The Washington University
Smart Port Card

John DeHart
Washington University
jdd@arl.wustl.edu
http://www.arl.wustl.edu/~jdd
SPC Personnel

- Dave Richard - Overall Hardware Design
- Dave Taylor - System FPGA
- Mike Richards - Board Layout
- Ed Spitznagel - Embedded NetBSD kernel
- John DeHart - Integration and Testing

- Anshul Kantawala - Embedded NetBSD kernel
- Zubin Dittia - APIC Driver
- Berkley Shands - APIC Driver

- Will Eatherton - Original SPC design
- Toshiya Aramaki - Original SPC design
Outline

• Hardware:
  – SPC as a PC
  – SPC Hardware Components

• Software
  – Kernel Building and Loading
  – Network Configuration

• Preliminary Performance Numbers
Typical Pentium PC

Cache -> CPU -> North-Bridge

North-Bridge -> DRAM

SouthBridge (PIIX3) (PIC, PIT, ...) -> PCI Bus

PCI Bus -> ISA Bus

ISA Bus -> ISA Devices

North-Bridge <-> SouthBridge

Addr/Data -> Ctrl

Ctrl -> Ctrl

Addr/Data/Ctrl <- Addr/Data

INIT -> NMI -> Intr

RTC, Uarts, Kbd/Ms, Floppy, Parallel

BIOS

Super-IO
What SPC Needs

Cache

CPU

North-Bridge

DRAM

SouthBridge (PIC, PIT, …)

APIC

RTC

BIOS

PCI Bus

Addr/Data

Ctrl

Ctrl

Addr/Data/Ctrl

NMI

Intr

INIT
SPC Architecture
SPC Photo Tour

- Switch Interface
- DRAM
- Link Interface
- APIC
- CPU Module
- PCI Bus
- System FPGA
- Serial Ports
SPC Components

• APIC
  • PCI Bus Master

• Pentium Embedded Module
  – 166 MHz MMX Pentium Processor
    • L1 Cache: 16KB Data, 16KB Code
  – L2 cache: 512 KB
  – NorthBridge - 33 MHz, 32 bit PCI Bus
    • PCI Bus Master

• System FPGA
  • PCI Bus Slave
  – Xilinx XC4020XL-1 FPGA
  – 20K Equivalent Gates
  – ~ 75% used
SPC Components (continued)

• Memory
  – EDO DRAM
  – 64MB (Max for current design)
  – SO DIMM

• Switch Interface - 1 Gb Utopia

• Link Interface - 1 Gb Utopia

• UART
  – Two Serial Ports
    • NetBSD system console
    • TTY port
System FPGA

- Coded in VHDL
- PCI slave device
- Replaces some of the PIIX3 (south bridge)
- Replaces some of the BIOS
- Replaces some of the Super IO Chip
System FPGA: PIIX3 Functionality

• Programmable Interrupt Controller (PIC)
  – Four Interrupts supported and statically assigned:
    • PIT (IRQ 0)
    • APIC (IRQ 5)
    • COM1 (IRQ 4)
    • COM2 (IRQ 3)
  – Static fully-nested interrupt priority structure.
  – Specific End of Interrupt is the only EOI mode supported

• Programmable Interval Timer (PIT)
  – generates a clock interrupt for NetBSD every ~10ms
System FPGA: BIOS Functionality

• Interrupt functionality replaced by static values

• Simple 16 word by 32-bit “ROM”
  – implements loop waiting for location 0xFFE00 to change value
  – then jumps to boot loader code

• Does NOT perform configuration of Northbridge
  – This will be done by the boot loader

• Does NOT perform PCI configuration of APIC
  – This will be done by the APIC Driver
System FPGA: Super IO Chip Functionality

- UART Interface
  - Two Serial lines supported
  - Fixed IRQs

- Real Time Clock
  - only the register accesses of the RTC are supported
  - no interrupts supported
  - i.e. supported only so NetBSD didn’t need to change
  - i.e. no alarms will be generated
Software Overview

- Building a Kernel
  - config
  - compile
- Getting Kernel onto device that is to execute it
  - download via APIC
- Initiating Kernel execution
  - boot loader
- Configuring Devices
  - APIC
Steps to an Executing Kernel on an SPC

