

Control for High Availability, Mission Critical Networked Visualisation Systems

Barco
Security and Monitoring Division

Marc Leeman, Ph.D.

Overview

- **Software Definition**
- Hardware Overview
- Barco SMD Case Study
 - Typical Causes for Errors
 - Design Considerations
 - Reliable, Predictable
 - Embedded Software
 - System Layout
 - Software Control
 - Fall Back: serial
- Conclusions

What is Software?

- Software \neq Java/C++
 - Algorithmic form
 - e.g. C code
 - Relational form
 - e.g. VHDL
- Software fundamentally is the unique image or representation of physical or material alignment that constitutes configuration to or functional identity of a machine, usually a computer.
- Not all fault tolerance needs to be done in one component (e.g. uC) on one die.

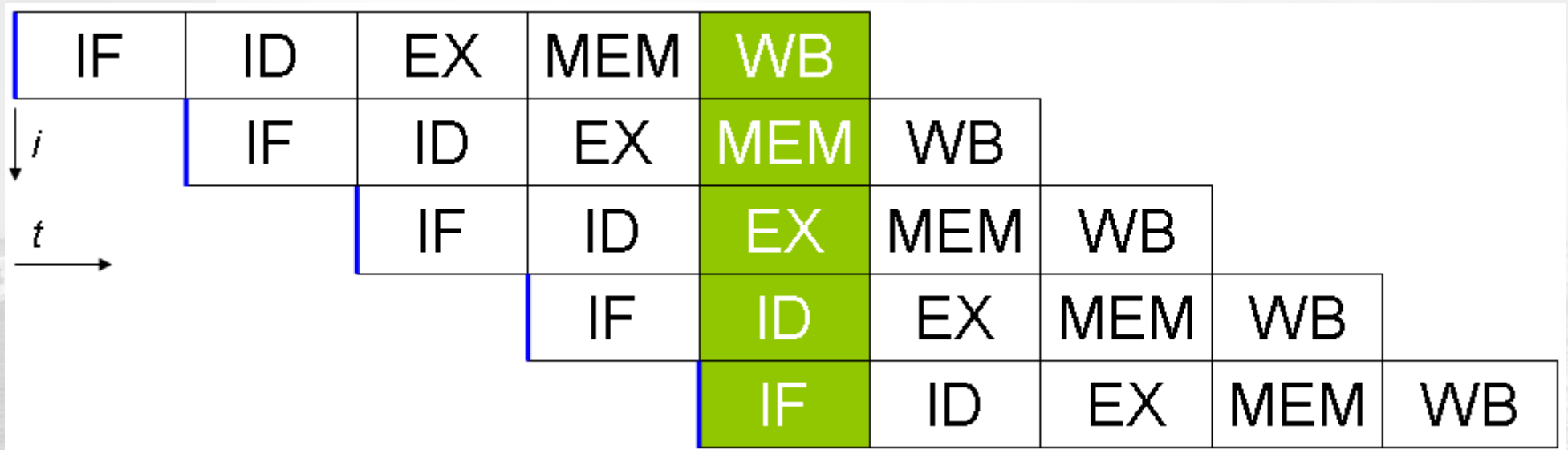
Overview

- Software Definition
- **Hardware Overview**
- Barco SMD Case Study
 - Typical Causes for Errors
 - Design Considerations
 - Reliable, Predictable
 - Embedded Software
 - System Layout
 - Software Control
 - Fall Back: serial
- Conclusions

