Hardware and Power Management

David Snowdon

daves@cse.unsw.edu.au

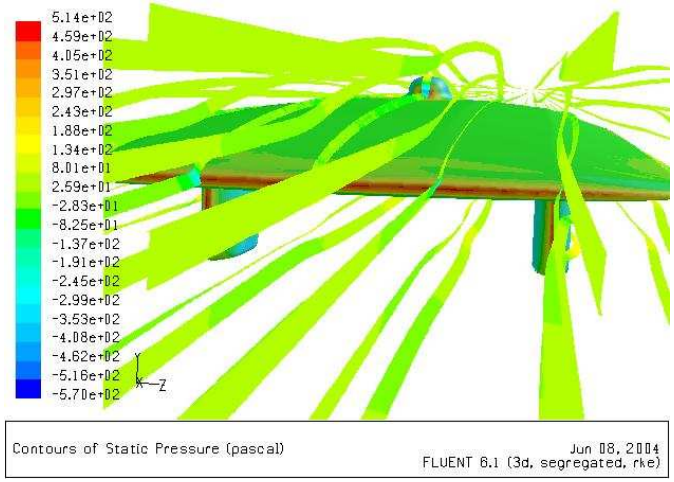
① About me

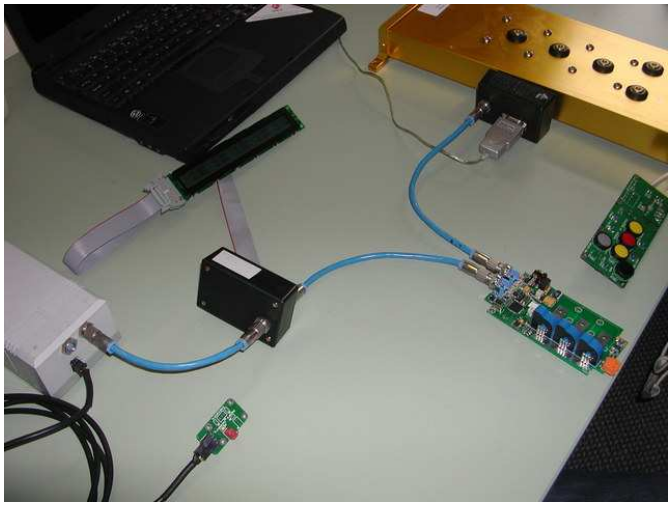
② How hardware exists

③ Power management

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

aka: convincing yourself that it just might work

Overview

Overview:

- why talk about hardware design in an OS course?
- hardware designs are difficult to change

Because:

- Operating systems interact closely with the hardware
- Hardware and system software engineers should work together
- OS developers are great embedded systems developers
- OS people think this stuff is cool

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 required performance == 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
- ☑ OS transparently handles multiprocessing

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

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

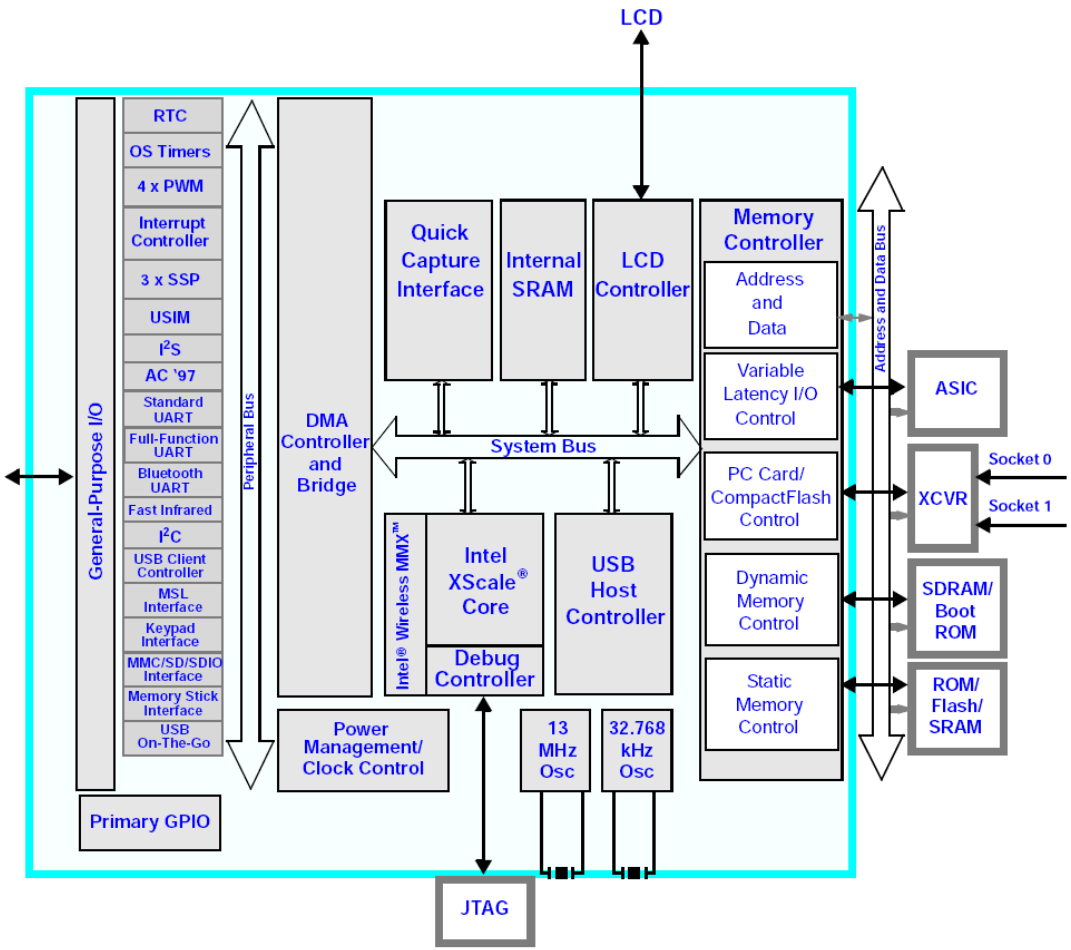
DESIGN

PXA270 based solution:

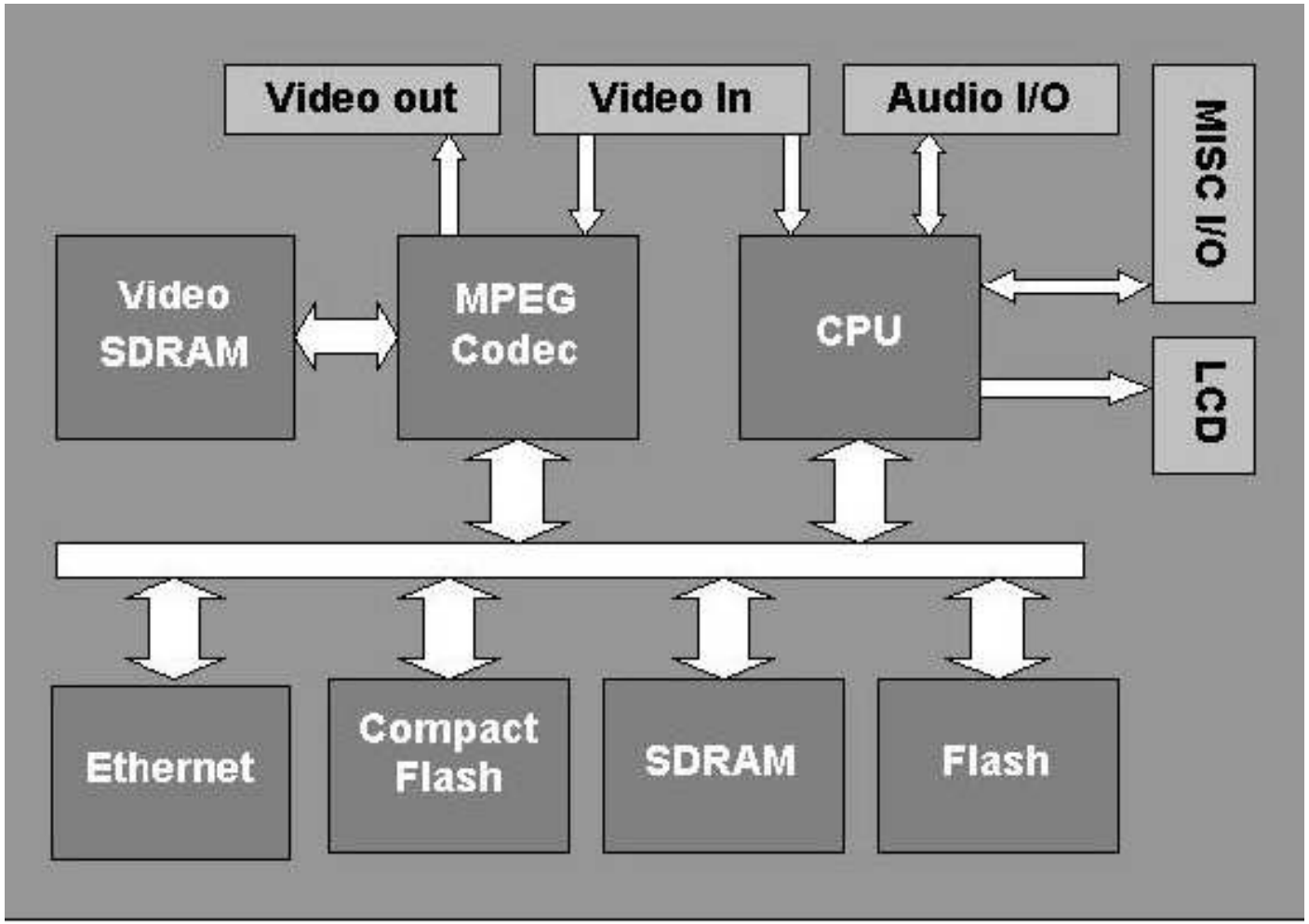
- ➜ 624MHz XScale (ARM) core — fast enough for video algorithms
- ➜ quick-capture interface for video input
- ➜ tricky circuit for overlay generation
- ➜ other useful built-in peripherals (DMA, SRAM, USB, CF, LCD)

- ☑ L4 was already ported
- ☑ good Linux support
- ☑ MMX and DSP instructions
- ☑ very low power
- ☒ requires some expensive PCB technologies

DESIGN


LCD
General-Purpose I/O
RTC
OS Timers
4 x PWM
Interrupt Controller
3 x SSP
USIM
I²S
AC '97
Standard UART
Full-Function UART
Bluetooth UART
Fast Infrared
I²C
USB Client Controller
MSL Interface
Keypad Interface
MMC/SD/SDIO Interface
Memory Stick Interface
USB On-The-Go
Primary GPIO
Peripheral Bus
DMA Controller and Bridge
Power Management/ Clock Control
Quick Capture Interface
Internal SRAM
LCD Controller
System Bus
Intel® Wireless MMX™
Intel XScale® Core
Debug Controller
USB Host Controller
13 MHz Osc
32.768 kHz Osc
JTAG
Memory Controller
Address and Data
Variable Latency I/O Control
PC Card/ CompactFlash Control
Dynamic Memory Control
Static Memory Control
Address and Data Bus
ASIC
XCVR
Socket 0
Socket 1
SDRAM/ Boot ROM
ROM/ Flash/ SRAM

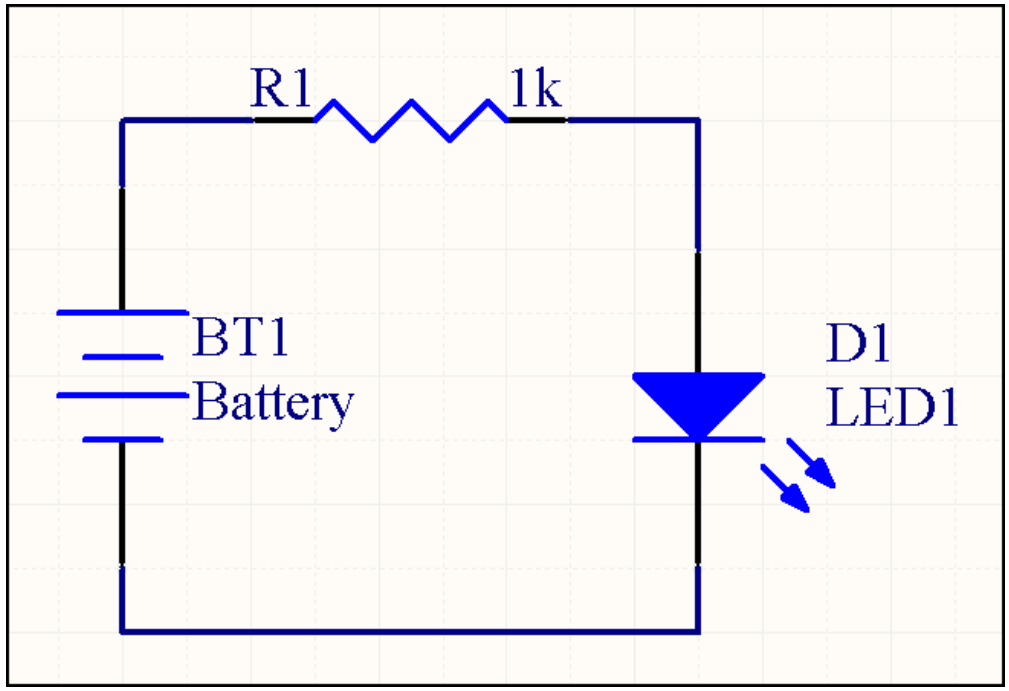
DESIGN


Video out
Video In
Audio I/O
MISC I/O
Video SDRAM
MPEG Codec
CPU
LCD
Ethernet
Compact Flash
SDRAM
Flash

SCHEMATICS

Schematics:

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

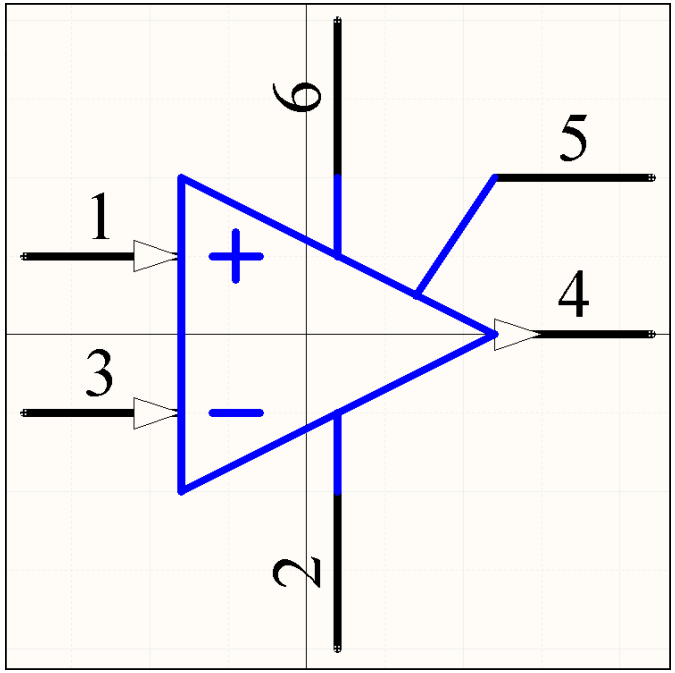


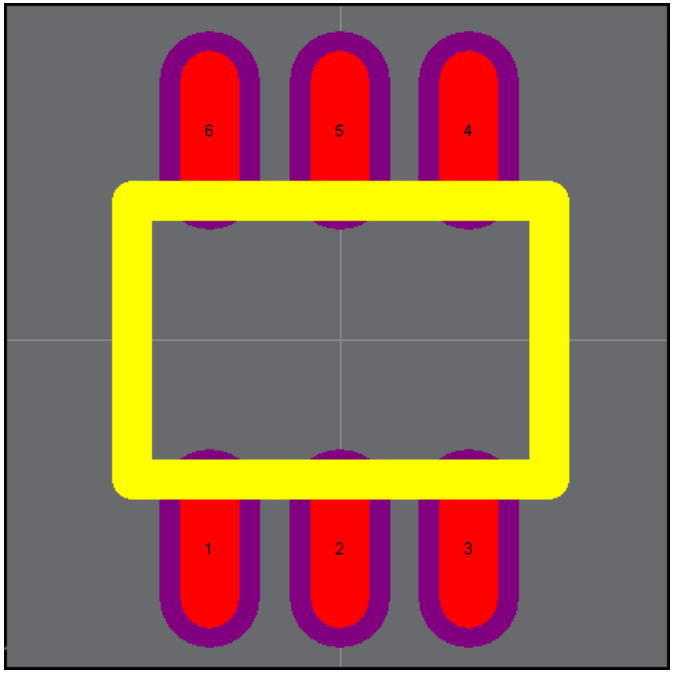
SCHEMATICS

Defining components:

