Xenomai and Realtime Image Processing Control

Jorge Ramirez-Ortiz
Xerox Ltd
AL7 1BU, UK

Xenomai Users Meeting [XUM]
Realtime Linux Workshop 2009, Dresden [Germany]
A multifunction system (fax, copy, print ..)

IP ASICs (each):
- ~1.5M gates
- ~1.6M bits
- ~280 pins
- Several hundreds regs

Scanner input
Test patterns
PCI Bridge
ip filters
N DMA channels
Y Compressors
Z Decompressors
W UARTs
Rotation module
Scanner loopback
Output formatter to laser...

Images
DDR
Image Processing Platform OS

- U-BOOT loader.
- GPOS GNU/Linux.
- Xenomai.
- Standard drivers and protocols.
- Proprietary drivers and protocols
- Abstraction Layers.
Several millions of proprietary lines of user code.

Hard real-time requirements on the ASICs control.
Initial IP Prototype Phase

- ASICs/FPGAs on a PCI card.
- Tcl/Tk prototype User Interface controlling ASICs.
- IP ASIC Controls initial port from legacy code via Xenomai skins.
- IP ASIC Controls extended to include scanner and marking engine protocol components [userspace]

- Platform: standard x86 PC on GNU/Linux.
- Level of effort: ~2 man/month [depends on expertise]
IP ASICS control and Benefits

- Userland Multitask Xenomai process
  - Controls PCI bridge and all devices [DMA, Compressors, etc].
  - Sets up DDR controller [if fitted]
  - Controls scanner interface and filters
  - Implements interrupt handling [>30 interrupt sources].
  - Implements video file system.
  - Implements UART/Serial drivers.
  - Implements IPC interface

- IP ASICS Control written in C++/STL.
- IP ASICS Control protected by the MMU.
- IP ASICS Control not under GPL [no kernel code]
- Debugged with GDB [including interrupt handlers].
- Level of effort: ~8 man/month [depends on expertise]
Conclusion

- We have used Xenomai as a hard realtime provider for a commercial grade GNU/Linux application.

- Writing on **C++/STL user land** accelerated our development phase
  - Simplified initial port from proprietary RTOS
  - No GPL’d code [intellectual property protected].
  - Simplified debugging via GDB including ISRs.
  - No need for ‘expensive’ kernel expertise.
  - Code reuse, modular programming, etc