Hardware Overview

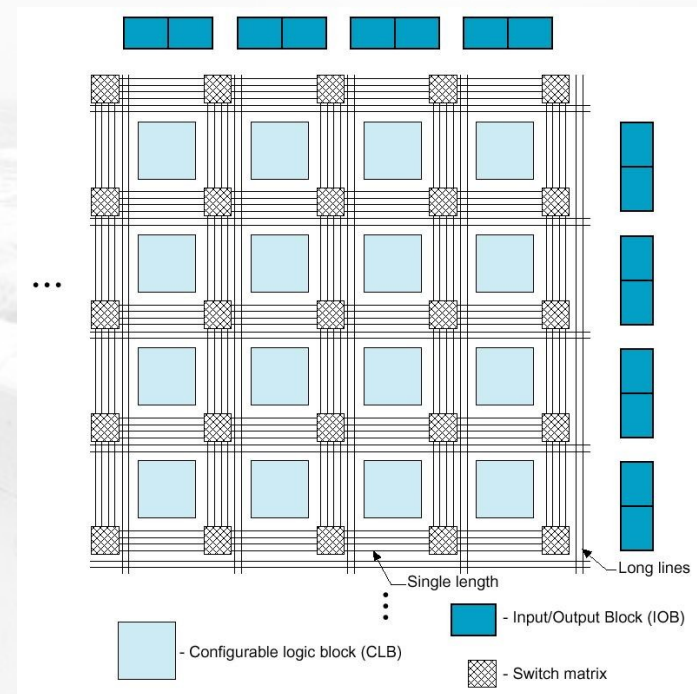
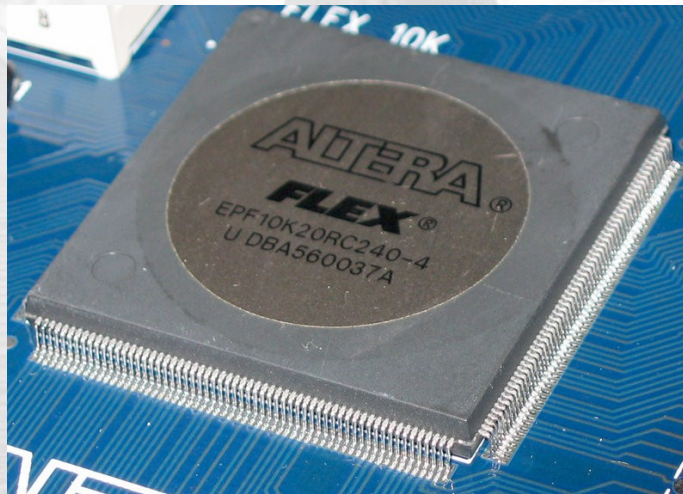
- microprocessor/microcontroller (uP/uC)
 - ia86, ia86-64
 - powerpc, arm, m68k, ...
- Digital Signal Processor
 - special class of uP
 - for digital signal processing (e.g. video and audio sampling and processing).
 - Instructions in parallel
 - Deep instruction pipeline
 - Specialised instructions (e.g. vector operations).

Instruction Pipeline



Hardware Overview (cont.)

- Field Programmable Gate Arrays
 - Parallel execution
 - Gates are programmed to mimic logic gates



Hardware Overview (cont.)

- Application Specific Integrated Silicon
 - expensive to develop
 - Highly dedicated, efficient and performant devices.
 - Performs one and only one function.
 - Off The Shelf usage (COTS)



Notes

- Distinction between these device classes is not so clear cut anymore
- uP has DSP properties
 - e.g. vector operations
- DSPs are equipped with uP cores
 - Da Vinci platform from TI
- FPGAs contain uP soft and hardcores