① schematic component

② footprint



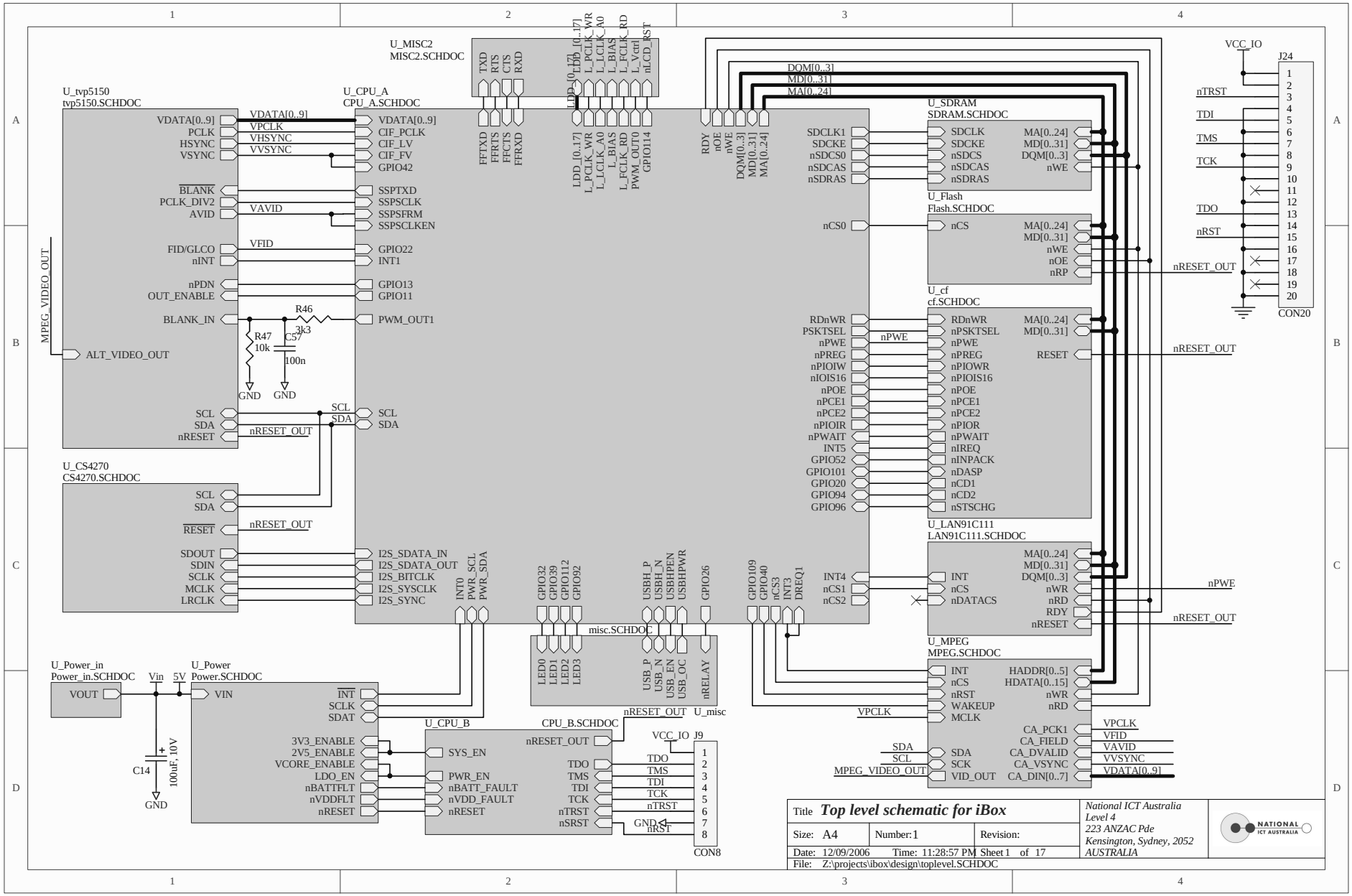


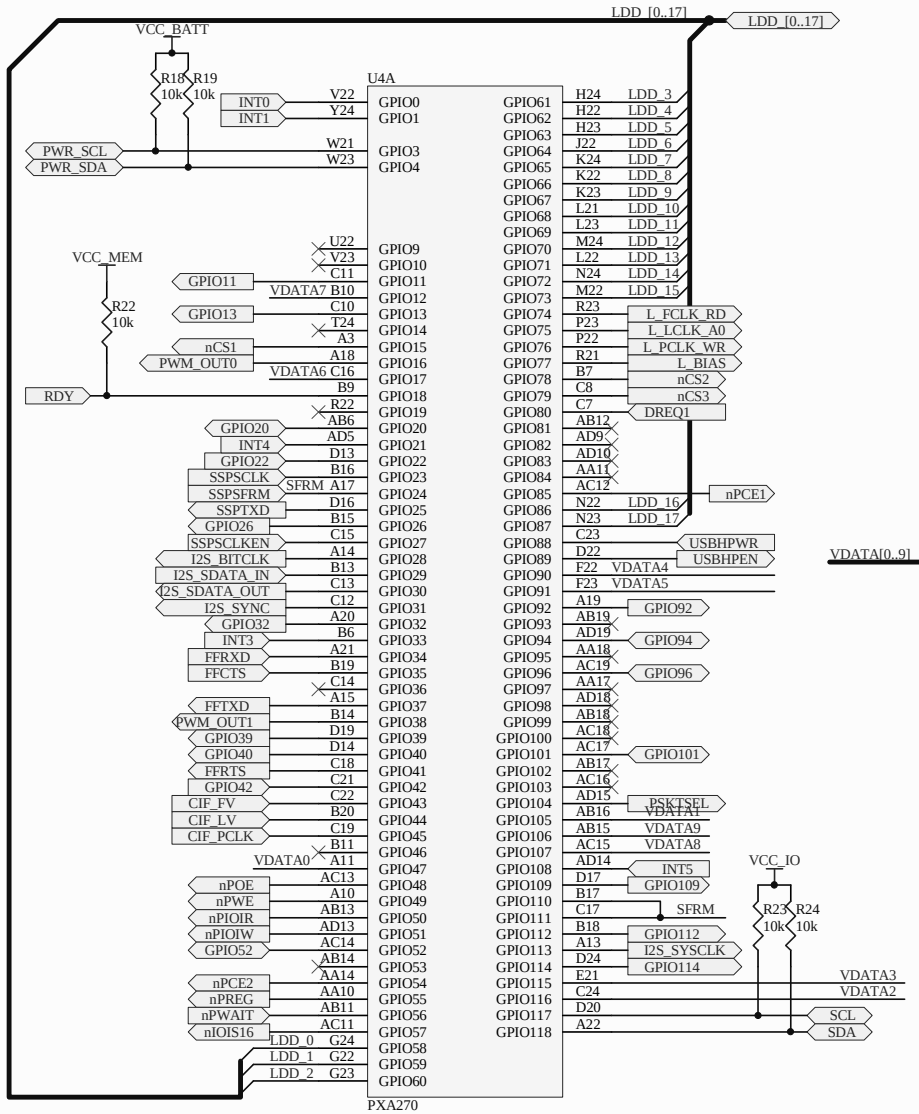
SCHEMATICS

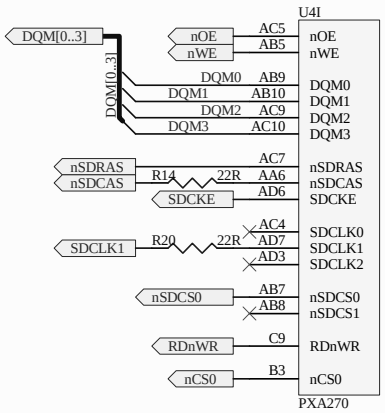
Things to think about:

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

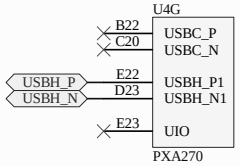
Hardware should be designed to make software elegant

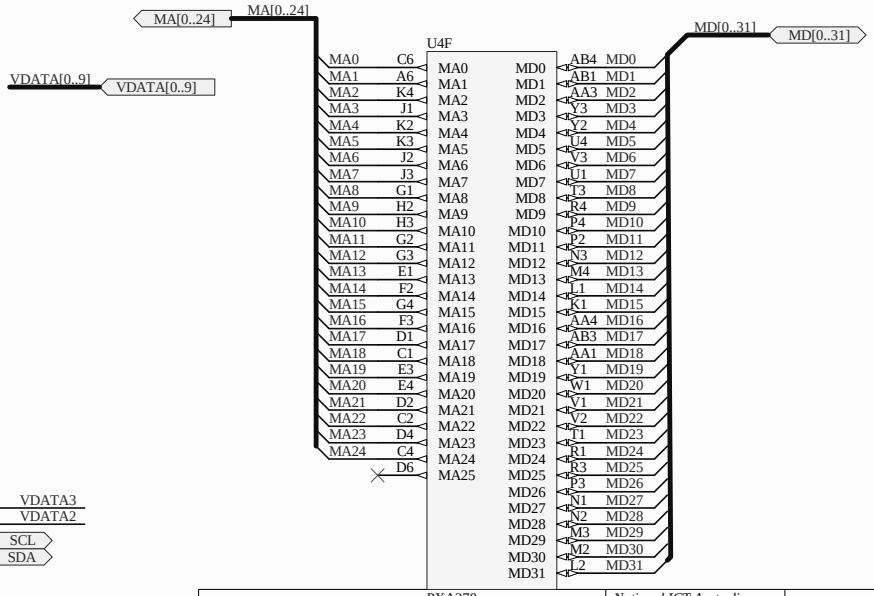

U_MISC2
MISC2.SCHDOC
U_tvp5150
tvp5150.SCHDOC
U_CPU_A
CPU_A.SCHDOC
U_SDRAM
SDRAM.SCHDOC
U_Flash
Flash.SCHDOC
U_cf
cf.SCHDOC
U_LAN91C111
LAN91C111.SCHDOC
U_MPEG
MPEG.SCHDOC
U_CS4270
CS4270.SCHDOC
U_Power_in
Power_in.SCHDOC
U_Power
Power.SCHDOC
U_CPU_B
CPU_B.SCHDOC
U_misc
misc.SCHDOC
J24
CON20
J9
CON8
VCC_IO
MPEG_VIDEO_OUT
R46
3k3
R47
10k
C57
100n
C14
100uF, 10V
GND
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
NATIONAL ICT AUSTRALIA





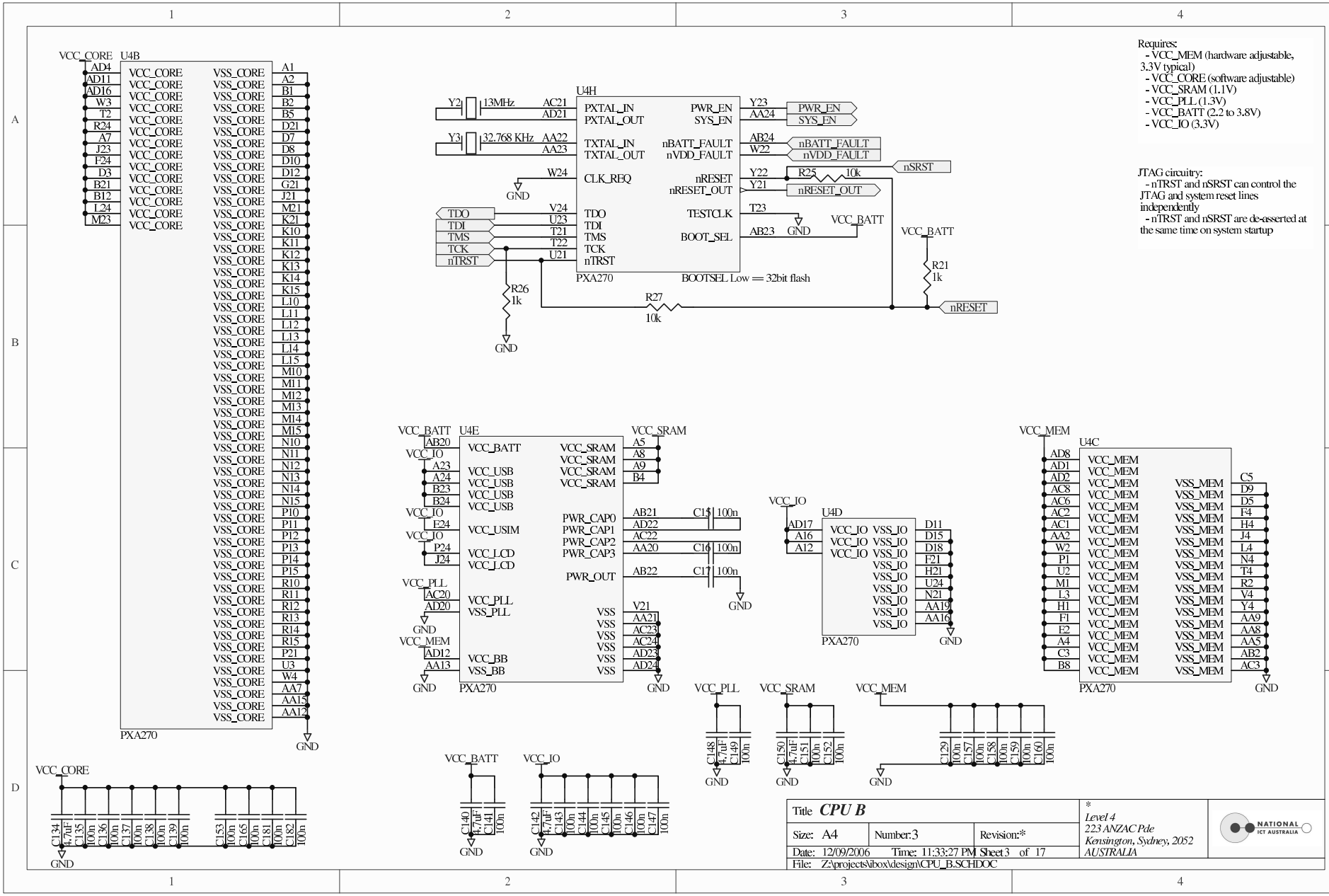
Design guide recommends: 20 ohm eries termination is required on the SDCAS and SDCLK lines. Not required on other lines. Requires a "Balanced T"

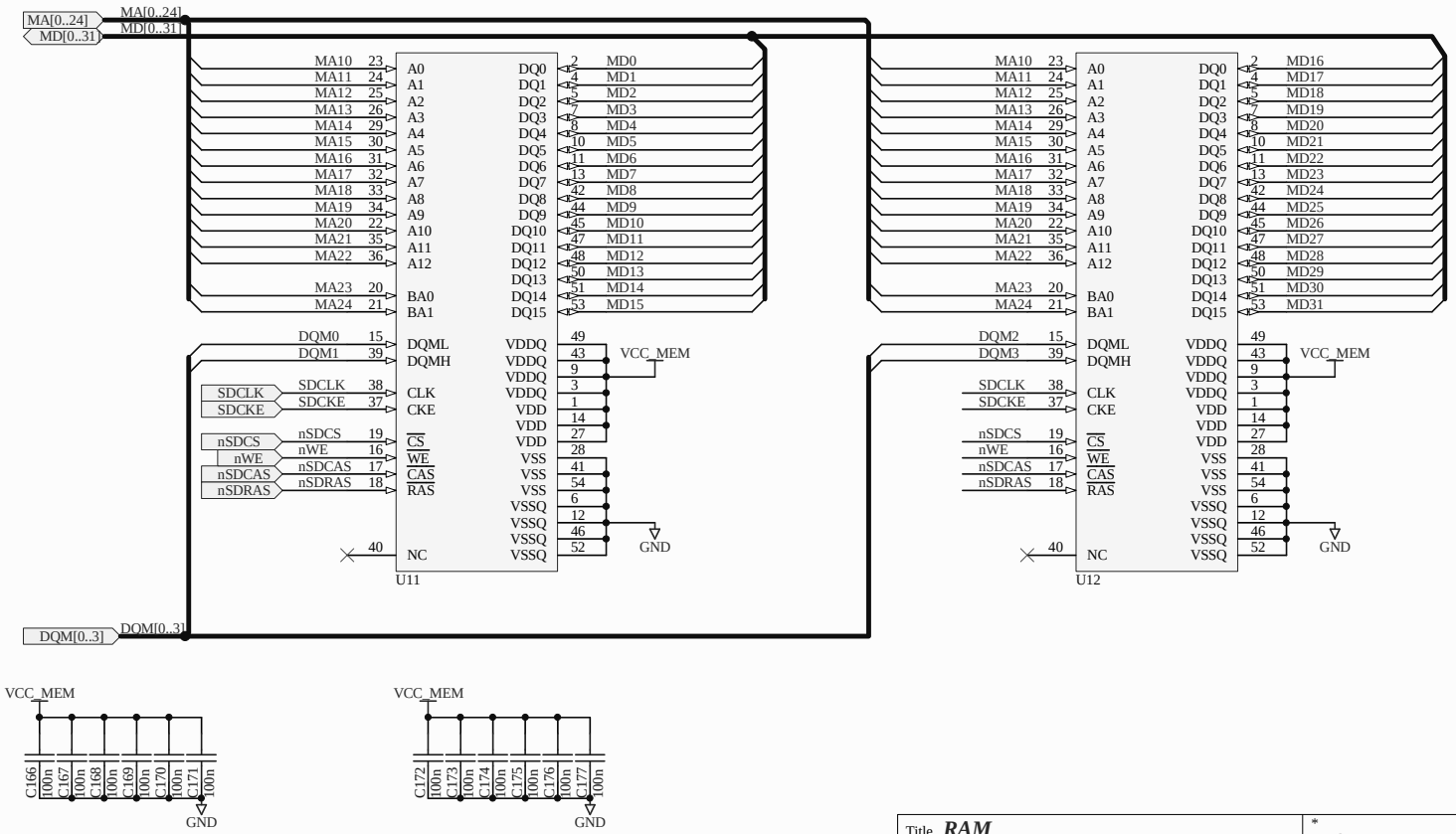




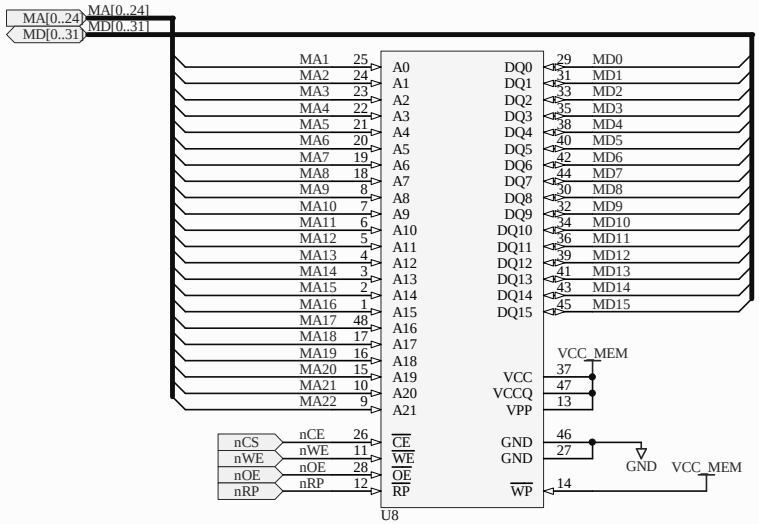
B17 is connected to SFRM to ease layout

Title	*CPU A*		*National ICT Australia* *Level 4* *223 ANZAC Pde* *Kensington, Sydney, 2052* *AUSTRALIA*
Size: A4	Number: 2	Revision: *	
Date: 12/09/2006	Time: 11:33:37 PM	Sheet 2 of 17	
File: Z:\projects\ibox\design\CPU_A.SCHDOC			


