

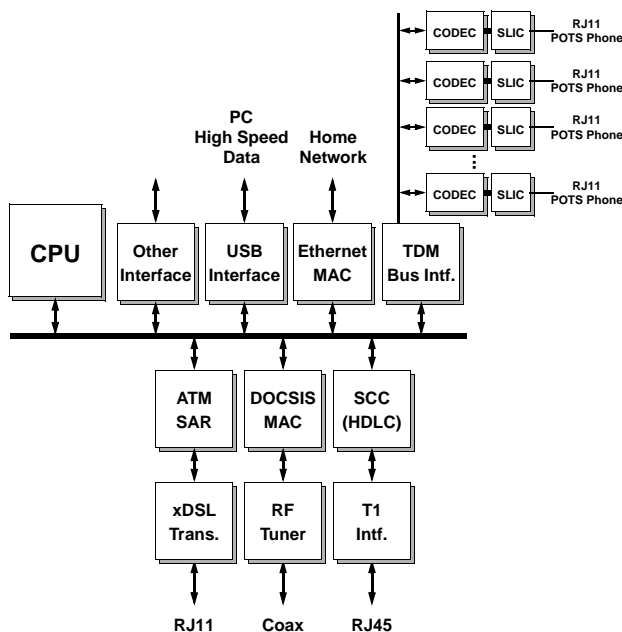


Architectural Considerations for CPU and Network Interface Integration

C.D. Cranor, R. Gopalakrishnan, P. Z. Onufryk

AT&T Labs - Research
Florham Park, NJ
{chuck,gopal,pzo}@research.att.com

Communications Processors



■ Applications

- Low-cost consumer devices
 - Broadband access devices
 - Packet telephones
 - Set-top boxes
 - Internet appliances

■ Design Challenges

- Cost
 - die size
 - package
- Time-to-market
- Flexibility

Communications Processor \neq CPU + NIC

Communications Processors

- ◆ **Single Chip**
 - Die size: 25 to 100 mm²

- ◆ **Shared Functionality**
 - Simple data link interfaces
 - Shared DMA controller

- ◆ **Low Latency**
 - Single bus
 - Tightly integrated memory controller

- ◆ **Small Buffers**
 - Motorola MPC860T 10/100 Ethernet interface has 448 bytes total for TX/RX FIFOs

- ◆ **Small Burst Transfers**

PCs/Workstations

- ◆ **System**
 - CPU die size 75 to 500 mm²
 - Popular ATM 25/155 SAR, 50 mm² in 0.5 μ m
 - Intel 82559 10/100 Ethernet controller, 34 mm² in 0.35 μ m

- ◆ **Replicated Functionality**
 - Standalone network interfaces
 - Each NIC has its own DMA controller

- ◆ **High Latency**
 - Multiple buses (CPU/memory and I/O)
 - Bus bridges

- ◆ **Large Buffers**
 - AMD Am79C975 10/100 Ethernet controller has 12K bytes total for TX/RX FIFOs

- ◆ **Large Burst Transfers**

Recipe for a Communications Processor

- **Define system architecture**

- **Develop application specific blocks**

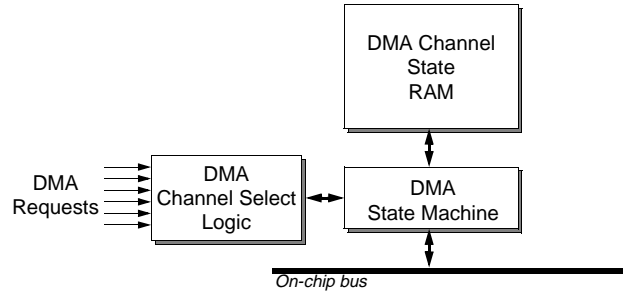
- **License CPU core**
 - Popular choices: MIPS, ARM and PowerPC

- **License commodity blocks**
 - 10/100 Ethernet MAC
 - USB
 - HDLC

- **Design a multi-channel DMA controller**

- **Tie the whole system together with on-chip bus**
 - Emerging standards (e.g., AMBA, CoreConnect, IP Bus) will simplify this task