• Configure Kernel
  – Start with a typical NetBSD kernel config file
  – add memory disk (filesystem will reside in kernel!)
  – serial console port
  – remove “extra” devices (ethernet, mouse, …)

• Filesystem
  – build filesystem
    • vnconfig(8) - configure vnode pseudo disk as a regular file
    • disklabel(8) - label it
    • newfs(8) - construct a new file system
  – mknod(8) - make device files
  – populate regular files from an existing system
… Steps to an Executing Kernel on an SPC

• Compile Kernel
• Make a **symbol only** version of kernel for /netbsd
  – ps, netstat, … use symbols from /netbsd
  – real kernel with file system CANNOT reside in itself!
• Add File System to Kernel
  – mdsetimage(8)
• Create an image version of Kernel
  – sets up bss section and symbol table
• Kernel bss file is ready to download!
… Steps to an Executing Kernel on an SPC

- Reset SPC
- Download Boot Loader
  - via APIC control cells
  - loaded at physical memory location: 0x0
- Flip bit at location 0xFFE00 to start boot loader
  - configure NorthBridge
    - cache, memory size, memory type, timing, etc
  - wait for kernel download to complete
    - looks for location 0x3FC to change from 0 to 1
  - set up parameters on stack and jump to kernel
    - kernel located at 0x100000
- Download kernel bss file
- Flip bit at location 0x3FC - kernel starts booting...
APIC Driver at SPC Boot Time

- PCI Configuration Space:
  - Base memory address:
    • APIC Physical memory in SPC starts at 0xF0000000
  - Memory mapping options:
    • Memory Enable
    • Master Enable
    • Write Invalidate Enable
  - IRQ
    • Driver normally would read this from PCI config space
    • APIC statically assigned IRQ 5 by System FPGA
    • For consistency, driver writes 5 to Interrupt Line Register
### PCI Configuration Space

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Vendor ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Command</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Code</th>
<th>Revision ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIST</td>
<td>Header Type</td>
</tr>
</tbody>
</table>

**Base Address Registers**

<table>
<thead>
<tr>
<th>Cardbus CIS Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem ID</td>
</tr>
<tr>
<td>Subsystem Vendor ID</td>
</tr>
<tr>
<td>Expansion ROM Base Address</td>
</tr>
</tbody>
</table>

| Reserved |
| Reserved |

<table>
<thead>
<tr>
<th>Max_Lat</th>
<th>Min_Gnt</th>
<th>Interrupt Pin</th>
<th>Interrupt Line</th>
</tr>
</thead>
</table>

**Configuration Space Header**
Defining a Route on APIC Network Interface

• Configure the APIC
  > ifconfig apic0 inet 192.168.10.1 netmask 0xffffffff0
    - network interface: apic0
    - our IP address: 192.168.10.1

• Configure a VCI
  > atm_ifconfig apic0 0xc5 open -aal5 -llc -besteffort
     -maxqueuebufs 100 -inport 0 -outports 1
    - network interface: apic0
    - vci: 0xc5
    - besteffort (vs. paced or lowdelay)
    - Tx buffer queue limit for this VCI: ‘maxqueuebufs 100’

• Configure the route
  > route add -iface 192.168.10.5 -link apic0:0.0.0.c5
    - destination: 192.168.10.5
    - network interface: apic0
    - vci: 0xc5
> atm_ifconfig
Usage: atm_ifconfig <interface> <vpivci> [open|close|modify [options...]]
Options are:
   -aal0 | -aal5
   -llc | -nollc <protocol>: use LLC/SNAP?
       protocol is inet | ratm | value of ethernet type
   -loopback
   -inport <input_port_number>
   -outports <bitvector_of_output_ports>
       bitvector: 1 => port 0, 2 => port 1, 3 => both
   -lowdelay | -paced <rate> | -besteffort
   -maxqueuebufs <max_descriptors_in_tx_queue>
       <rate> is in Kbits/sec
APIC Configuration