Cost/Design

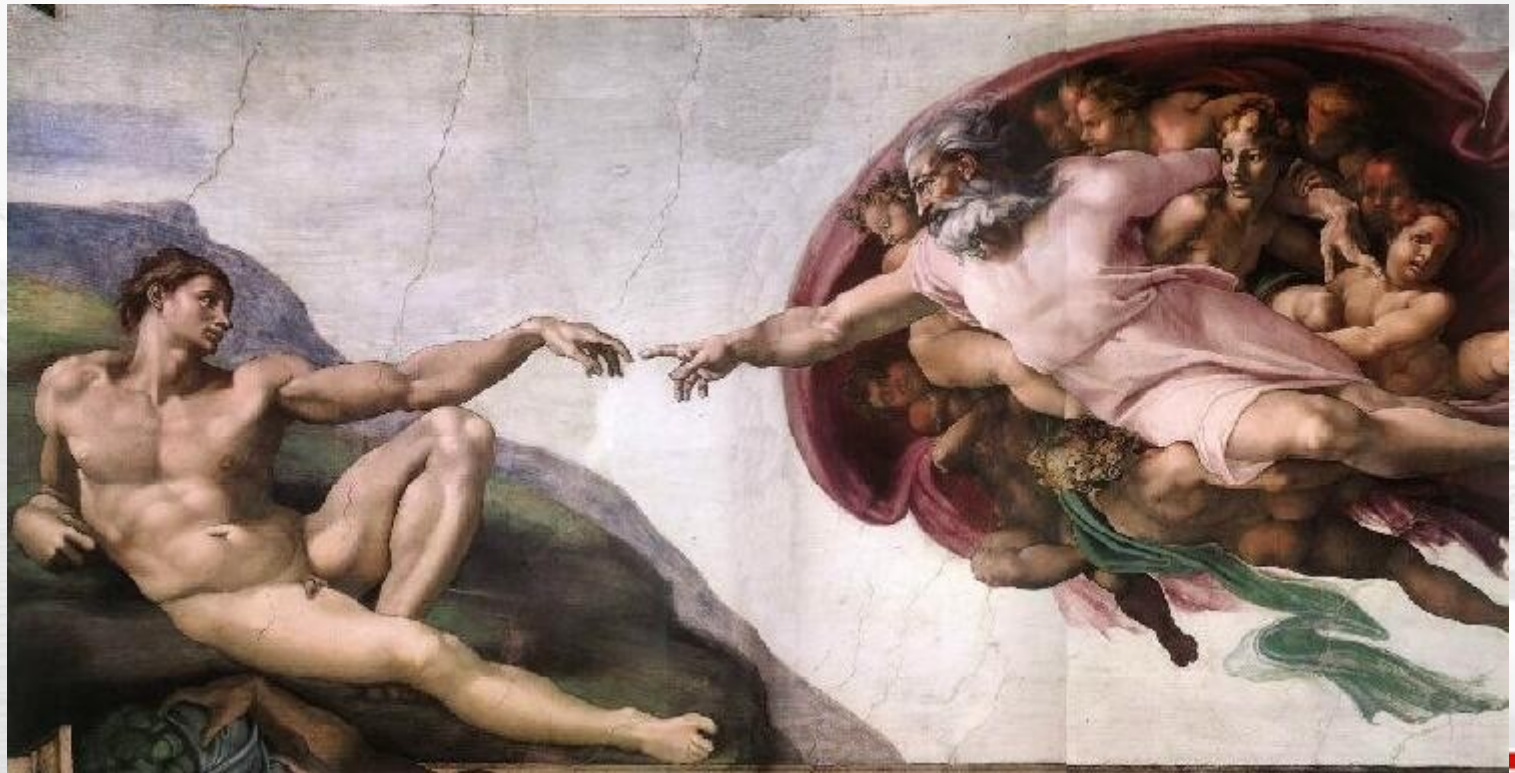
- Application Specific Integrated Circuits (ASICs) are cheap to produce but expensive to create
 - large initial investment costs
 - large volumes
- FPGAs are more expensive but lower initial development cost
 - small to middle sized volumes.

What is Software? (cont.)

- Hardware can be viewed as inanimate matter; it needs to be correct to function properly.
 - Body
- Software makes the hardware alive.
 - Flexible.
 - Can compensate for errors (cf. brain function recovery).

What is Software? (cont.)

- *Writing software is playing god (on a system level)?*



Overview

- Software Definition
- Hardware Overview
- **Barco SMD Case Study**
 - Typical Causes for Errors
 - Design Considerations
 - Reliable, Predictable
 - Embedded Software
 - System Layout
 - Software Control
 - Fall Back: serial
- Conclusions

Barco SMD case study

- Streaming Video Card
- Take in video and decode it
 - mpeg{2|4}, wavelet based, mjpeg, mxpeg, ...
 - vendor 'extensions'
 - up to 8 streams on one card
 - 1 Altera Stratix II FPGA
 - 5 DSP C64 DSPs
 - 1 PowerPC 8347 processor



Typical Causes for Errors

- Hardware failure/bugs
 - MAC hangs
- Software bugs
 - segfault, out of memory.
- Unforeseen circumstances.
- Peripheral network
 - switches, routers, network settings, ...
- STEUs (STupid End Users)
 - the most dangerous are not those that don't know the system but those that think they know the system.

Design Considerations

- Dependability
 - High Availability
 - Reliable
 - Predictable
 - Maintenance
 - Security
- Cost
- *With these in mind, select an optimal combination of devices that adequately cover the functionality*

Reliable

- Hardware
 - High uptime
 - Error correction
 - Redundancy
 - Component Level
 - e.g. networking interfaces
 - Board Level
 - Hot swap
 - in case of hardware failure/other unresolvable problems
 - System Level
 - e.g. hot stand-by

Reliable (cont.)