U4B
PXA270
U4H
PXA270
BOOTSEL Low = 32bit flash
U4E
PXA270
U4D
PXA270
U4C
PXA270
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
Size: A4
Number: 3
Revision: *
Date: 12/09/2006
Time: 11:33:27 PM
Sheet 3 of 17
File: Z:\projects\ibox\design\CPU_B.SCHDOC
Level 4
223 ANZAC Pde
Kensington, Sydney, 2052
AUSTRALIA
NATIONAL ICT AUSTRALIA



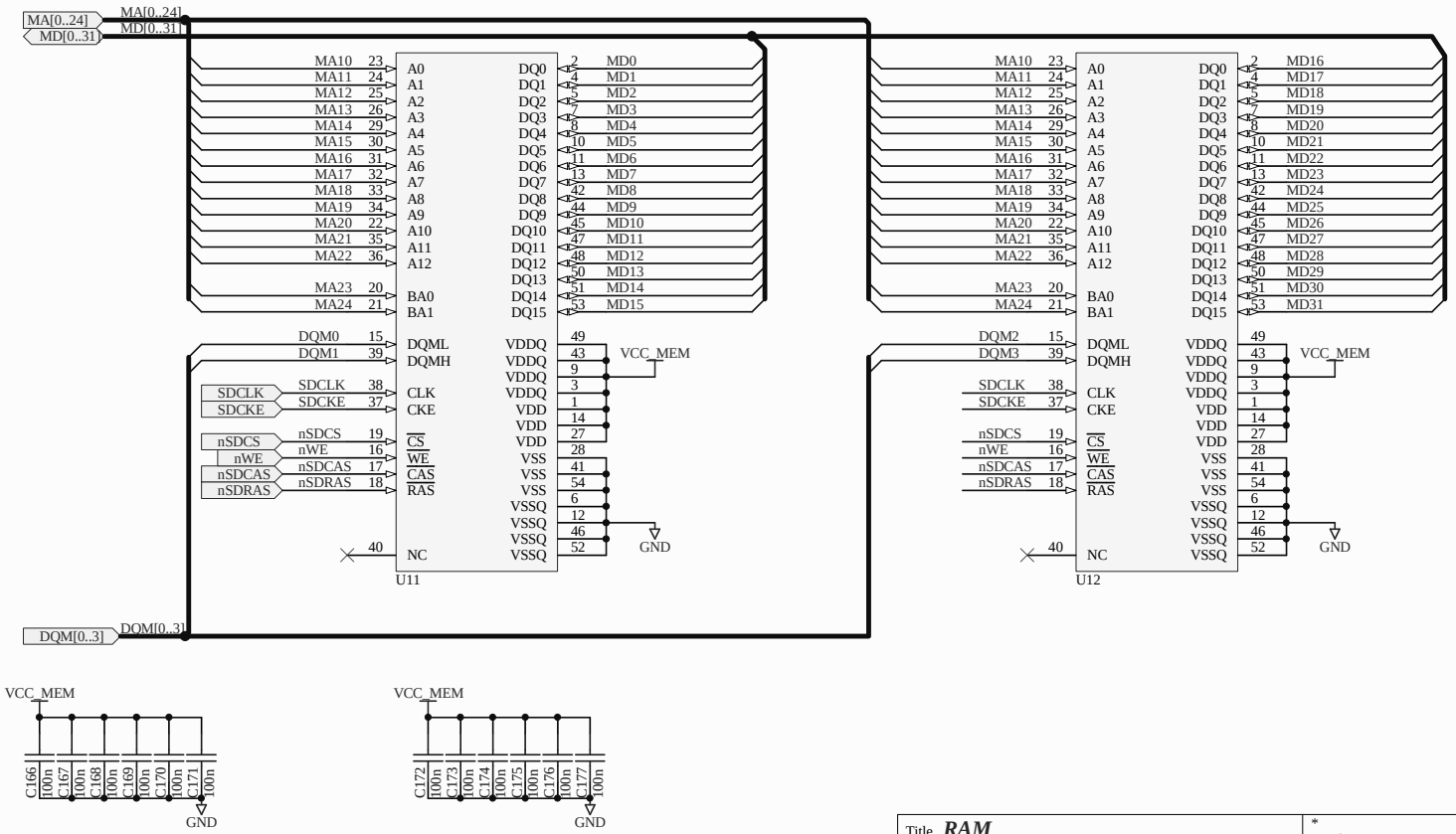
Title *RAM*			* Level 4 223 ANZAC Pde Kensington, Sydney, 2052 AUSTRALIA
Size: A4	Number: 4	Revision: *	
Date: 12/09/2006	Time: 11:37:29 PM	Sheet 4 of 17	
File: Z:\projects\ibox\design\SDRAM.SCHDOC			



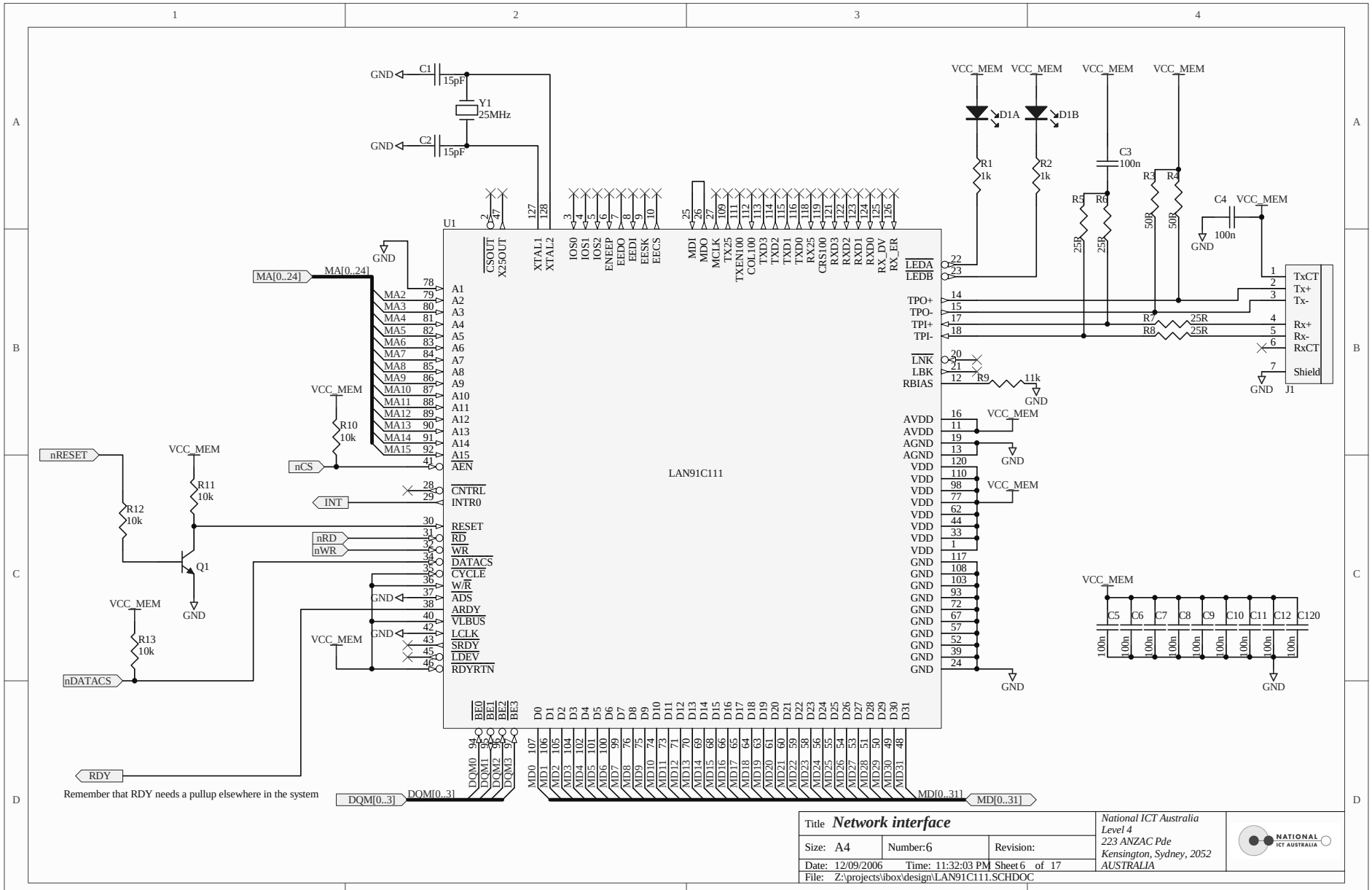
Title	***Flash memory***		
Size: A4	Number:5	Revision:*	
Date: 12/09/2006	Time: 11:37:37 PM	Sheet 5 of 17	
File: Z:\projects\ibox\design\Flash.SCHDOC			

*
Level 4
223 ANZAC Pde
Kensington, Sydney, 2052
AUSTRALIA

NATIONAL ICT AUSTRALIA

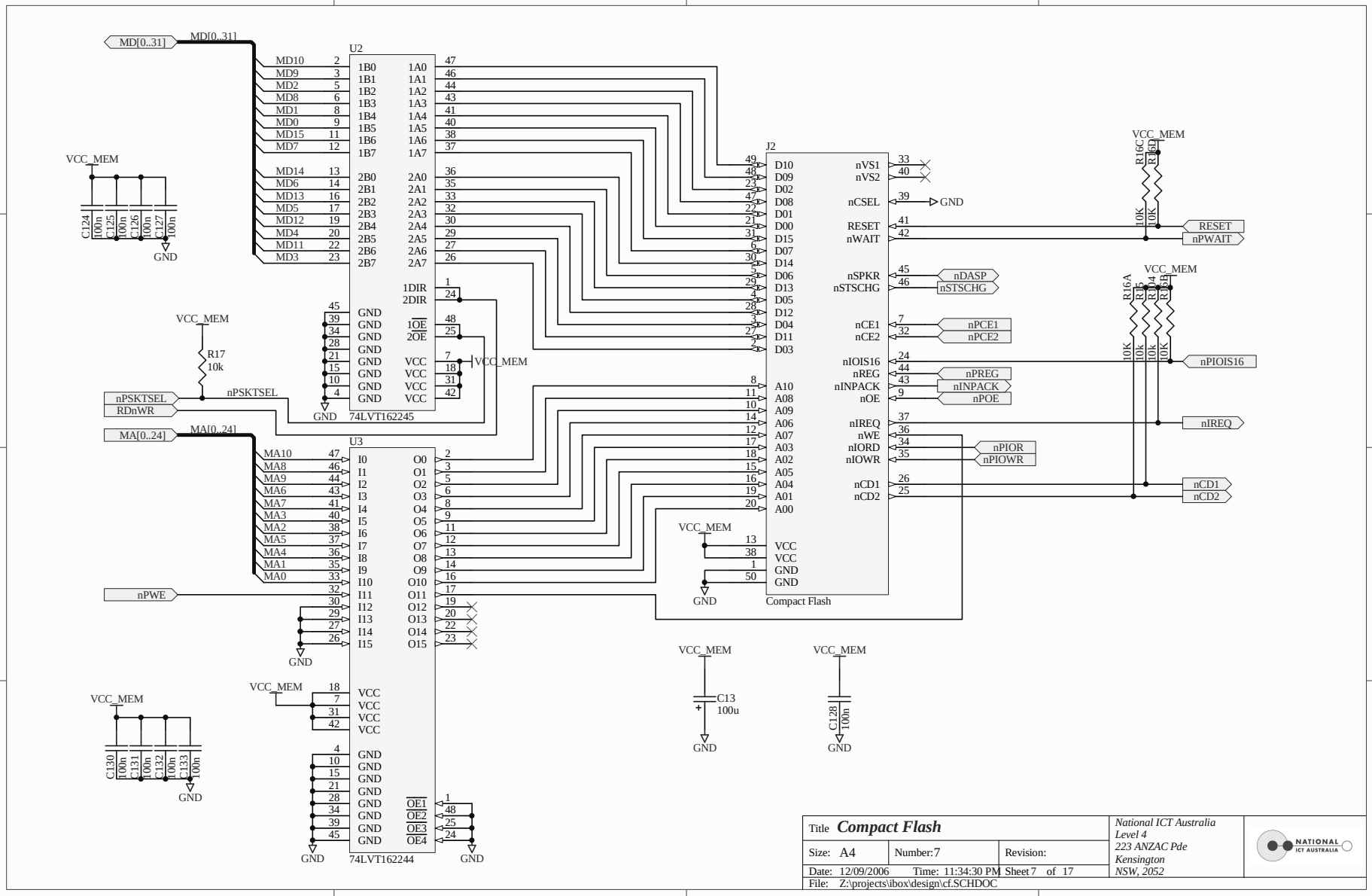


Title *RAM*			* Level 4 223 ANZAC Pde Kensington, Sydney, 2052 AUSTRALIA
Size: A4	Number: 4	Revision: *	
Date: 12/09/2006	Time: 11:37:29 PM	Sheet 4 of 17	
File: Z:\projects\ibox\design\SDRAM.SCHDOC			

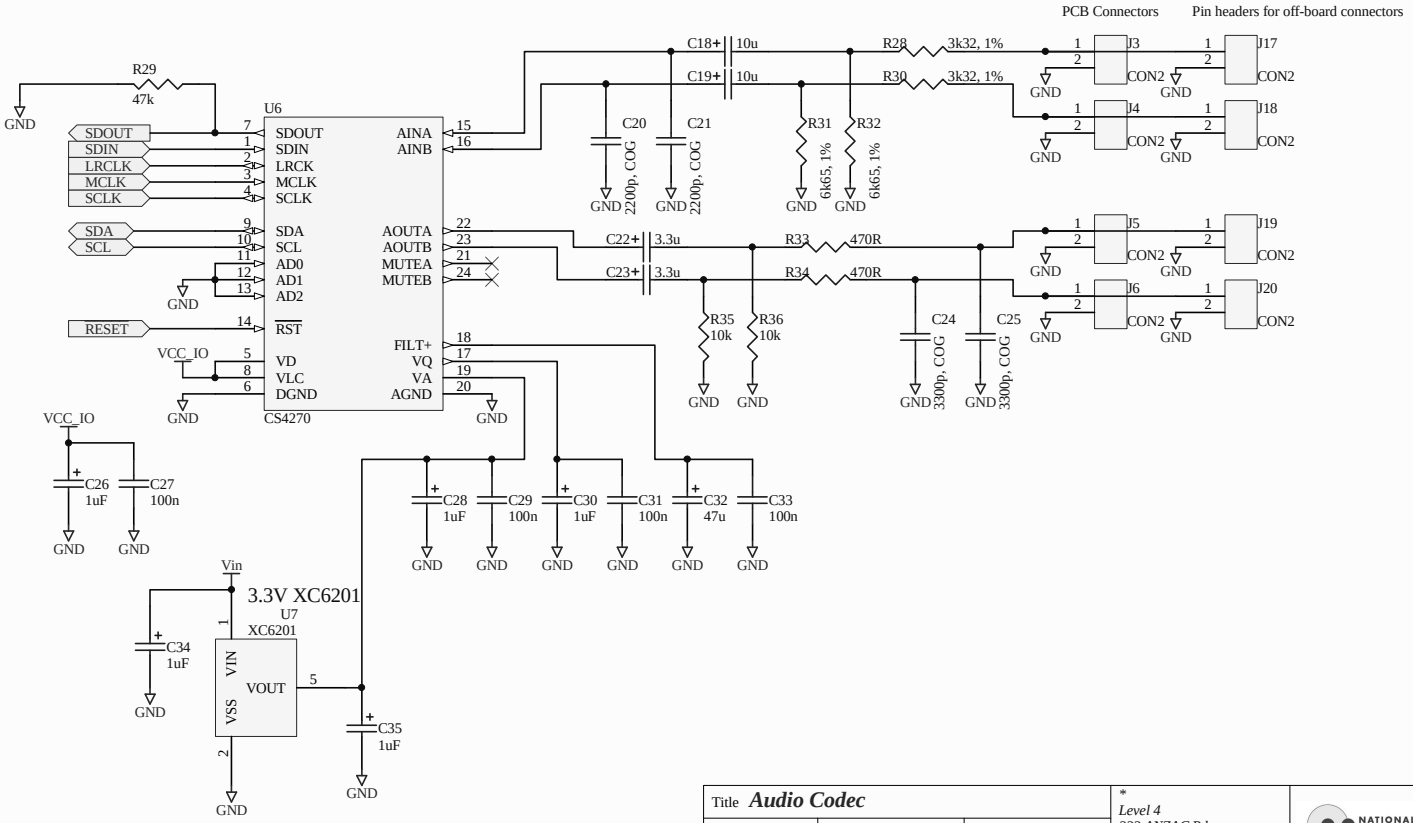


Title	***Network interface***		*National ICT Australia*
Size: A4	Number:6	Revision:	*Level 4*
Date: 12/09/2006	Time: 11:32:03 PM	Sheet 6 of 17	*223 ANZAC Pde*
File: Z:\projects\ibox\design\LAN91C111.SCHDOC			*Kensington, Sydney, 2052*
			AUSTRALIA