Missing Ingredient – DMA

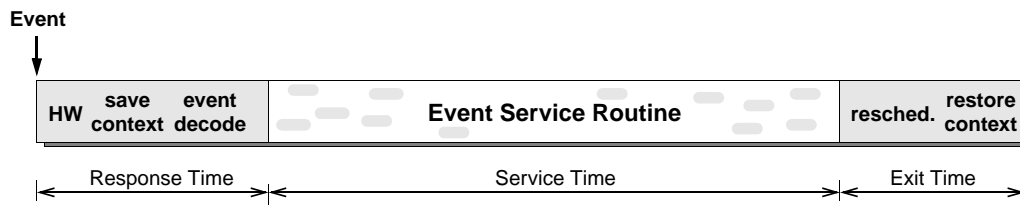


- **DMA often most complex block that must be designed**
- **Channels have slightly different requirements**
 - Different descriptor structure and information
 - Interface specific processing (e.g., ATM SAR, TDM mux/demux)
- **Features abandoned due to time-to-market pressure**
 - Cache consistency during DMAs
 - Unaligned data transfers
- **Alternate approach — add 2nd processor for data movement and interface specific processing**

UNUM

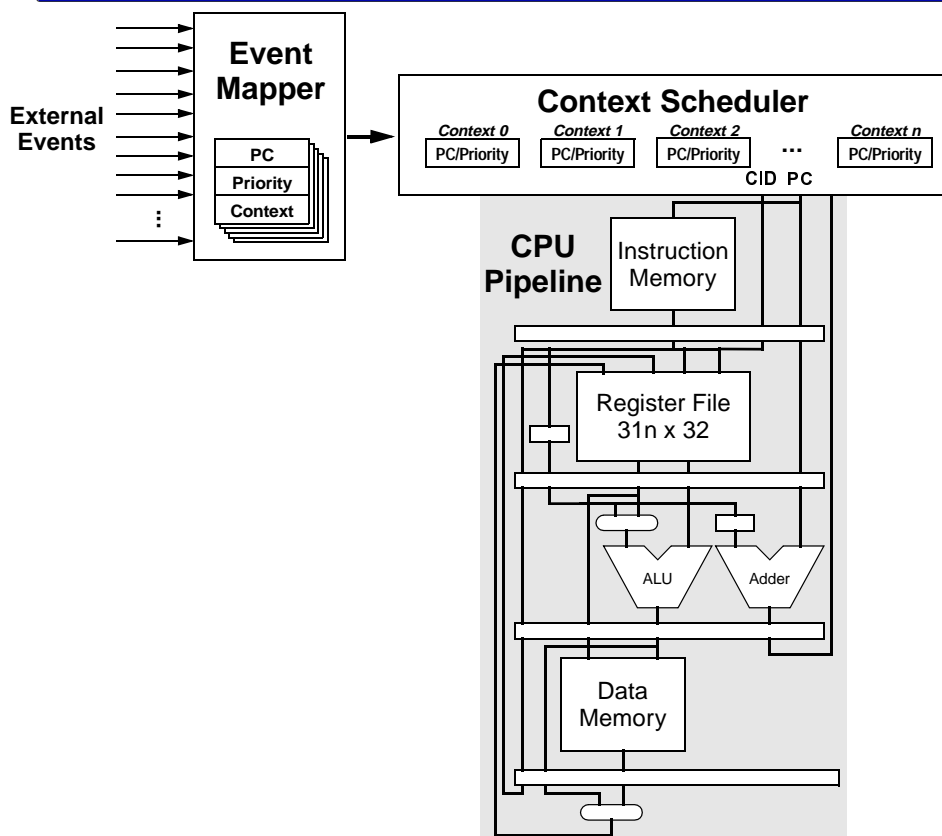
- **Goal: Single processor that performs all data movement, interface specific processing, and application processing**
 - Eliminates DMA
 - One software environment
 - Availability of standard UNUM core would reduce time-to-market
 - Provides flexibility
- **Standard CPUs not well suited**
 - Originally designed as workstation processors
 - Optimized for SPEC like benchmarks
- **UNUM approach: Integrate functionality required for data movement and interface specific processing**
 - Efficient event processing
 - High performance data movement
 - Can be used to enhance existing ISAs