- Software
 - Correct **BEFORE** the faults occur
 - Inspect and correct data stream
 - Negotiate settings
 - Cope with Hardware changes
 - e.g. restore settings
 - Cope/Mask Hardware failures
 - e.g. MAC lock
 - Corrupted flash sectors
 - Catch unforeseen circumstances
 - power-failure while writing to persistent media

Predictable

- A predictable situation is always to be preferred over a unknown state.
- Uncertainty = 666.
 - High cost for recovery and reconfiguration.
 - Downtime.
- Basis for recovery.
- *It is better to resort to an old predictable situation than to a another unpredicatable one.*

Embedded Software

- Systems are stand-alone/headless
 - no keyboard
 - no screen
- Some (older) systems do not provide a serial interface
- Access:
 - shell
 - BDI/JTAG probe

Small is Beautiful

	speed increase	RAM reduction	Power reduction	Cost reduction
More speed			- CPU can run slower or stay longer in power saving mode	- Slower, cheaper CPU
Less RAM	- Faster Allocations - Less swapping - Sometimes less cache flushing		- fewer/smaller RAM chips: less dynamic and standby power - CPU with less cache: less power	- Fewer/cheaper RAM chips - CPU with less cache: cheaper
Less space	- Faster application loading from storage and in RAM - Sometimes simpler, faster code	- less ram usage	- Fewer/smaller storage chips: less power	- Fewer/cheaper storage
Less power				- Cheaper batteries
Less complexity				- Less design errors

Overview

- Software Definition
- Hardware Overview
- Barco SMD Case Study
 - Typical Causes for Errors
 - Design Considerations
 - Reliable, Predictable
 - Embedded Software
 - **System Layout**
 - Software Control
 - Fall Back: serial
- Conclusions

System Layout

- Bootloader: Das U-Boot
 - flexible, reliable, redundancy, ...
- Kernel: Linux
 - reliable, modifiable, flexible, ...
 - open: no hidden complexity
 - closed 3rd party Oses: difficult to obtain a measure of reliability
 - Code quality of 'closed source' Oses is poor (cf. Windows CE and XP embedded).
 - No peer review

System Layout (cont.)

- System level:
 - uclibc + busybox
 - uclibc: Smaller footprint wrt to glibc applications
 - busybox: shared (startup) code
 - Custom application firmware, CGI, mDNS and dropbear for shell access.

Das U-Boot

- Hardware initialisation and setup
 - set CPU (MPC8245 and MPC8347)
Registers
 - some registers can only be set @startup
 - GPIO/Interrupt
 - MMU activation
- Loads Linux kernel from flash in memory and extracts it
 - adjust PC
- serial, hardware inspection, scripting, ...

```
GtkTerm
File Configuration Control signals View Help

U-Boot 1.1.4 (Aug 28 2006 - 06:04:21) MPC83XX

Clock configuration:
  Coherent System Bus: 264 MHz
  Core:                 396 MHz

  Local Bus:           33 MHz
CPU:   MPC83xx, Rev: 1.1 at 396 MHz
Board: Barco BARCO834XG1 Platform
I2C:   ready
DRAM:  Initializing
SDRAM on Local Bus is NOT available!
  DDR RAM: 256 MB
FLASH: Chip: S29GL128N
16 MB
PCI1: 32-bit on PMC1
PCI2: 32-bit on PMC2, PMC3
PCI:   Bus Dev VenId DevId Class Int
      00 1e 1172 0004 ff00 00
In:    serial
Out:   serial
Err:   serial
Net:   TSEC0, TSEC1

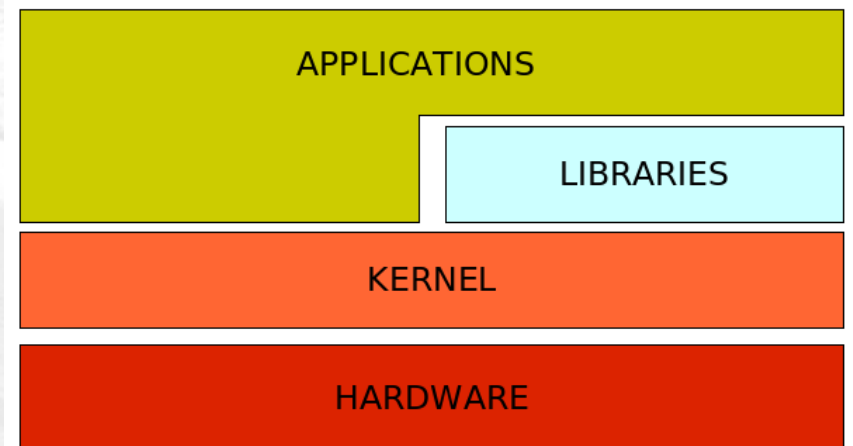
Type run net_nfs to get kernel via tftpboot
and mount root filesystem over NFS
use bootp_nfs to get an IP and boot over NFS

Hit any key to stop autoboot: 0
=> █
```