Control Processor

Switch Fabric

IPP
OPP
SPC
TI

Sys. FPGA
32-64 MB
North Bridge
Pentium
Cache

January 10,11  2000

Kits Workshop
Defining APIC Ports for Routing

atm_ifconfig apic0 0xca open -aal5 -llc -besteffort -inport $INPORT -outports $OUTPORTs

<table>
<thead>
<tr>
<th>INPORT</th>
<th>OUTPORTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Using the SPC in our Active Network Demo

Video Display 1

Video Source

Cross Traffic Source 1

SPC1

Switch

Fiber

Video Display 2

SPC2

Video Display 3

Cross Traffic Source 2

Cross Traffic Sink
SPC1: APIC Configuration

#!/bin/sh
ifconfig apic0 inet 192.168.10.1 netmask 0xffffffff0

#loopback
route add 192.168.10.1 127.0.0.1

# to SPC2
atm_ifconfig apic0 0x0000ca open -aal5 -llc -besteffort
   -maxqueuebufs 100 -inport 0 -outports 1
route add -iface 192.168.10.10 -link apic0:0.0.0.ca
SPC1: APIC Configuration

# to nmvc0 - Video Source
atm_ifconfig apic0 0x0000c2 open -aal5 -llc -besteffort
               -maxqueuebufs 100 -inport 1 -outports 2
route add -iface 192.168.10.2 -link apic0:0.0.0.c2

# to nmvc1 - Video Display 1
atm_ifconfig apic0 0x0000c3 open -aal5 -llc -besteffort
               -maxqueuebufs 100 -inport 1 -outports 2
route add -iface 192.168.10.3 -link apic0:0.0.0.c3

# to nmvc2 - Video Display 2
atm_ifconfig apic0 0x0000c4 open -aal5 -llc -besteffort
               -maxqueuebufs 100 -inport 0 -outports 1
route add -iface 192.168.10.4 -link apic0:0.0.0.c4
# to nmvc3
atm_ifconfig apic0 0x0000c5 open -aal5 -llc -besteffort -maxqueuebufs 100 -inport 0 -outports 1
route add -iface 192.168.10.5 -link apic0:0.0.0.c5

# to wooster
atm_ifconfig apic0 0x0000c6 open -aal5 -llc -besteffort -maxqueuebufs 100 -inport 1 -outports 2
route add -iface 192.168.10.6 -link apic0:0.0.0.c6
spc.cfg: unmodified: line 1 to gussie
# to gussie through gantry
route add 192.168.10.7 192.168.10.10
# open multipoint VC for video
atm_ifconfig apic0 0x0000d1 open -aal5 -llc -besteffort -maxqueuebufs 100 -inport 1 -outports 2
# open multipoint VC for CT
atm_ifconfig apic0 0x0000d6 open -aal5 -llc -besteffort -maxqueuebufs 100 -inport 1 -outports 2
atm_ifconfig apic0 0x0000c7 open -aal5 -llc -besteffort -maxqueuebufs 100 -inport 0 -outports 1
route add -iface 192.168.111.112 -link apic0:0.0.0.c7
# multipoint out
atm_ifconfig apic0 0x0000e1 open -aal5 -llc -besteffort -maxqueuebufs 100 -inport 0 -outports 1
route add -iface 192.168.111.111 -link apic0:0.0.0.e1
Linux on an SPC - What would we need?

- Interrupts
  - End of interrupt usage
- RTC?
- Boot procedure
  - stack parameters
  - symbol table
- Embedded file system
  - kernel config, utilities for building downloadable kernel
- APIC driver
  - Currently have:
    - skeleton
    - Berkley’s user space code
Preliminary SPC Performance (ANN Kernel)

• Throughput and Delay

• Processing:
  – active forwarding
  – payload adder

• Systems:
  – SPC (166 MHz P5)
  – 450 MHz PII
Test Setup

Delay Measurement

Delay Measurement

Ping through SPC

Throughput Measurement

Switch

SPC

CPU

MEM

APIC

Ping through SPC
Throughput (Forward)

![Graph showing throughput vs packet size for different packet sizes and processor types.]

- **SPC**
- **450 MHz PII**
Throughput (Adder)
Throughput - Mb/s (Forward)