Event Processing



- **Moving functions from hardware to software:**
 - Dramatically increases number of events
 - Decreases work performed in event service routines
- **Some processors have alternate register set for fast interrupts**
 - Interrupts are stateless, overhead still exists in retrieving and updating event service routine state
- **UNUM**
 - Multiple hardware contexts for event processing
 - Zero overhead context switching with state preservation
 - Priority based hardware scheduling of contexts

UNUM CPU



Data Movement

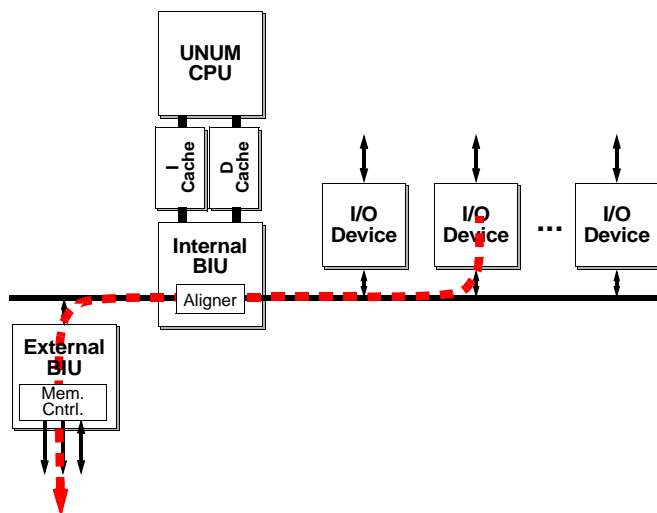
■ Problems with PIO

- Twice bus bandwidth of fly-by DMAs
- Inefficient unaligned address transfers
- Difficult to generate burst transfers
- Pollutes data cache
- Ties up CPU

■ UNUM integrates data movement instructions into CPU

- Data movement instructions support:
 - Fly-by transfers
 - Unaligned address transfers
 - Burst transfers
- Software implementation of data movement allows:
 - Customization of descriptor structure and information
 - Interface specific processing
- Concurrent execution with CPU pipeline possible (like multiplier)
 - Allows CPU to execute as long as no cache misses

UNUM Data Movement



■ TD2M r1, r2, r3

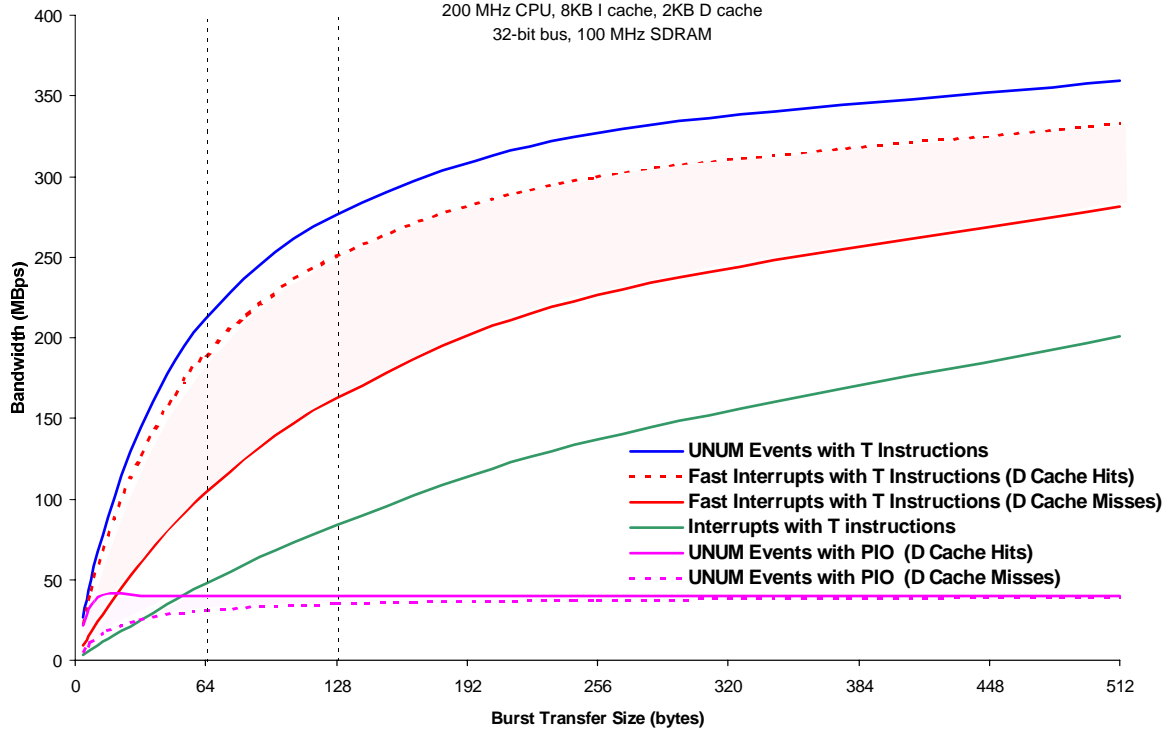
- Operands:
 - r1 - current address (ca)
 - r2 - byte count (bc)
 - r3 - device (dev)
- Aligner takes care of unaligned transfers
- D-cache entries invalidated during data movement
- I/O device can signal end-of-transfer
 - $r1 = 1 + \text{ending address}$