Linux Kernel

- Linux \neq OS \neq distribution!
 - GNU/Linux, GNU+Linux, ...
- Single point of access to hardware
 - All application must pass through the kernel to access components
- User separation
- (Dependability)



The Linux Kernel

- Basic hardware detection and initialisation
 - In the kernel image is needed:
 - minimal configuration to allow recognition of root FS
 - hardware with for root partition
 - SCSI, IDE, LIBATA, FLASH, MAC/PHY, ...
 - File system
 - ext2, ext3, jfs, xfs, nfs, reiserfs, ...
 - What is not needed:
 - Firewall (iptables), keyboard, mouse, ... (peripherals)
 - Can be postponed to the final stages of booting

The Linux Kernel (cont.)

- Good Practice
 - only include the needed functionality
 - only include the needed complexity
- Modules offer an extra level of abstraction: it is/is not in the kernel.

```
Terminal
File Edit View Terminal Tabs Help
Created: 2006-11-20 15:20:42 UTC
Image Type: PowerPC Linux Kernel Image (gzip compressed)
Data Size: 938805 Bytes = 916.8 kB
Load Address: 00000000
Entry Point: 00000000
Verifying Checksum ... OK
Uncompressing Kernel Image ... OK
[ 23.856161] serial8250.0: ttyS0 at MMIO 0xe0004500 (irq = 9) is a 16550A
[ 23.862994] serial8250.0: ttyS1 at MMIO 0xe0004600 (irq = 10) is a 16550A
[ 23.881287] Gianfar MII Bus: probed
[ 23.885035] eth0: Gianfar Ethernet Controller Version 1.2, 00:04:a5:04:05:93
[ 23.892228] eth0: Running with NAPI enabled
[ 23.896410] eth0: 256/256 RX/TX BD ring size
[ 23.900911] eth1: Gianfar Ethernet Controller Version 1.2, 00:04:a5:04:05:93
[ 23.908099] eth1: Running with NAPI enabled
[ 23.912281] eth1: 256/256 RX/TX BD ring size
[ 23.916922] TCP bic registered
[ 23.919998] NET: Registered protocol family 1
[ 23.924365] NET: Registered protocol family 17
[ 25.438147] Sending BOOTP requests .<6>PHY: 0:01 - Link is Up - 100/Full
[ 28.298107] .. OK
[ 33.442067] IP-Config: Got BOOTP answer from 10.0.0.6, my address is 10.2.4.10
[ 33.459223] IP-Config: Complete:
[ 33.462300] device=eth1, addr=10.2.4.10, mask=255.0.0.0, gw=10.0.0.1,
[ 33.469204] host=10.2.4.10, domain=, nis-domain=(none),
[ 33.474868] bootserver=10.0.0.6, rootserver=10.0.0.6, rootpath=/home/firmware
[ 33.485559] Looking up port of RPC 100003/2 on 10.0.0.6
[ 33.494800] Looking up port of RPC 100005/1 on 10.0.0.6
[ 33.621512] VFS: Mounted root (nfs filesystem).
[ 33.626199] Freeing unused kernel memory: 104k init
[ 34.926278] Barco BCD Streaming Platforms (BARCO834XG1) flash device: 1000000 at 0
[ 34.935535] Barco BCD Streaming Platforms (BARCO834XG1) Flash Layout: Found 1 x16k
[ 34.945929] Amd/Fujitsu Extended Query Table at 0x0040
[ 34.951172] Barco BCD Streaming Platforms (BARCO834XG1) Flash Layout: CFI does no.
[ 34.962394] number of CFI chips: 1
[ 34.965791] cfi_cmdset_0002: Disabling erase-suspend-program due to code brokenne.
[ 34.982283] cmdlinepart partition parsing not available
[ 34.996278] RedBoot partition parsing not available
[ 35.001247] Using barco834xgl partition definition
[ 35.006037] Creating 8 MTD partitions on "Barco BCD Streaming Platforms (BARCO834:
[ 35.015176] 0x00000000-0x00040000 : "Das U-Boot"
[ 35.019966] 0x00040000-0x00060000 : "Configuration Space 0"
[ 35.025679] 0x00060000-0x00080000 : "Configuration Space 1"
[ 35.031390] 0x00080000-0x001c0000 : "Factory Kernel"
[ 35.036497] 0x001c0000-0x00700000 : "Factory Filesystem"
[ 35.041953] 0x00700000-0x00840000 : "Upgrade Kernel"
[ 35.047064] 0x00840000-0x00d80000 : "Upgrade Filesystem"
[ 35.052529] 0x00d80000-0x01000000 : "JFFS2 Flash Filesystem"
[ 35.231515] JFFS2 version 2.2. (NAND) (C) 2001-2006 Red Hat, Inc.
[ 35.281026] Generic RTC Driver v1.07

Barco Control Rooms
barco login:
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.1 | VT102 | Offline
```