NATIONAL ICT AUSTRALIA

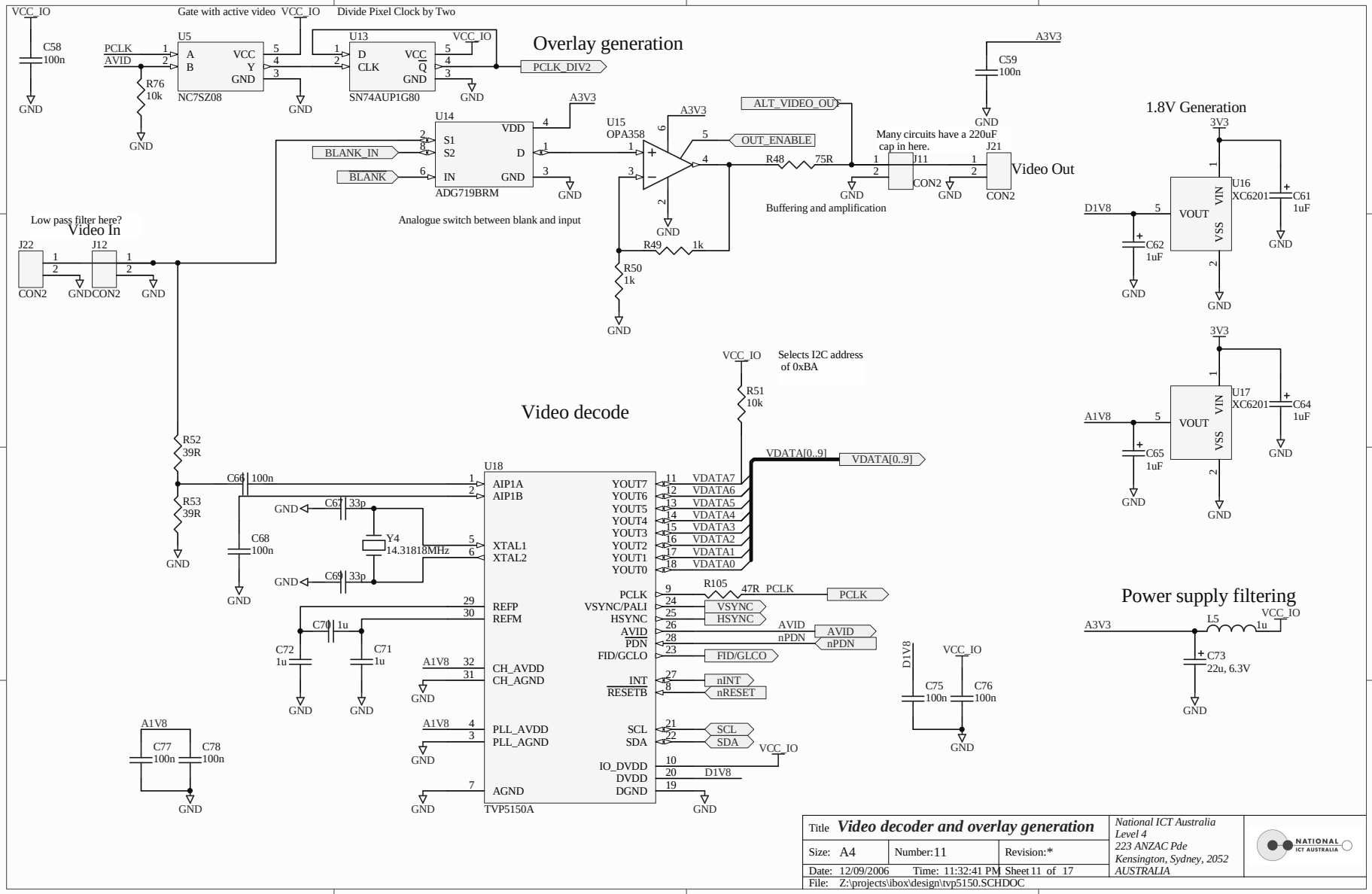

MD[0..31]
MD[0..31]
U2
74LVT162245
VCC_MEM
GND
R17
10k
nPSKTSEL
RDnWR
MA[0..24]
MA[0..24]
U3
74LVT162244
nPWE
J2
Compact Flash
RESET
nPWAIT
nDASP
nSTSCHG
nPCE1
nPCE2
nPIOIS16
nPREG
nINPACK
nPOE
nIREQ
nPIOR
nPIOWR
nCD1
nCD2
R16G
R16A
C13
100u
C128
100n
Title Compact Flash
Size: A4
Number: 7
Revision:
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

Selects master mode



Title **Audio Codec**			* Level 4 223 ANZAC Pde Kensington, Sydney, 2052 AUSTRALIA
Size: A4	Number:8	Revision:*	
Date: 12/09/2006	Time: 11:37:47 PM	Sheet 8 of 17	
File: Z:\projects\ibox\design\CS4270.SCHDOC			

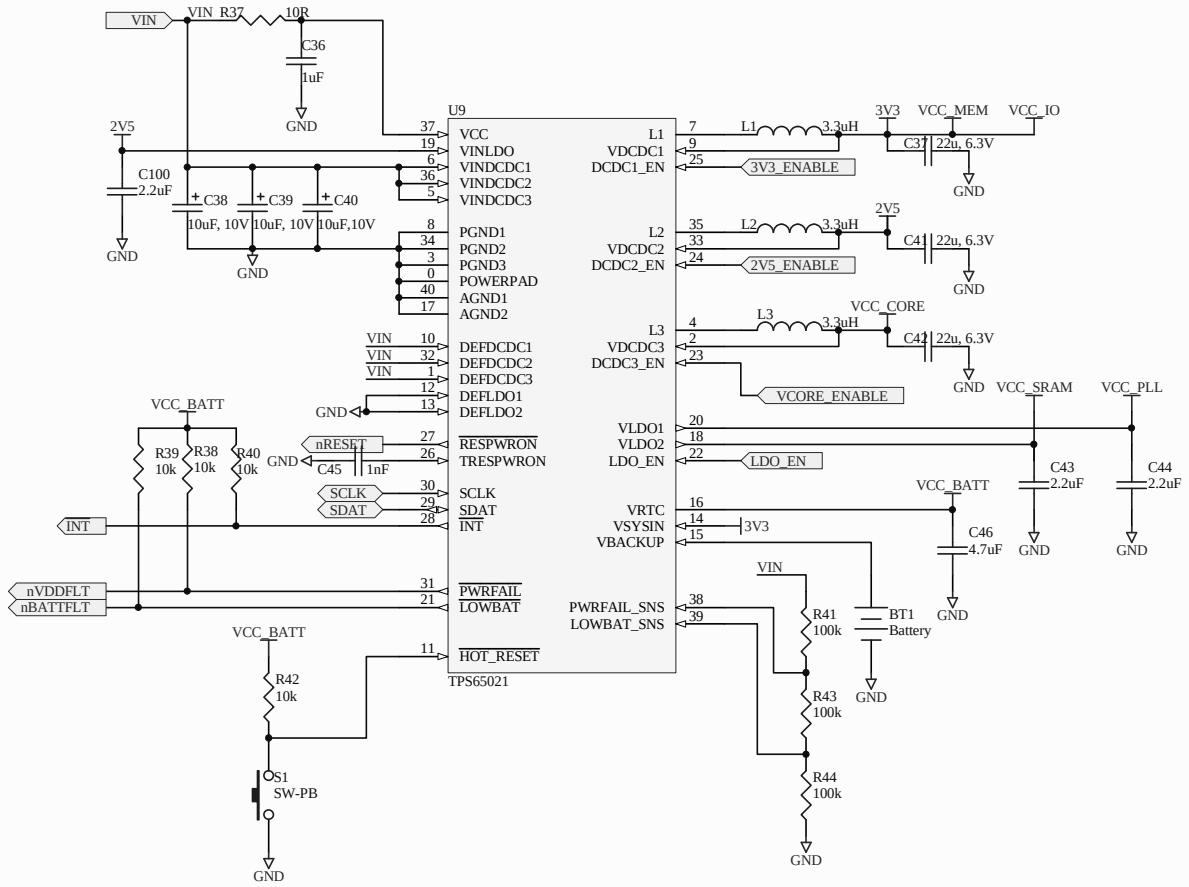
NATIONAL ICT AUSTRALIA



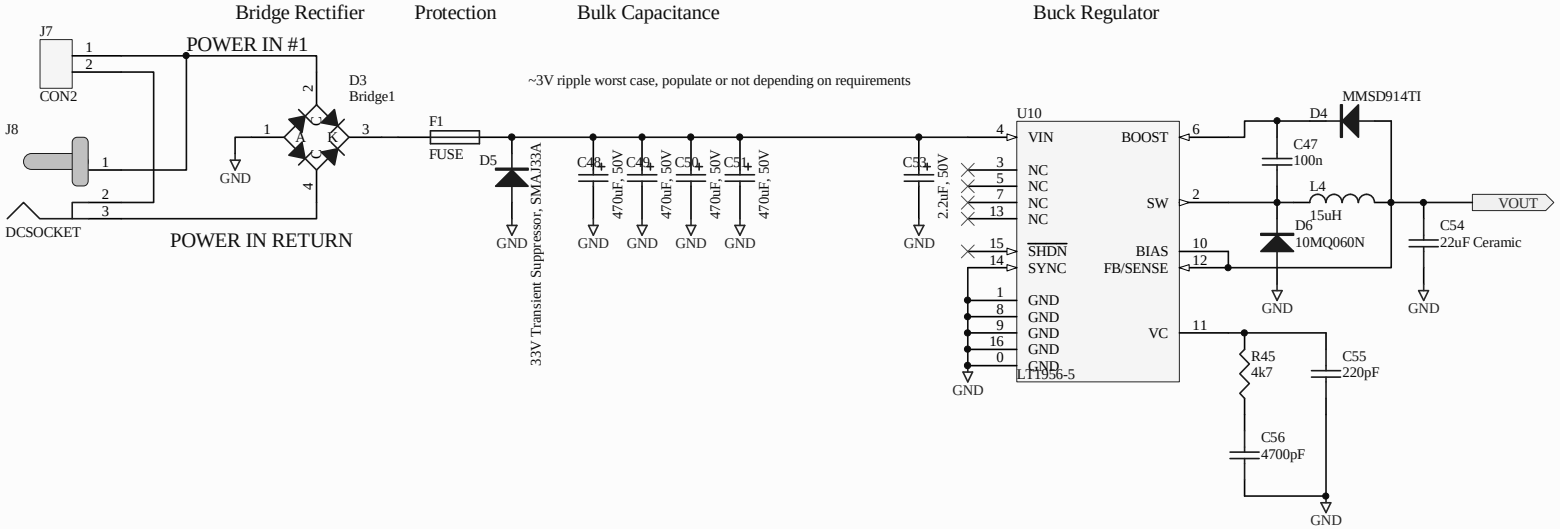
Title	***Video decoder and overlay generation***	*National ICT Australia*
Size: A4	Number: 11	Revision: *
Date: 12/09/2006	Time: 11:32:41 PM	Sheet 11 of 17
File: Z:\projects\ibox\design\tvp5150.SCHDOC		

Level 4
223 ANZAC Pde
Kensington, Sydney, 2052
AUSTRALIA

NATIONAL ICT AUSTRALIA



Title **Power supplies**			*National ICT Australia* *Level 4* *223 ANZAC Pde* *Kensington* *NSW, 2052*
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			

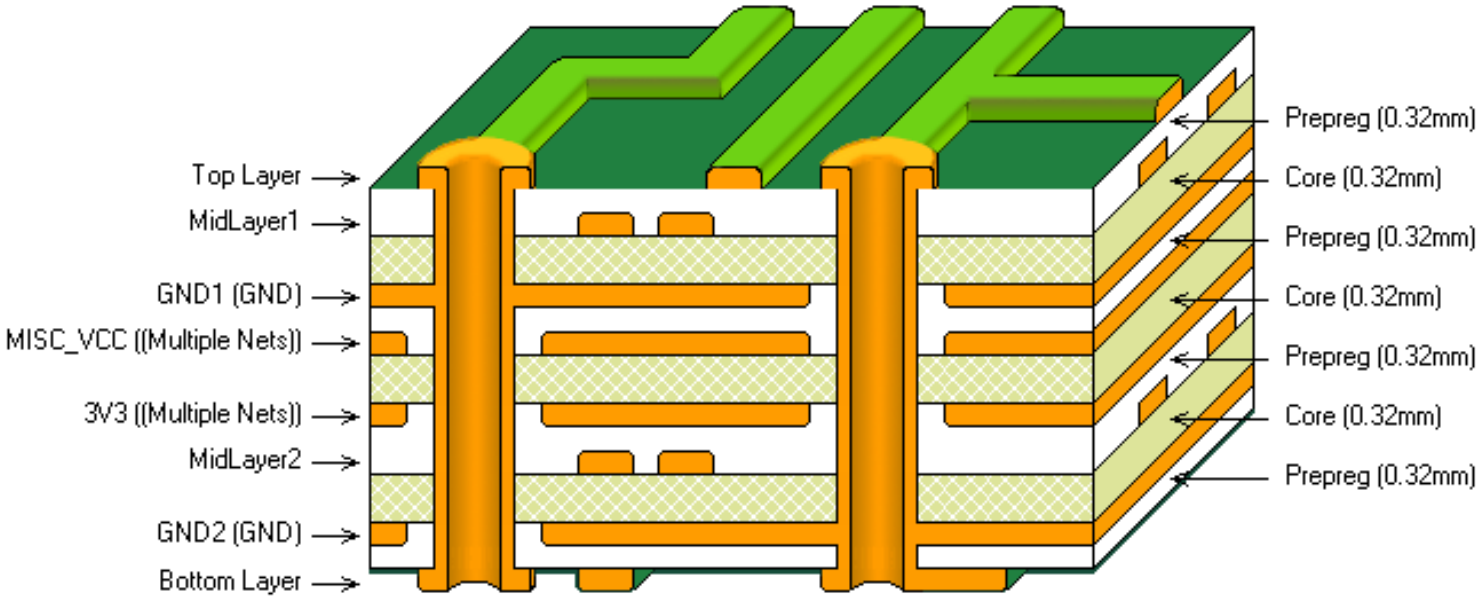


Title	***Power input and conditioning***		*National ICT Australia* *Level 4* *223 ANZAC Pde* *Kensington* *NSW, 2052*
Size: A4	Number: 10	Revision:	
Date: 12/09/2006	Time: 11:34:46 PM	Sheet 10 of 17	
File: Z:\projects\ibox\design\Power_in.SCHDOC			

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

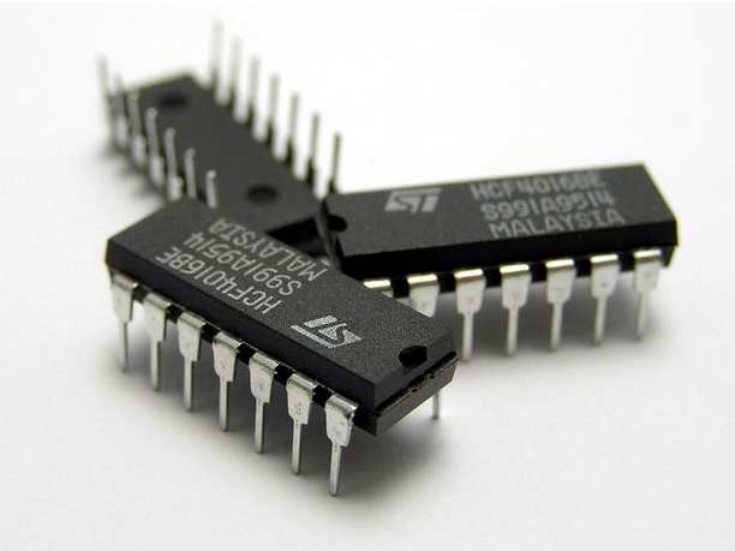


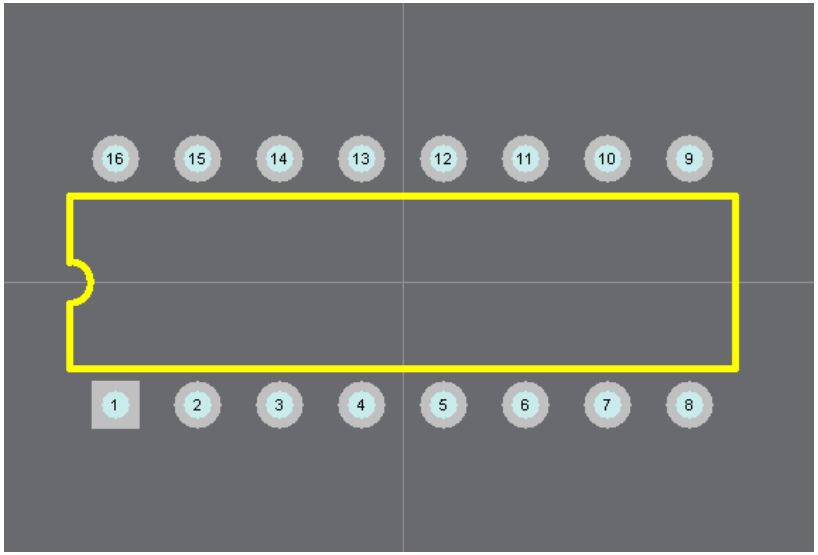
LAYOUT

Component footprints:

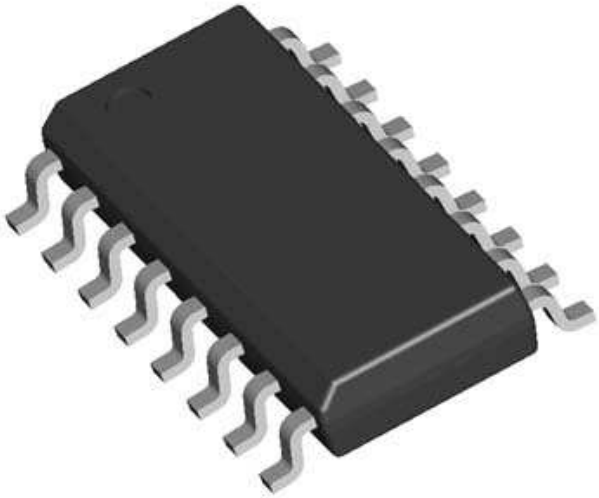
- through-hole
- surface-mount

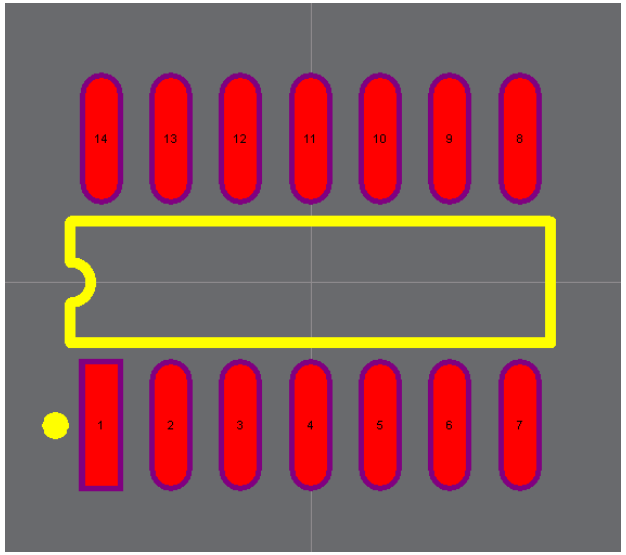
LAYOUT


HCF4016BE
S991A995I4
MALAYSIA

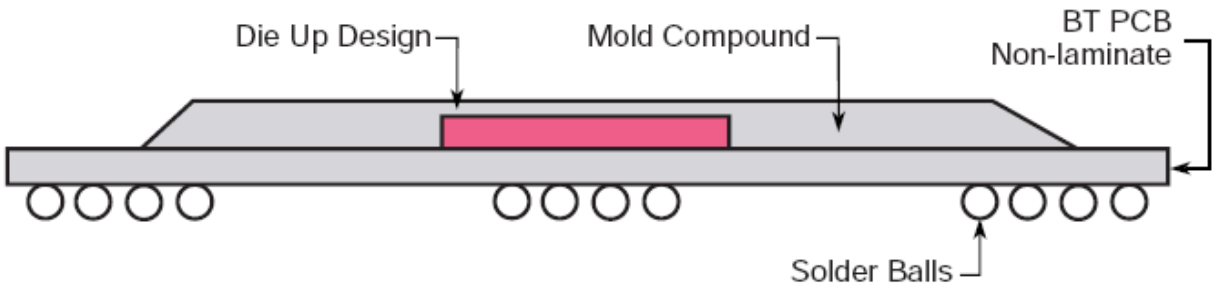

16
15
14
13
12
11
10
9
1
2
3
4
5
6
7
8

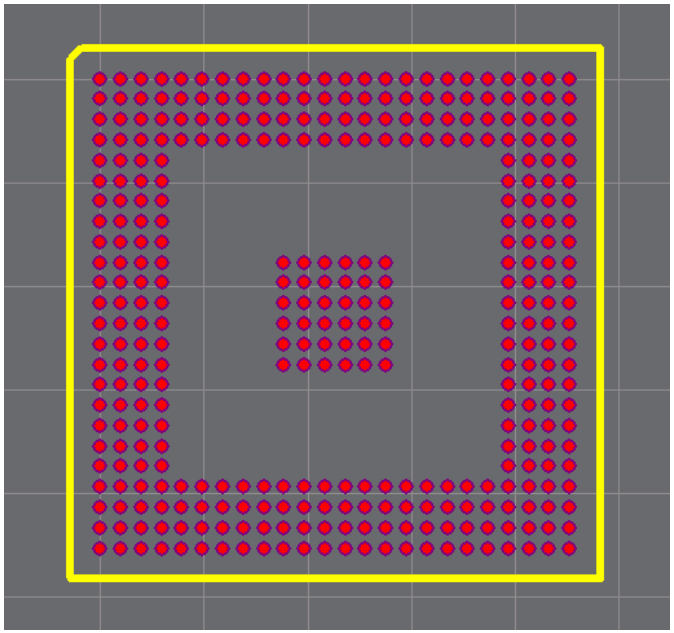
LAYOUT





LAYOUT


Die Up Design
Mold Compound
BT PCB
Non-laminate
Solder Balls



LAYOUT

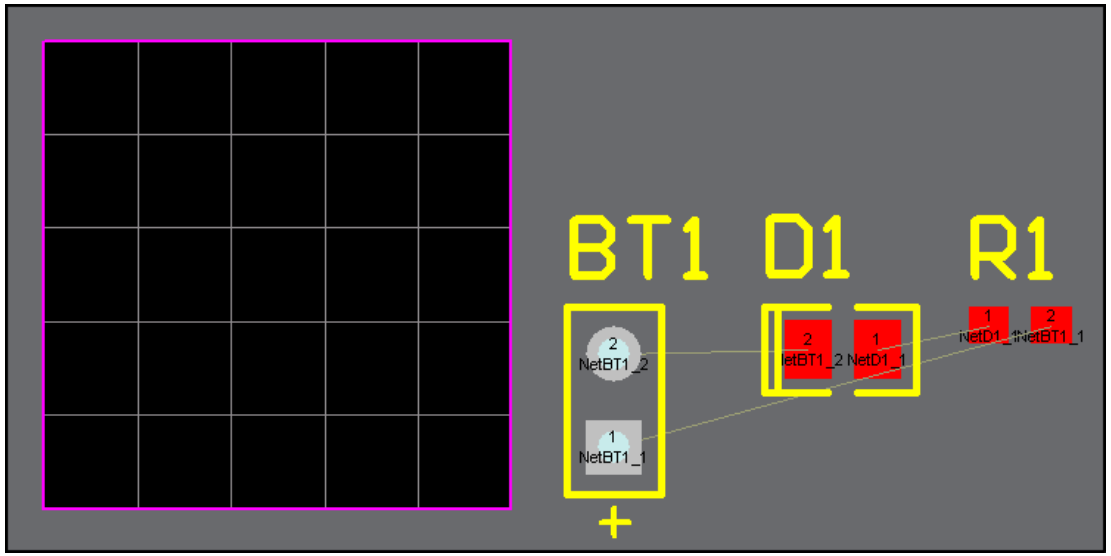
Trade-offs:

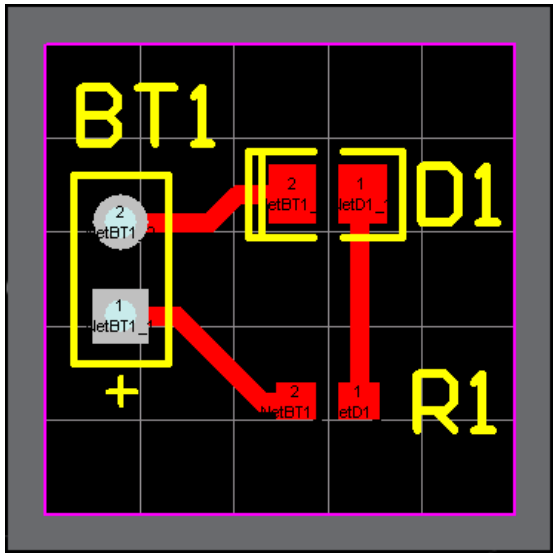
- ➜ number of layers vs. size
- ➜ component size vs. manufacturability
- ➜ component size vs. signal integrity

LAYOUT

Routing:

- ➜ the physical pins are connected with traces of copper
- ➜ the design software only allows connections defined by the schematic
- ➜ design rules are defined to ensure manufacturability
- ➜ 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

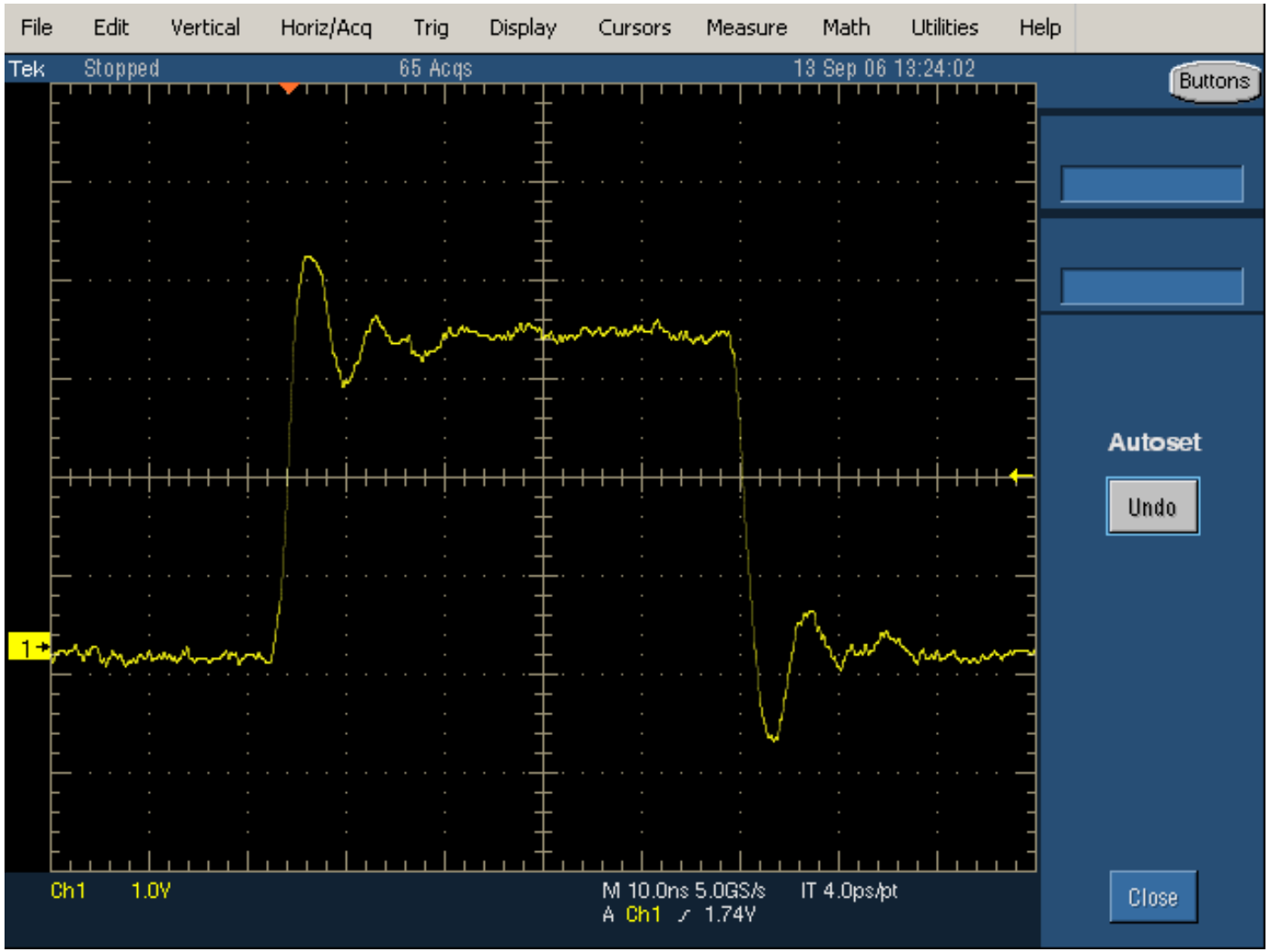
Signal integrity: Digital systems are still analogue!

- ➜ 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

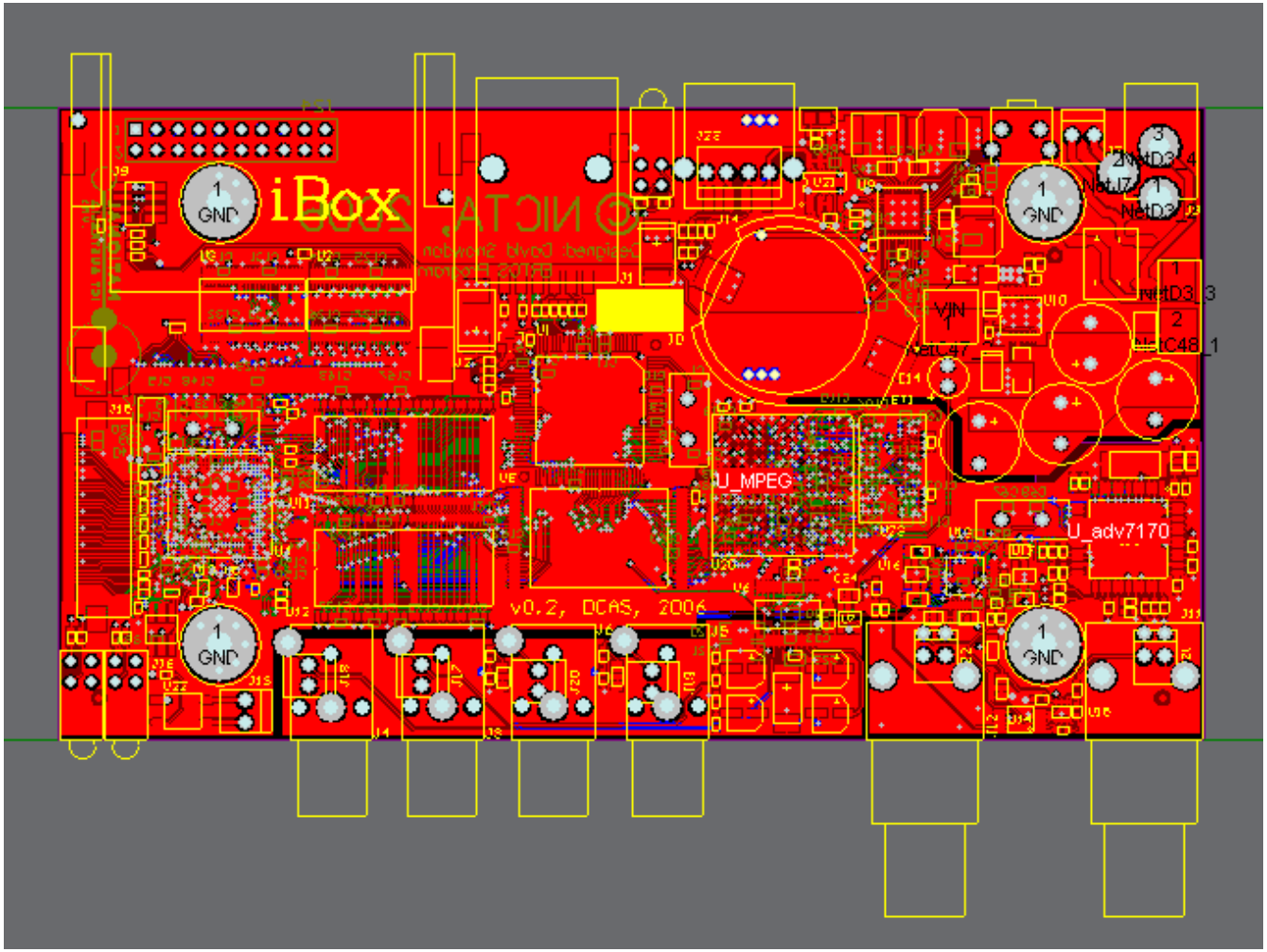

File
Edit
Vertical
Horiz/Acq
Trig
Display
Cursors
Measure
Math
Utilities
Help
Tek
Stopped
65 Acqs
13 Sep 06 13:24:02
Buttons
Autoset
Undo
Close
1
Ch1
1.0V
M 10.0ns 5.0GS/s
IT 4.0ps/pt
A Ch1
1.74V

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:

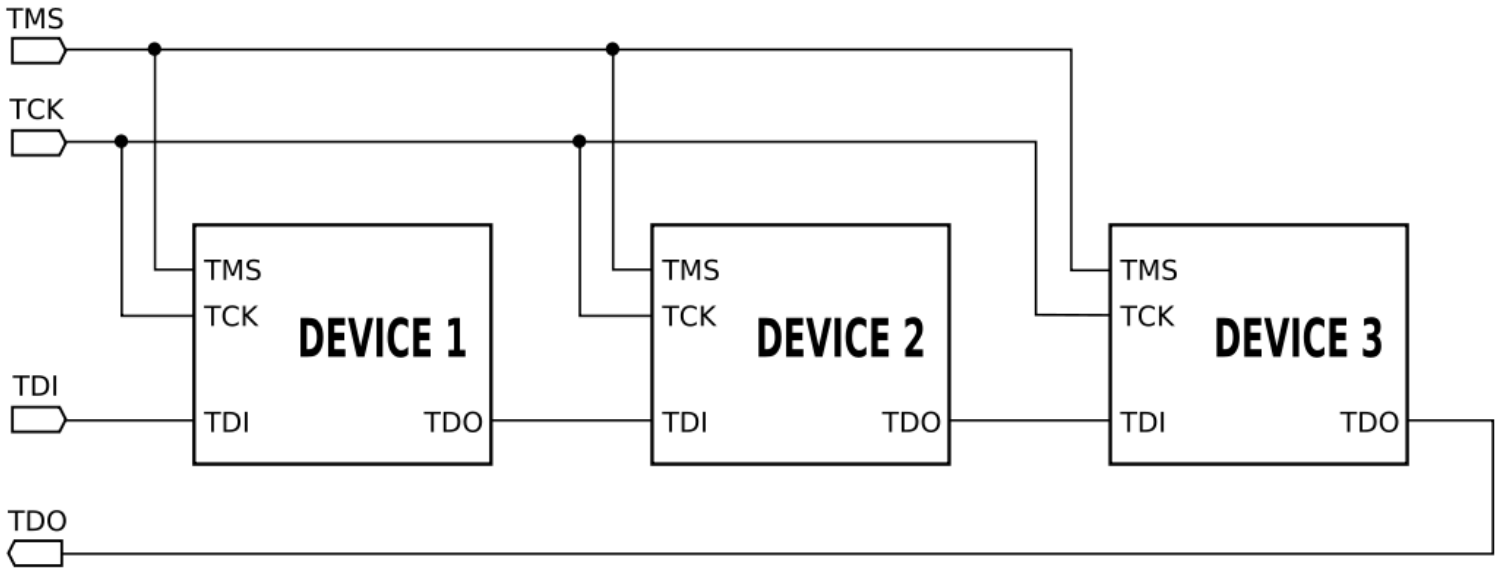
- wave soldering

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

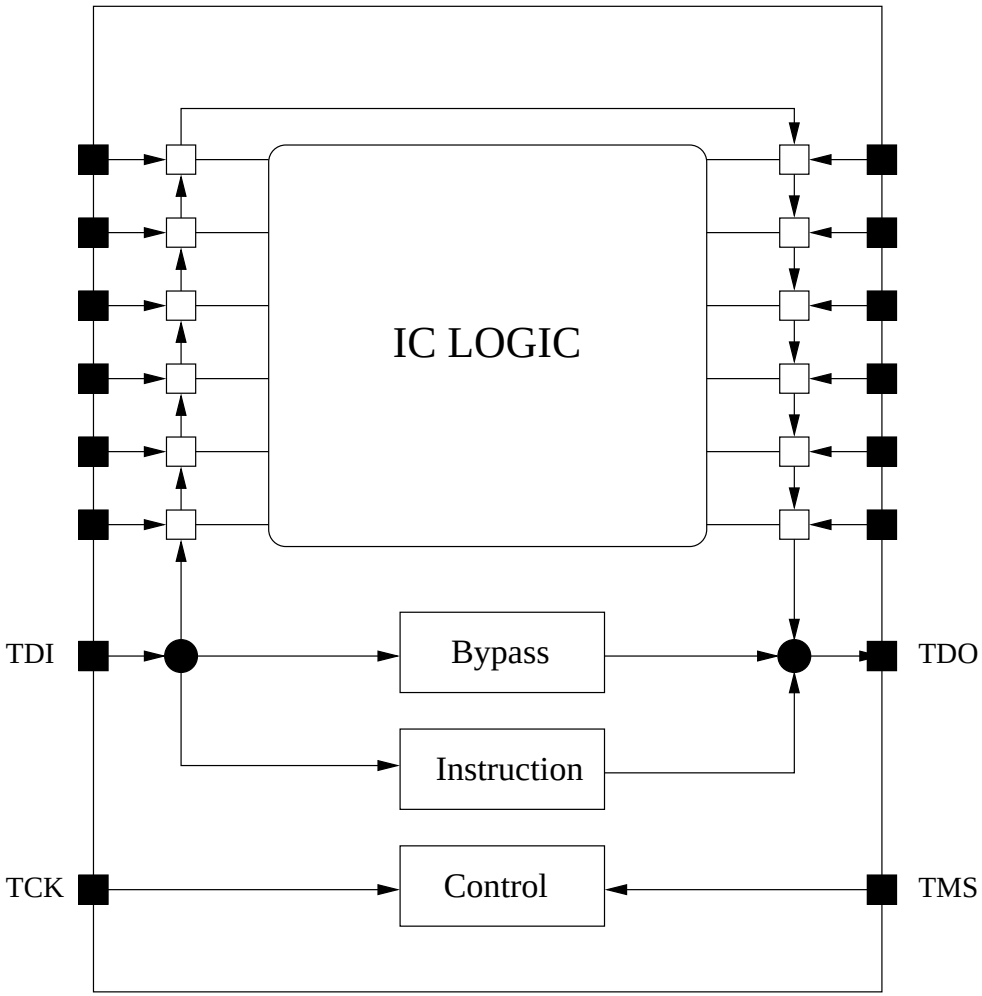


BRING-UP

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 operating system
- ➜ 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 NSLU2 uses Redboot

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

Porting an OS to the new platform:

- configure for PXA270
- configure specific devices (network, CF, USB, etc)

BRING-UP

Bringing up the iBox:

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

DEMO

DEMO

POWER MANAGEMENT

aka: doing more with less

Power Management

Overview:

① Why?

② Concepts

③ OS directed power management

④ Evaluation techniques

⑤ Energy accounting

Power Management

Why do we care about power?

- ➜ Thermal management
- ➜ Electricity consumption
- ➜ Energy limited devices

Power-conscious systems can use energy more effectively.

Power Management

Why do we care about power?

- ➜ Thermal management
  - ① Enclosure size and noise
  - ② Performance
- ➜ Electricity consumption

- ➜ Energy limited devices

Power-conscious systems can use energy more effectively.

Power Management

Why do we care about power?

- Thermal management
  1. Enclosure size and noise
  2. Performance
- Electricity consumption
  1. Environment
  2. Cost
- Energy limited devices

Power-conscious systems can use energy more effectively.

Power Management

Why do we care about power?

- ➜ Thermal management
  - ① Enclosure size and noise
  - ② Performance
- ➜ Electricity consumption
  - ① Environment
  - ② Cost
- ➜ Energy limited devices
  - ① Battery powered
  - ② Solar powered

Power-conscious systems can use energy more effectively.

Causes of energy use

Where does the energy go?:

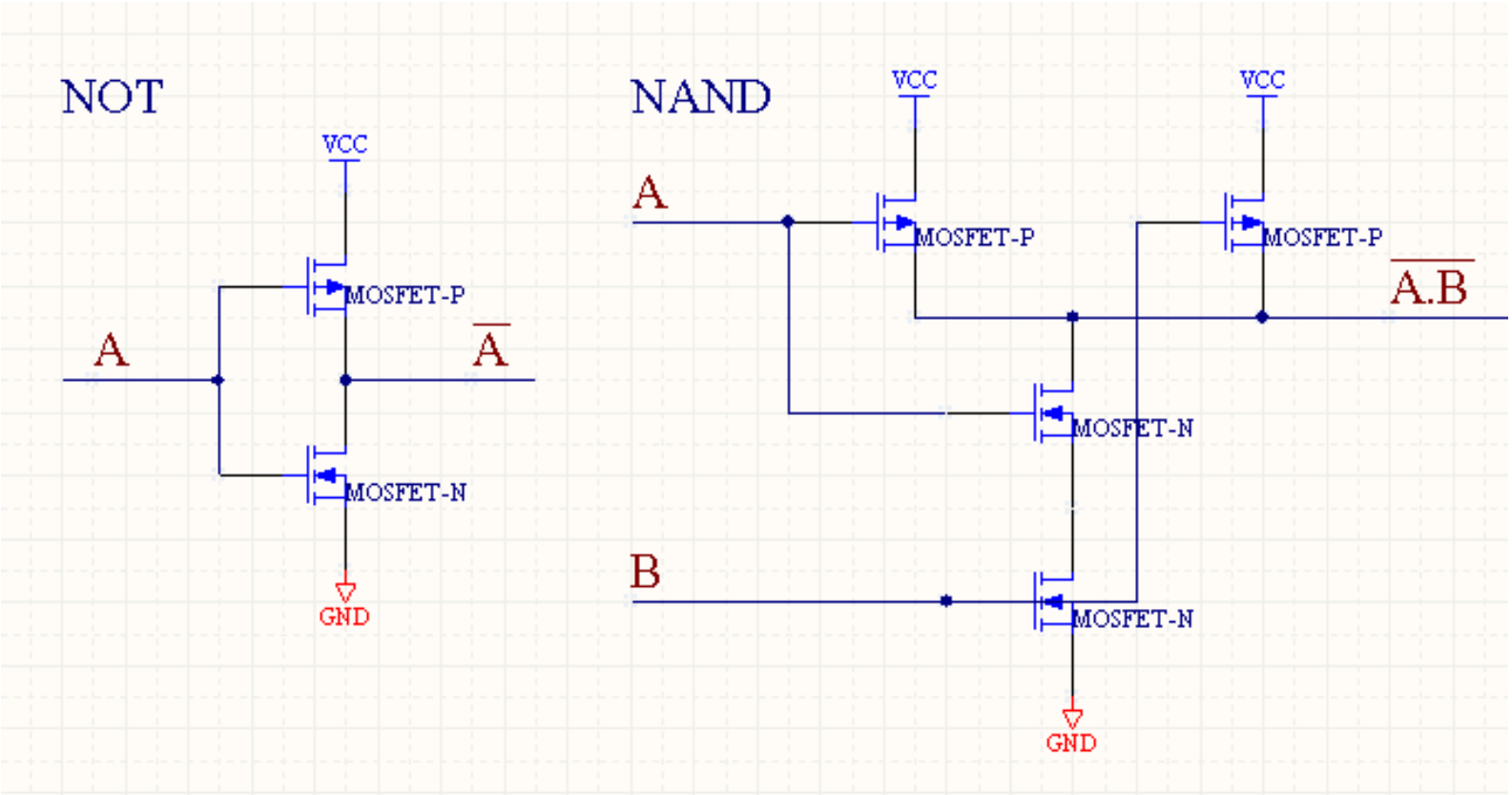
- switching circuits
- peripherals
- voltage conversion
- batteries

Causes of energy use

Switching circuits:

- ➜ short circuit power
- ➜ gate capacitance
- ➜ leakage

CAUSES OF ENERGY USE



Causes of energy use

Gate capacitance:

- to turn on a transistor, we charge the gate beyond its threshold voltage
- to turn it off, we discharge it
- the charge disappears and is the main source of power consumed
- the charge transferred is: $q = Cv$
- the energy transferred is: $E = \frac{1}{2}Cv^2$

Causes of energy use

Hardware techniques:

- clock gating

Causes of energy use

Software energy consumption:

- ➜ more instructions = more energy
- ➜ different instructions use different energy
- ➜ different data use different energy

"Source Code Optimisation and Profiling of Energy Consumption in Embedded Systems": Simunic, et al, 2000.

Dynamic voltage scaling

Dynamic frequency scaling:

- ➜ reduce the system frequency to reduce the power
- ➜ each switch we lose $E = \frac{1}{2}Cv^2$
- ➜ C comes from the gate and interconnect capacitance
- ➜ frequency is the rate at which we switch
- ➜ conclusion: power scales linearly with frequency

Dynamic voltage scaling

Dynamic frequency scaling:

- ➜ reduce the system frequency to reduce the power
- ➜ each switch we lose $E = \frac{1}{2}Cv^2$
- ➜ C comes from the gate and interconnect capacitance
- ➜ frequency is the rate at which we switch
- ➜ conclusion: power scales linearly with frequency

The energy required to complete a given amount of work is unchanged

Dynamic voltage scaling

Dynamic voltage scaling:

- ➜ if we reduce the frequency, the transistors don't have to switch as fast
- ➜ transistor speed is dependent on the driven voltage
- ➜ if we reduce the frequency, we can reduce the voltage
- ➜ given $E = \frac{1}{2}Cv^2$, we get quadratic energy savings

Dynamic voltage scaling

Voltage/frequency switching overheads:

- ➜ frequency generation is usually via a crystal and PLL
- ➜ clock multipliers control the PLL
- ➜ clock dividers generate lower frequencies
- ➜ the PLL has to *settle* following a change
- ➜ on the PXA255 (XScale) this is $\approx 500\mu s$
- ➜ or, always run the PLL at full speed and change dividers
- ➜ on the PXA255 this takes 20 cycles.

DYNAMIC VOLTAGE SCALING

`Scheduling for Reduced CPU Energy`: Weiser, et al, 1994.

- ➜ assumption: idleness is bad
- ➜ eliminate idleness using DVS
  - if idle time is large, decrease speed
  - if idle time is small, increase speed
  - adjust the speed periodically

Other algorithms:

- ➜ lots of ways of predicting idleness

DYNAMIC VOLTAGE SCALING

`Balancing Batteries, Power and Performance: System Issues in CPU Speed-Setting for Mobile Computing`: Thomas Martin, 2001

Peukert:

$$Q = \frac{k}{I^{\alpha}}$$

- ➜ battery capacity is related to rate of discharge
- ➜ steady discharge may increase the battery life
- ➜ can DVS be used to attain this?

Dynamic voltage scaling

QoS issues:

- ➜ reducing frequency reduces performance
- ➜ missed deadlines
- ➜ reduced QoS
- ➜ real-time techniques may be employed

DYNAMIC VOLTAGE SCALING

QoS issues:

- ➜ reducing frequency reduces performance
- ➜ missed deadlines
- ➜ reduced QoS
- ➜ real-time techniques may be employed

every man and his dog has an RT DVS technique

Dynamic voltage scaling

Trouble in wonderland:

- ignoring leakage, transistors only use power when they switch
- not every part of the circuit switches on every clock cycle
- different sections run at different frequencies
- off-chip devices aren't aren't affected

Dynamic voltage scaling

Trouble in wonderland:

- ignoring leakage, transistors only use power when they switch
- not every part of the circuit switches on every clock cycle
- different sections run at different frequencies
- off-chip devices aren't aren't affected

Also:

- We can't ignore leakage!

The simple models only work in special cases

DVS: Myths and Facts

"Power Management and Dynamic Voltage Scaling: Myths and Facts": Snowdon, et al, 2005.

- ➜ most academic research uses simulation for evaluation
- ➜ the naive models lead to incorrect results.

DVS: Myths and facts

"Power Management and Dynamic Voltage Scaling: Myths and Facts": Snowdon, et al, 2005.

- ➜ most academic research uses simulation for evaluation
- ➜ the naive models lead to incorrect results.

So:

- ➜ we take real measurements of a real system
- ➜ we run an appropriate benchmark suite (MiBench)

DVS: Myths and Facts

"Power Management and Dynamic Voltage Scaling: Myths and Facts": Snowdon, et al, 2005.

- ➜ most academic research uses simulation for evaluation
- ➜ the naive models lead to incorrect results.

So:

- ➜ we take real measurements of a real system
- ➜ we run an appropriate benchmark suite (MiBench)

Results:

- ➜ real measurements differ dramatically from models
- ➜ optimum processor setting is dependent on workload

DVS: Myths and facts

The naive model:

➜ $P \propto fv^2$

Limitations:

➜ ignores "static" power (core-frequency independent)

➜ ignores external components

- affects computation time
- energy consumption scales differently to the CPU

DVS: Myths and facts

Experimental setup:

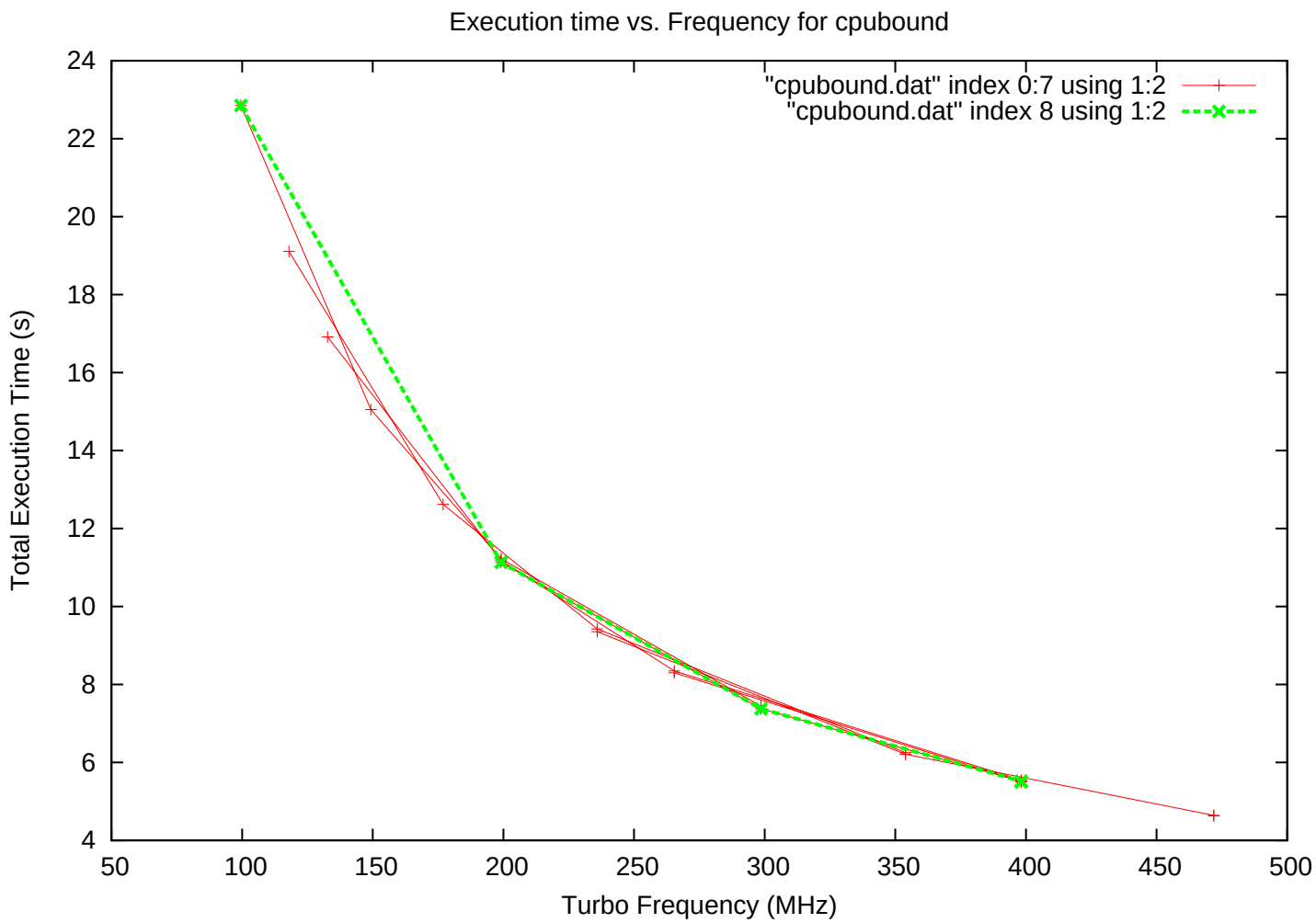
- ➜ single board computer – PLEB 2
- ➜ instrumented power supplies