TD2M	ca, bc, dev	Transfer from Device to Memory
TD2C	ca, bc, dev	Transfer from Device to Cache
TD2MC	ca, bc, dev	Transfer from Device to Memory and Cache
TM2D	ca, bc, dev	Transfer from Memory to Device
TM2DD	ca, bc, dev	Transfer from Memory to Device and Discard

Data Movement Benchmark

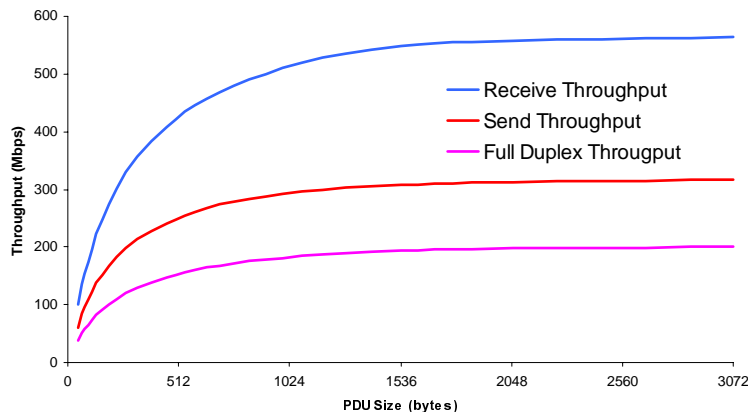
Transfer of 1518 byte Packet

200 MHz CPU, 8KB I cache, 2KB D cache
32-bit bus, 100 MHz SDRAM



UNUM ATM Soft-SAR

- Implemented ATM SAR in software using 3 UNUM contexts
 - Complete AAL5 SAR
 - Network Interface: Input/Output FIFOs, CRC-32 hardware, Utopia
 - Round robin transmit cell scheduling
- Maximum throughput at CPCS layer (infinite line rate and FIFOs)



- Full duplex at 25 Mbps requires only 13% of CPU
 - Requires only 4 cell input and output FIFOs

Conclusions

- **UNUM - Add CPU features for data movement and interface specific processing**
 - Eliminates DMA
 - Speeds time-to-market
 - Provides “full featured” data movement
 - Increases flexibility
 - Fast events + data movement ⇒ fly-by processing
 - Soft interfaces (e.g., ATM SAR)
 - Applications: Encryption, coding, overload control, packet classification, packet telephony

- **UNUM targeted at low-cost SOC designs for consumer devices**
 - May not be appropriate for other applications
 - PCs/workstations or parallel processors/SANs
 - High speed packet processors for switches and routers