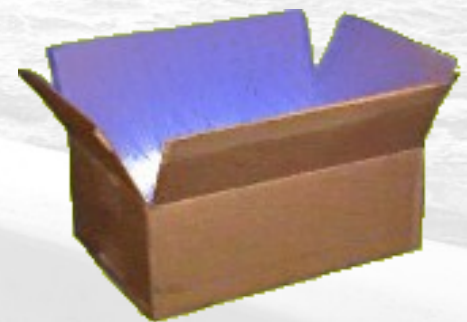
uClibc

- glibc (GNU C library):
<http://www.gnu.org/software/libc/>
Found on most computer type GNU/Linux machines
Size on `arm`: approx 1.7 MB
- uClibc: <http://www.uclibc.org/>
Found in more and more embedded Linux systems!
Size on `arm`: approx 400 KB (you save 1.2 MB!)

<i>C program</i>	<i>Compiled with shared libraries</i>		<i>Compiled statically</i>	
	<i>glibc</i>	<i>uClibc</i>	<i>glibc</i>	<i>uClibc</i>
Plain "hello world"	4.6 K	4.4 K	475 K	25 K
Busybox	245 K	231 K	843 K	311 K

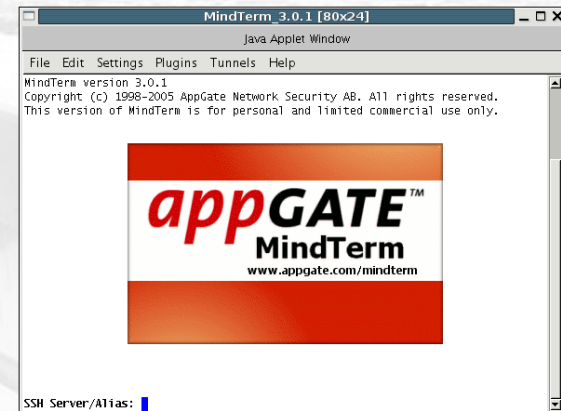
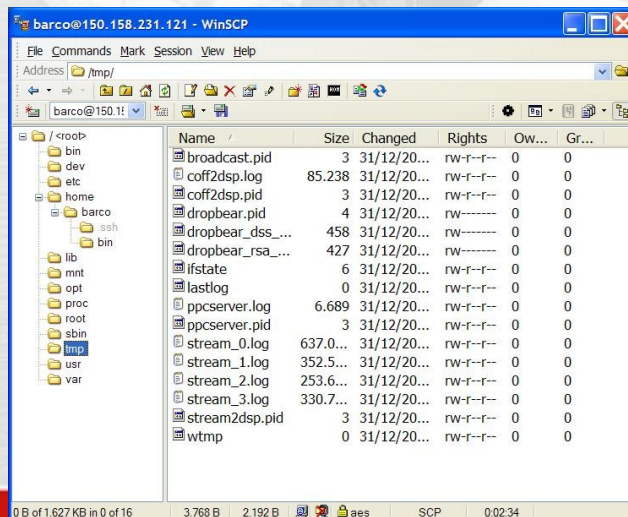
busybox