Valid configurations:

V_{core} (V)	f_{cpu} (MHz)	$f_{internalbus}$ (MHz)	f_{mem} (MHz)
1.0	100	50	100
1.0	200	100	100
1.1	300	100	100
1.3	400	100	100
1.3	400	200	100

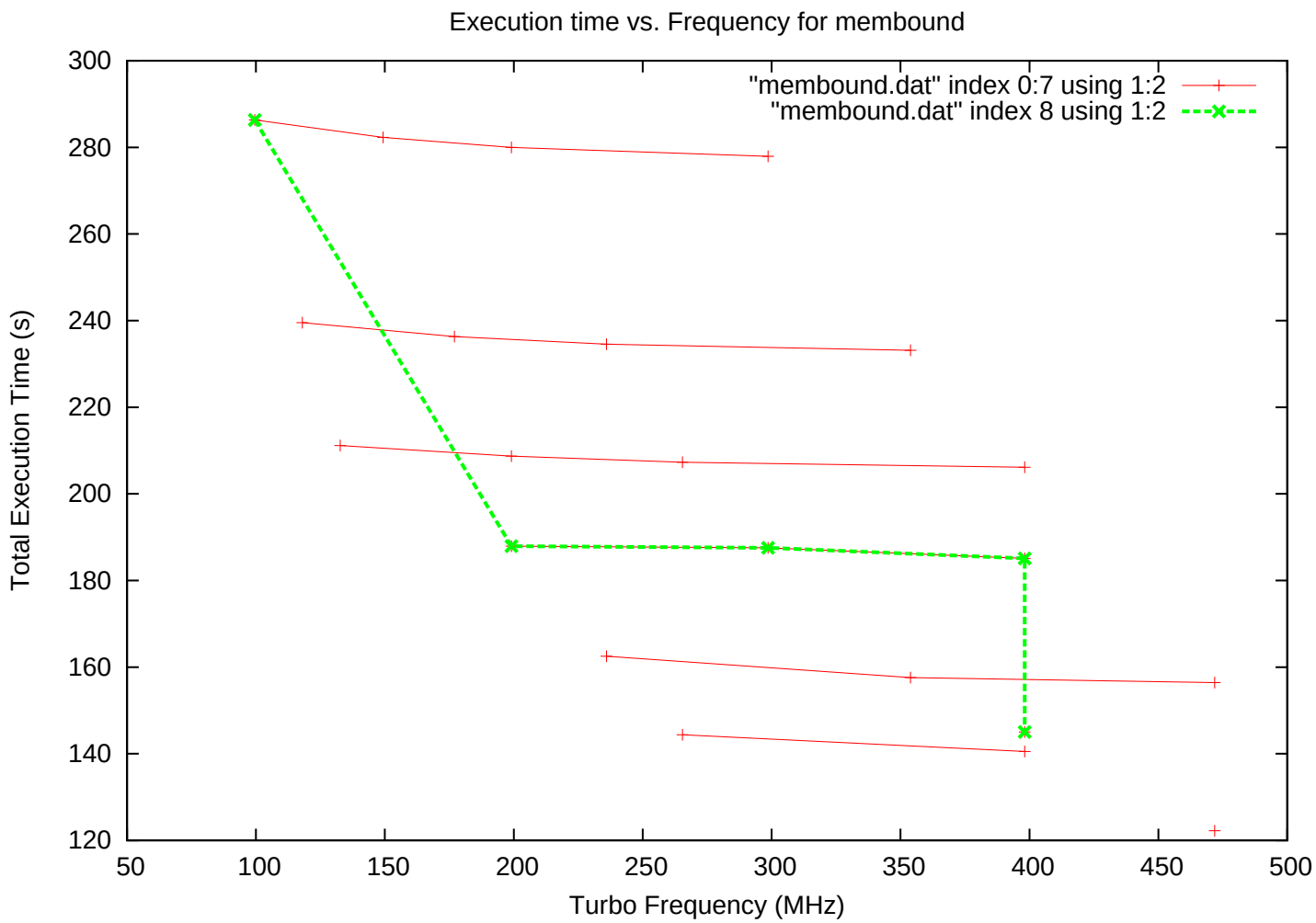
DVS: Myths and facts

Execution time for CPU bound:



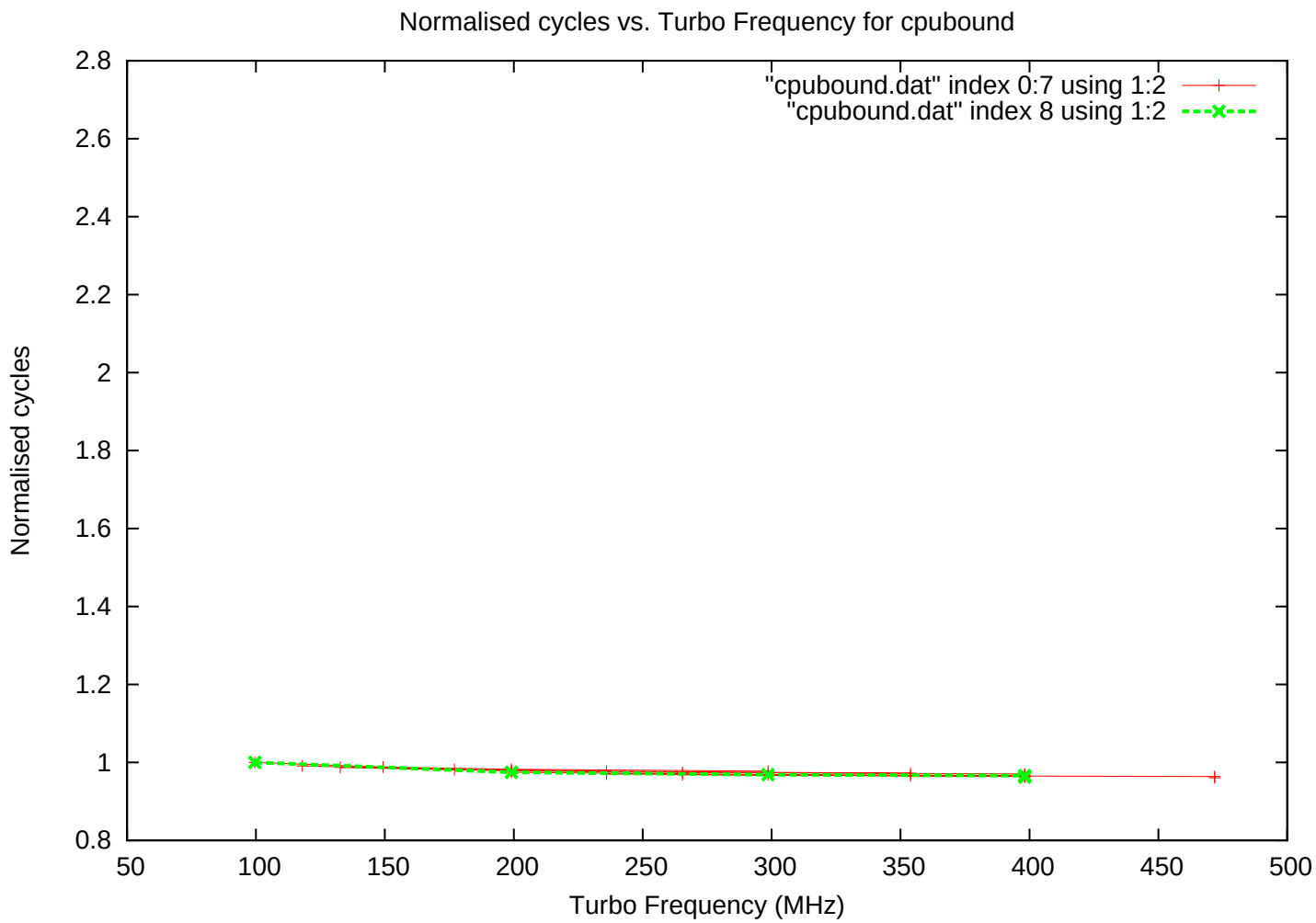
DVS: Myths and facts

Execution time for Memory bound:



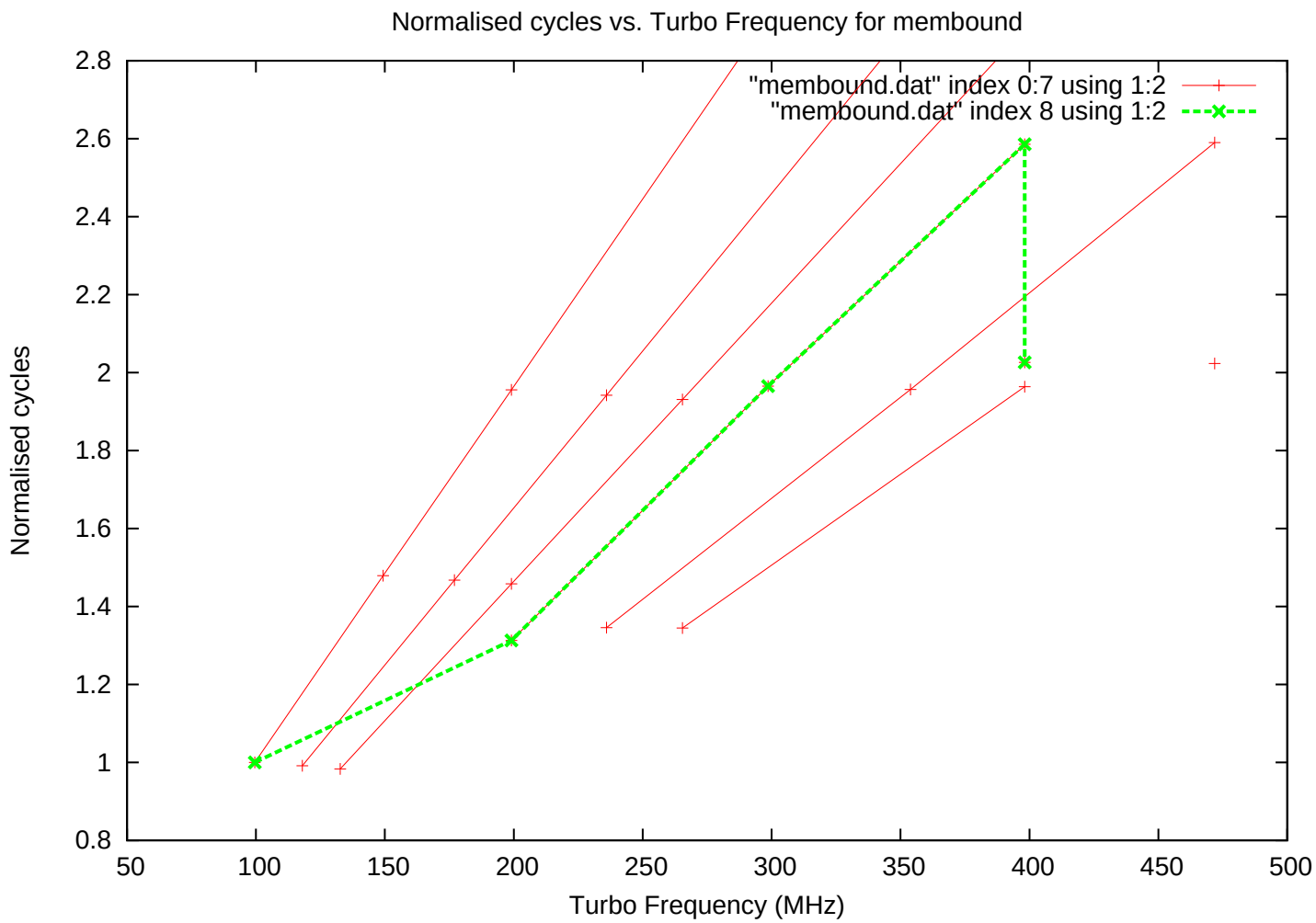
DVS: Myths and facts

Number of cycles for CPU bound:



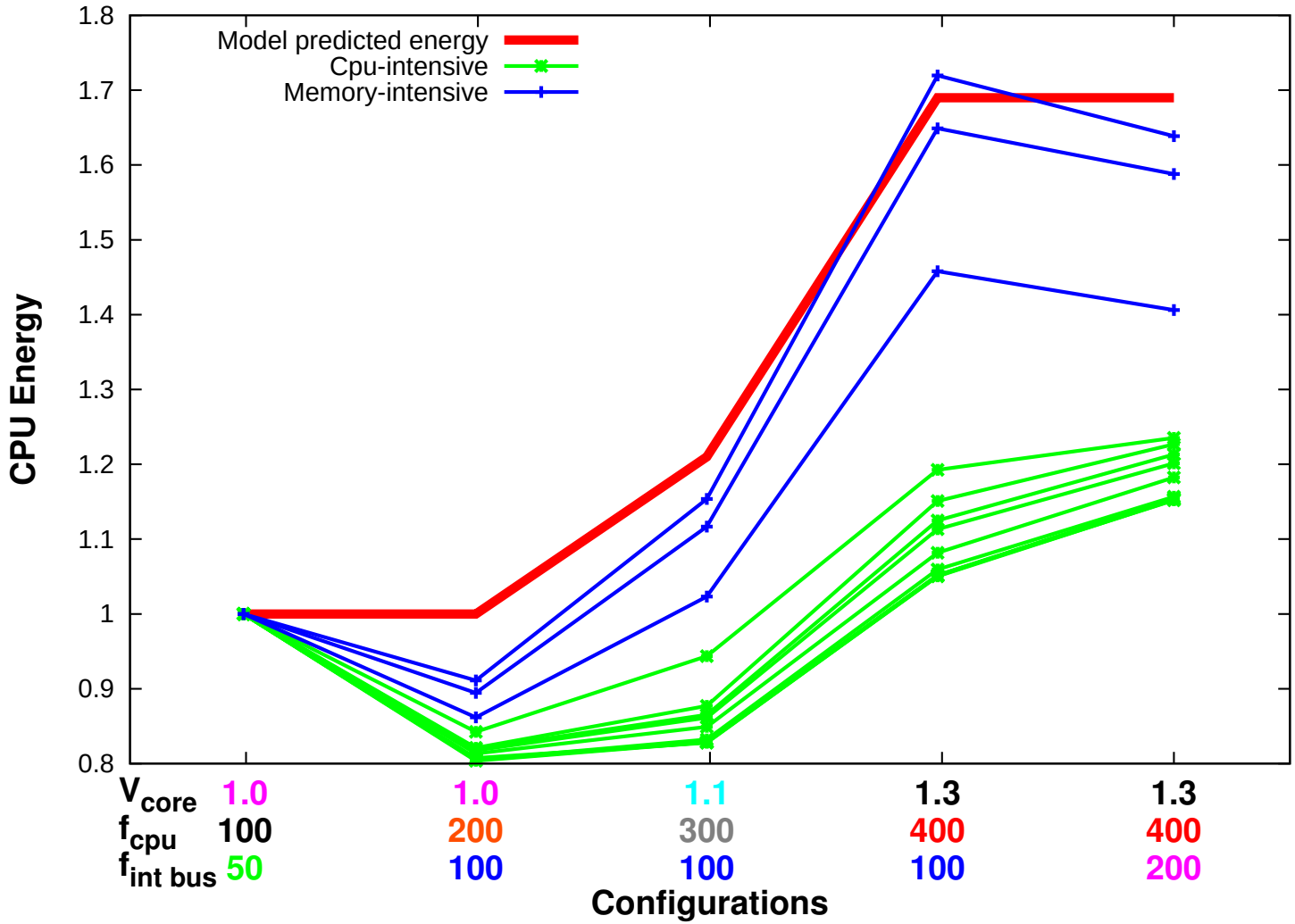
DVS: Myths and facts

Number of cycles for Memory bound:



