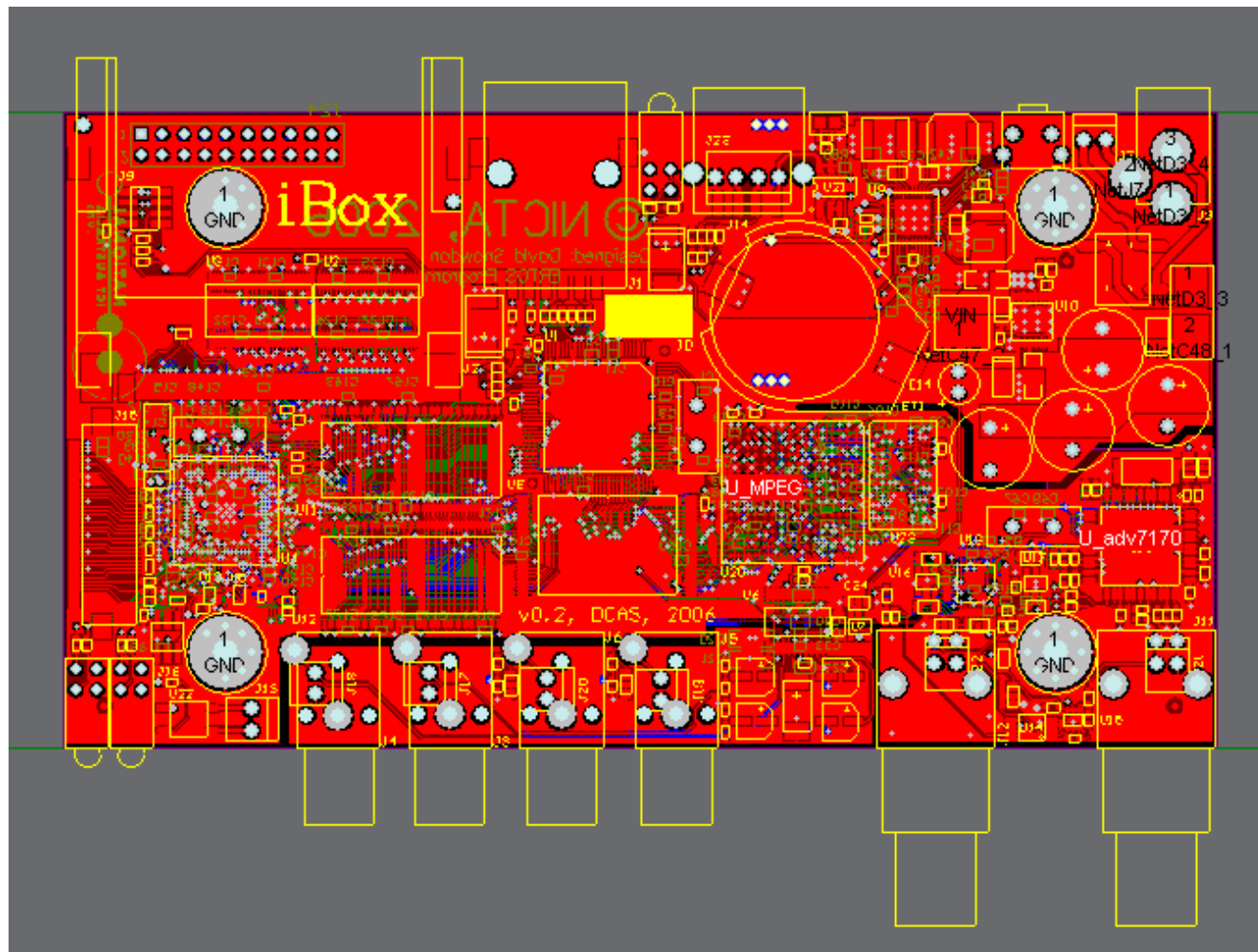


LAYOUT

iBox layout:

- 624MHz PXA270 is a uBGA — 0.5mm between balls
- We need *very* small vias — laser drilling
- We need buried and blind vias

LAYOUT



MANUFACTURE

Surface-mount:

- Solder stencil
- Pick and place
- Reflow

Through-hole:

- wave soldering

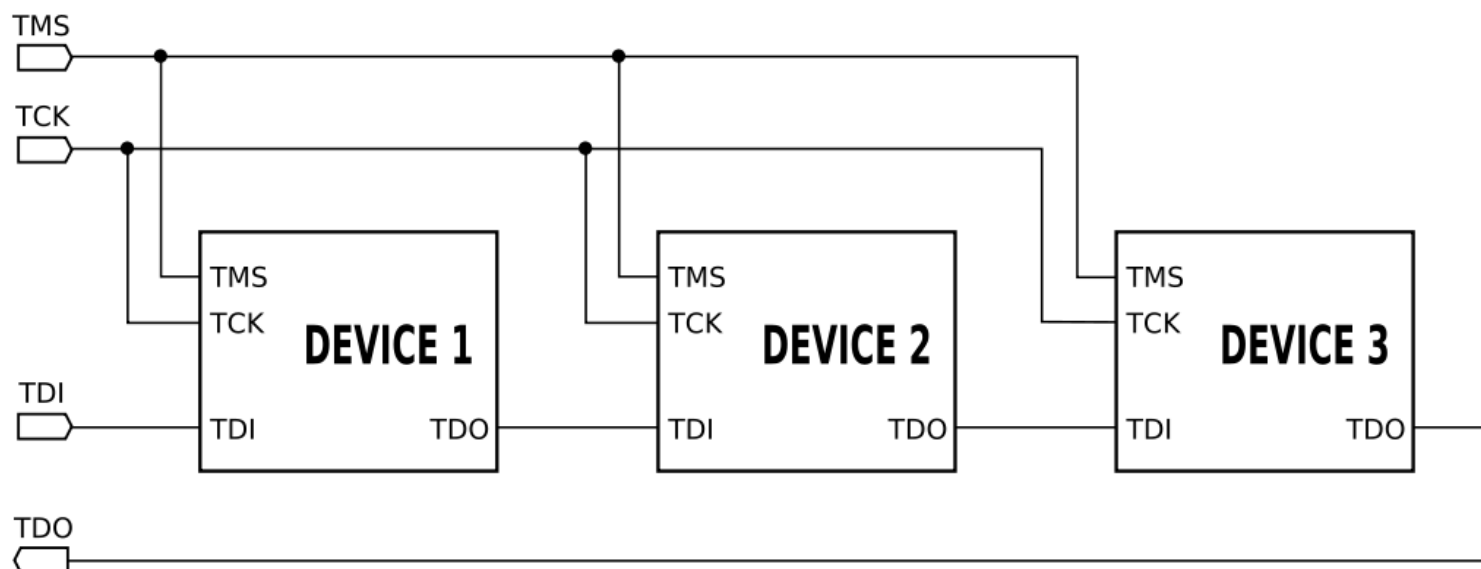


BRING-UP

JTAG: Joint Test Action Group – a standard for board testing

- boundary scan
- debugging
- custom commands

JTAG can be daisy-chained – multiple chips on one port

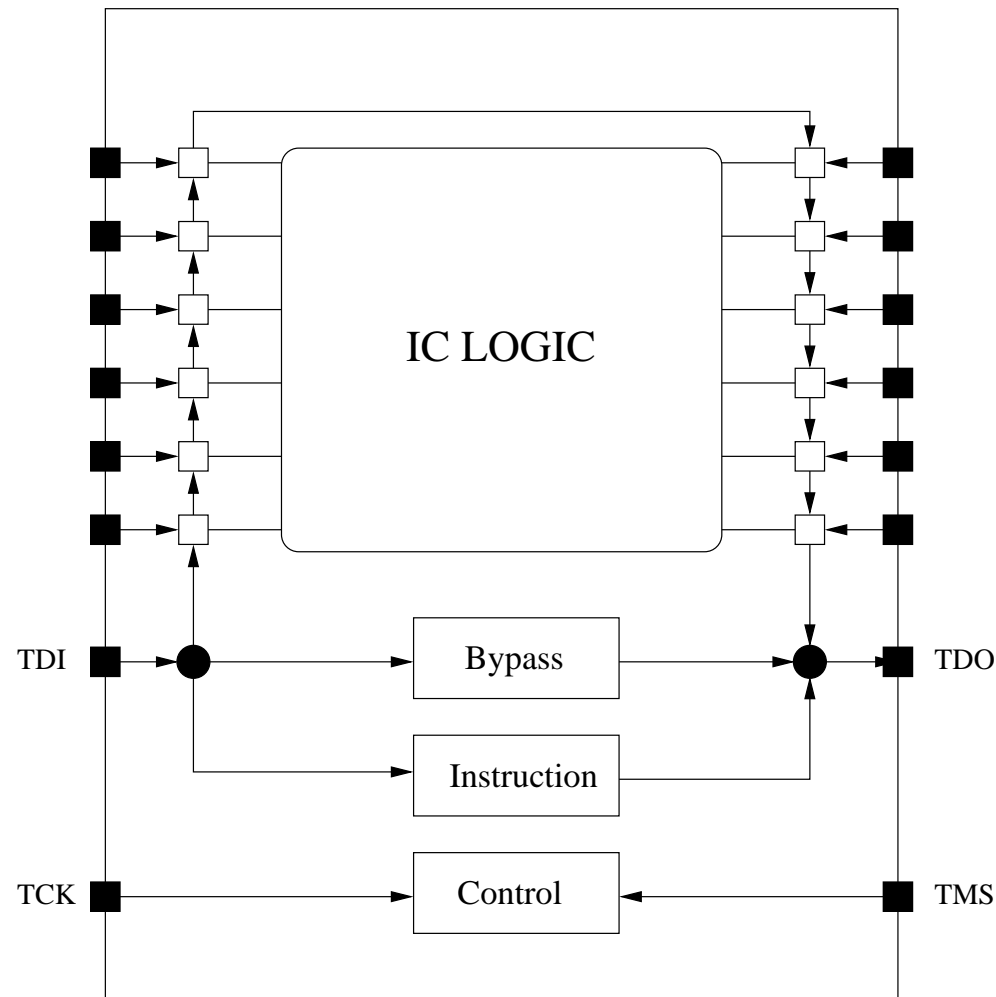


BRING-UP

PCB Testing:

- PCBs are tested using a “bed of nails”
- this doesn't work with BGA and PGA packages
- JTAG defines boundary scan: the device pins can be controlled
- can be used in combination with “bed of nails”

BRING-UP



BRING-UP

iBox JTAG:

- Boundary scan lets us control all of the pins on the PXA270
- so we can manually control the bus
- so we can program the flash memory

BRING-UP

Debug:

- some processors have extra, debugging JTAG instructions
- Xscale cores can execute instructions from a mini-instruction cache
- breakpoints
- single step
- watch variables

Trace:

- a *trace port* exports which instructions are run
- analysis tools can step backward

BRING-UP

Bootloader:

To load software, you need software?

- prepares the system for boot
- handles device-specific tasks
- loads and jumps to the system software entry
- commonly saved in read-only memory as firmware
- can be very simple or complex
- some systems use multi-stage loaders

The iBox uses U-Boot, the Micros lab AVR uses SAIL

BRING-UP

Start up on the PXA270:

- ① start executing at 0x00000000 (Flash, on iBox)
- ② configure the GPIO pins
- ③ initialise the static memory and SDRAM
- ④ initialise the IRQs
- ⑤ set the core clock frequency
- ⑥ enable the caches
- ⑦ copy into RAM
- ⑧ set up a stack
- ⑨ read configuration
- ⑩ set up devices and other services

BRING-UP

Bringing up the iBox:

- Power supplies
- Soldering the PXA270
- Testing using JTAG
- Flash
- SDRAM
- Bootloader
- Ethernet
- OS and the rest

BRING-UP

Thoughts:

- You guys know how to talk to hardware
- You guys have seen a few busses, interfaces, etc.
- The iBox is just more of the same!

Other:

- Problems are fixable.
- Find the information.
- Convince yourself it will work.

DEMO

DEMO