- combine unix utilities in one single small executable
- share startup code
- configurable commands
- configurable functionality of the commands
- applets easy to add



System Level

- CGI
 - Provide access levels: unauthorised access can result in parameter corruption
- Dropbear
 - Encrypted and Secure access



Overview

- Software Definition
- Hardware Overview
- Barco SMD Case Study
 - Typical Causes for Errors
 - Design Considerations
 - Reliable, Predictable
 - Embedded Software
 - System Layout
 - Software Control
 - Fall Back: serial
- Conclusions

What Now?

- We've minimised the possible fault footprint by carefully choosing our software modules
- Finished?
- **NO**
- We need to compensate for hardware/component failure
 - Software Control

Software Control

- Though hardware provisions are taken for failures, software fine-grained software control is required at all levels to tackle failures
 - Board Level
 - CPU configuration
 - Operating System
 - UserSpace applications
 - System Level

Board Level

- CPU configuration
 - RTC/Watchdog
 - Hardware provision in silicon to detect CPU lockups
 - reset CPU
- Network
 - bonding module (bonding.ko)
 - 2 physical interfaces provide one logical interface to the applications
 - one fails, the second interface takes over

Board Level (cont.)

- bonding.ko is the kernel interface
- ifenslave binds physical interfaces to the bonding interface
- a monitoring (e.g. 10 ms) interval is provided.

Board Level (cont.)

- Configuration/Save
 - CRC32 checks validity/corruption of data (e.g. jffs2.ko)
- Flash
 - writing to flash is slow
 - erase flash: set all words to 0xffffffff
 - toggle relevant bits back to 0
 - done on block size (64/128 kB).
 - **Writing one bit in flash can cause the erase of 128 kB and re-writing $(128 * 2^{1024} - 1)$ to 0**

Board Level (cont.)

- Redundant flash partition
 - CRC checked
 - bit valid toggle
 - write to flash
 - toggle current flash data as not valid
 - burn data to flash with valid bit
 - restore from flash
 - check for valid bit
 - check CRC
 - if CRC is not valid, check backup
 - if backup CRC not valid; restore with sensible defaults

Board Level (cont.)

- Current fails systems are within one device
- FPGA plays master over processor
- watchdog daemon passes alive beat to FPGA
 - if watchdog is not asserted, FPGA brings processor in reset.

Board Level (cont.)

- DSP: Processor without MMU
 - no protection for different memory segments
 - rogue pointers can corrupt data/program memory
 - a reload of the program/OS is required
- Processor watches DSP (#decoded frames)
 - re-loads when #frames does no longer increment/changes

System Level

- Several boards in one case
 - 'board manager'
- Several cases in one system
 - boards backup their data to central point
 - when one board fails and is replaced (hot plug), board manager provides the needed details to retrieve settings from failed boards from central point

System Level (cont.)

- At startup of a system
 - Code is loaded from the board itself
 - modular
 - changes isolated per board
 - Code is loaded via a backplane
 - changes are system/node wide
 - 1 upgrade upgrades the system/node

Low-Level Fall-Back

- Serial interface
 - simple protocol
 - 2 lines/pins
- Spawn a shell on /dev/ttyS0
 - serial shell access
- Debugging messages are sent to serial
- Access to bootloader
 - inspect hardware
 - replace flash images

Overview

- Software Definition
- Hardware Overview
- Barco SMD Case Study
 - Typical Causes for Errors
 - Design Considerations
 - Reliable, Predictable
 - Embedded Software
 - System Layout
 - Software Control
 - Fall Back: serial
- **Conclusions**

Conclusions

- Reliability needs to be taken into account during the entire product design.
- HW/SW reliability are complementary:
 - SW masks/corrects HW faults
 - HW masks/corrects SW faults
- HW/SW Co-design

Conclusions (cont.)

- You can only write reliable software as long as you keep a clear view on the underlying hardware.
 - e.g.:
 - What are the effects of code on an instruction level?
 - What are the effects of code on memory usage?
 - How does memory usage influence runtime behaviour?