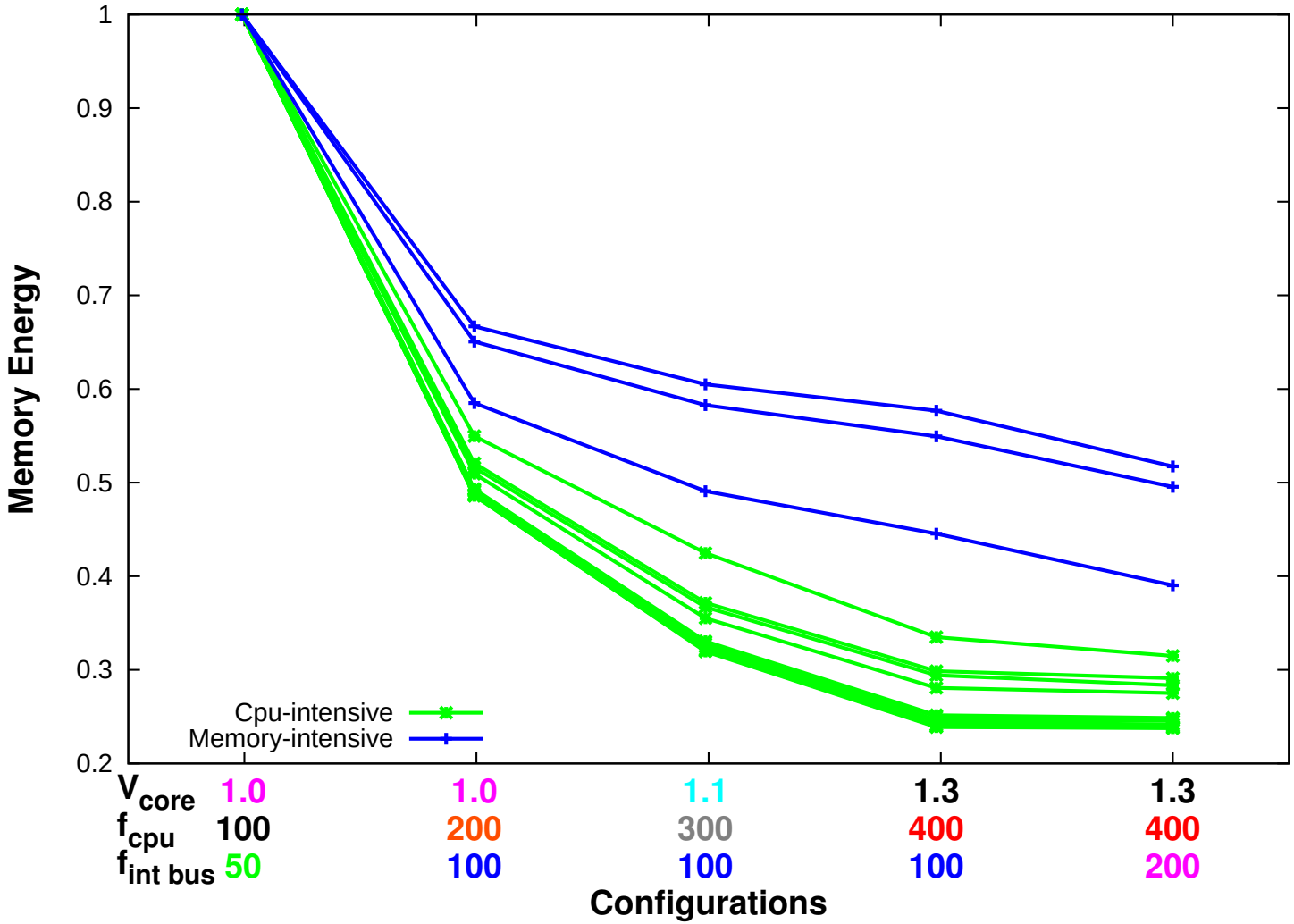
DVS: Myths and facts

CPU energy:



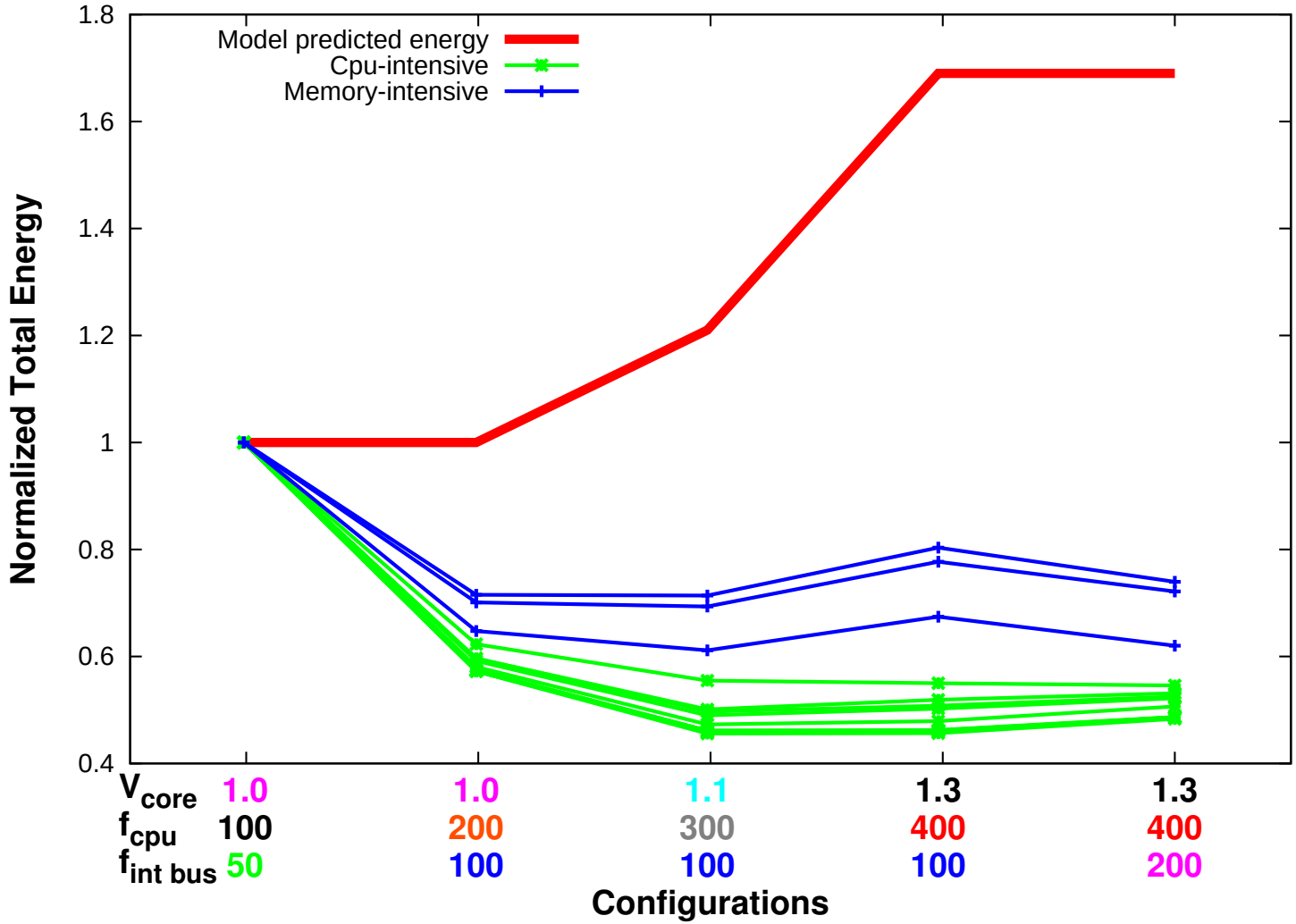
DVS: Myths and facts

Memory energy:



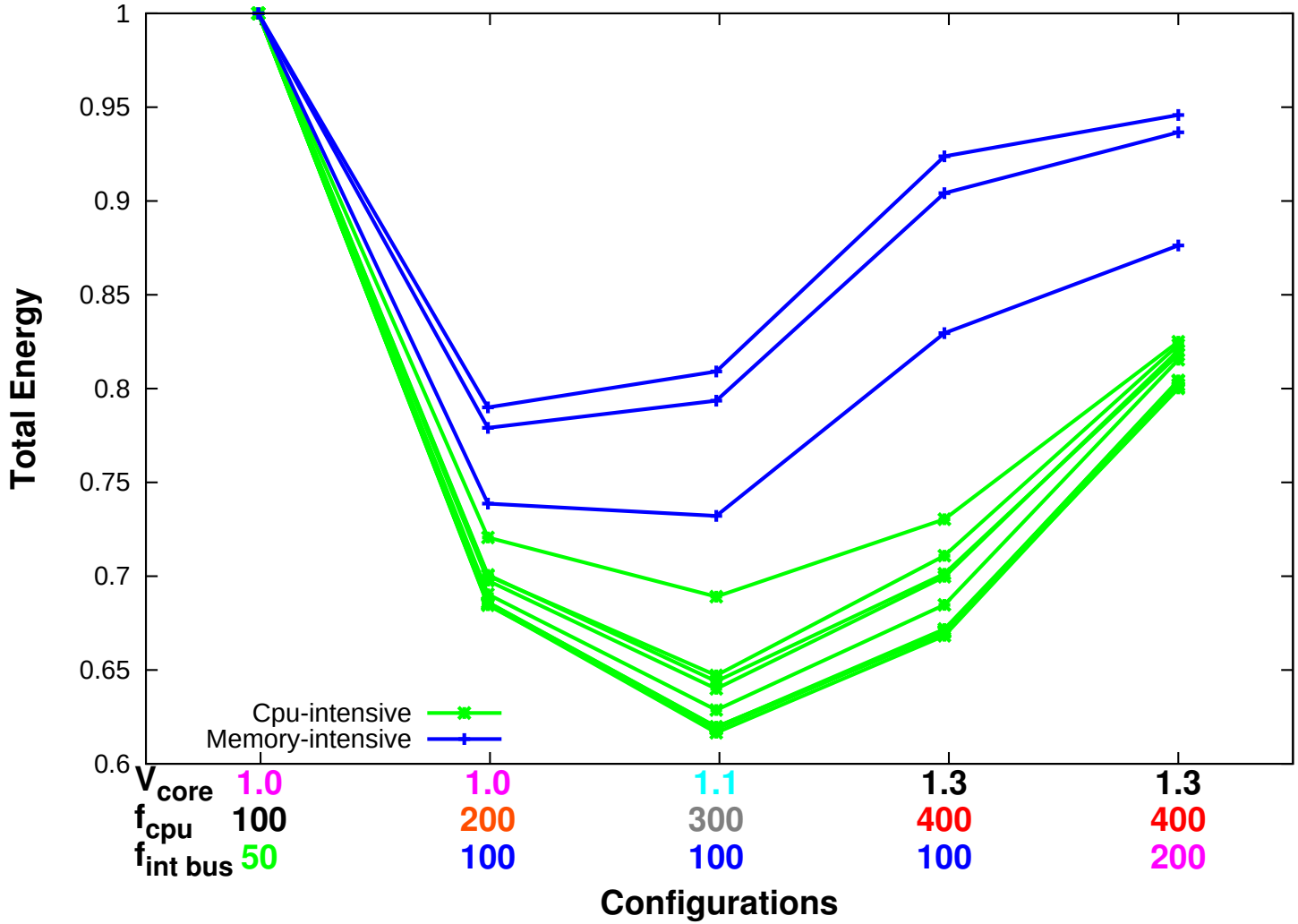
DVS: Myths and facts

Total energy:



DVS: Myths and facts

Padded with low-power mode power:



DVS: Myths and facts

Conclusions:

- the memory/cpu frequency ratio affects the number of cycles
- the most effective DVS policy depends on usage
- the lowest-energy setting depends on the workload
- increasing frequency doesn't improve membound performance
- idling in a low-power mode may be better than running at a low frequency

DVS: Myths and facts

"`Process cruise control: event-driven clock scaling for dynamic power management`": Weissel and Bellosa, 2002.

① maintain per-thread CPU performance counters

② use cache miss and instruction counters

③ use a lookup table to select a frequency

④ construct the table for ¡ 10% performance hit

Demonstrated energy savings of up to 37% with a 10% slowdown for gzip

DVS: Myths and facts

"`Process cruise control: event-driven clock scaling for dynamic power management`": Weissel and Bellosa, 2002.

① maintain per-thread CPU performance counters

② use cache miss and instruction counters

③ use a lookup table to select a frequency

④ construct the table for ¡ 10% performance hit

Demonstrated energy savings of up to 37% with a 10% slowdown for gzip

How would this affect a μK based system?

SLEEP MODES

Sleep modes:

- processors and devices have low-power modes
- these modes switch off parts of the processor
- some state or functionality will be lost
- lower-power modes tend to lose more functionality
- lower-power modes take longer to enter and exit

SLEEP MODES

Sleep modes:

- ➜ processors and devices have low-power modes
- ➜ these modes switch off parts of the processor
- ➜ some state or functionality will be lost
- ➜ lower-power modes tend to lose more functionality
- ➜ lower-power modes take longer to enter and exit

How do we choose when to sleep?

- ➜ processor(s)
- ➜ devices

SLEEP MODES

Case study: PXA255

	Power
200MHz Run/Turbo	$178mW$
200MHz Idle	$63mW$
33MHz Idle	$45mW$
Sleep	$0.175mW$

SLEEP MODES

Case study: Hard disk

	Sleep delay	Wake delay	Power
Idle	0	0	$657mW$
Standby	0.85s	1.03s	$235mW$

SLEEP MODES

Transition management:

- ➜ some energy is saved while "asleep"
- ➜ some energy is wasted while transitioning
- ➜ introduced delay reduces quality-of-service

We need to predict the future

SLEEP MODES

"`Idleness is not sloth`": Golding, et al, 1995.

- ➜ an "idleness detector" should predict the start time and duration of an idle period

Idle start:

- ➜ timeout based: (static, variable, adaptive)
- ➜ rate-based: (static, adaptive thresholds)
- ➜ periodic: (maintain a DPLL)

SLEEP MODES

"`Idleness is not sloth`": Golding, et al, 1995.

- ➜ an "idleness detector" should predict the start time and duration of an idle period

Idle start:

- ➜ timeout based: (static, variable, adaptive)
- ➜ rate-based: (static, adaptive thresholds)
- ➜ periodic: (maintain a DPLL)

Idle duration:

- ➜ static (infinite, none, fixed)
- ➜ moving average (filtered, unfiltered)
- ➜ backoff (geometric increase, arithmetic decrease)
- ➜ periodic

SLEEP MODES

OS support:

- Cooperative IO (Weissel, et al)
- Ghosts in the machine (Anand, et al)
- Power-aware page allocation (Lebeck, et al)
- Timer ticks (IBM Watch project)

Other possible power down areas:

- LCD backlight (user context sensing, selective lighting)
- wireless network
- sound card

SLEEP MODES

Conclusions:

- keep it simple - DTT with break-even time does OK.
- adaptive techniques can save power
- operating system co-operation improves
- Douglis et al present potential reductions of 60% beyond a 5s timeout

ADAPTATION

Adaptation:

- ➜ applications can sometimes adjust their computation best
  - MPEG/JPEG
  - audio bitrate
  - game FPS
  - distributed process migration
  - map viewer detail (Flinn)
- ➜ applications can scale their QoS to meet a goal
  - a given battery lifetime
  - a given CPU temperature
- ➜ requires operating system feedback

EVALUATION TECHNIQUES

Evaluation techniques:

- ➜ Simulation
- ➜ State-based accounting
- ➜ Indirect measurement
- ➜ Real measurement

EVALUATION TECHNIQUES

- Simulation
  - necessary information may not be available
  - non-trivial to build
  - usually limited to CPU/memory.
  - computing power vs. detail
  - difficult to interface
- State-based accounting
  - assumes a constant power in each state
- Indirect measurements
  - only really applies to the CPU and memory (at the moment)
  - Insufficient performance counters for energy-related events
- Real power measurements
  - may measure background activity incorrectly
  - overhead vs. resolution
  - cumbersome difficult to obtain setup

EVALUATION TECHNIQUES

Powerscope (Energy profiling):

- Flinn, 99
- designed to be similar to performance profile
- A host computer monitors a computer being profiled
- a computer-controlled multimeter samples the actual energy used
- the profiled computer records PIDs and PC for each sample
- post-processing attributes energy to procedures/processes

EVALUATION TECHNIQUES

Procedure	Elapsed Time (s)	CPU Power (W)	Memory Power (W)
synth_1to1	**21.520**	**0.676**	**0.458**
dct36	3.741	0.657	0.612
III_antialias	1.226	0.652	0.488
dct64	4.594	0.652	0.481
do_layer3	0.226	0.639	0.567
main	0.016	0.635	0.893
dct12	0.069	0.635	0.701
stream_head_shift	0.001	0.625	0.000
III_dequantize_sample	3.642	0.622	0.547
set_pointer	0.017	0.613	0.649
III_hybrid	0.523	0.608	0.724
init_layer3	0.032	0.604	0.419
audio_flush	0.007	0.598	0.427
III_get_scale_factors_1	0.164	0.575	0.703
read_frame	0.065	0.571	0.821
III_get_side_info	0.129	0.569	0.753
stream_read_frame_body	**0.011**	**0.543**	**1.160**
decode_header	0.037	0.530	0.742
play_frame	0.021	0.528	0.574
wav_write	0.009	0.523	0.475
stream_head_read	0.004	0.521	0.832
__divsi3	0.002	0.508	0.000

ENERGY ACCOUNTING

Energy accounting:

- ECOSystem (Zeng, et al, 2001)
  - *currentcy* is used as a primitive for energy allocation
  - currentcy is only valid for a certain time
  - resource containers are used for resource charging
  - model-based techniques are used to charge RCs.
  - internal policy is used to select resource containers
- LEA
  - charges using real power measurements

ENERGY ACCOUNTING

Energy accounting:

- ECOSystem (Zeng, et al, 2001)
  - *currentcy* is used as a primitive for energy allocation
  - currentcy is only valid for a certain time
  - resource containers are used for resource charging
  - model-based techniques are used to charge RCs.
  - internal policy is used to select resource containers
- LEA
  - charges using real power measurements

Energy Budgeting:

- a budget is allocated to each resource container
- resource containers in debt are not scheduled
- the budget is periodically refreshed

ENERGY ACCOUNTING

''Event-Driven Energy Accounting for Dynamic Thermal Management'': Bellosa, et al, 2003

- ➜ use CPU performance counters to estimate CPU temp
- ➜ account the energy contribution of each RC
- ➜ schedule "hot" processes interspersed with "cold" processes
- ➜ can maintain a given temperature limit