

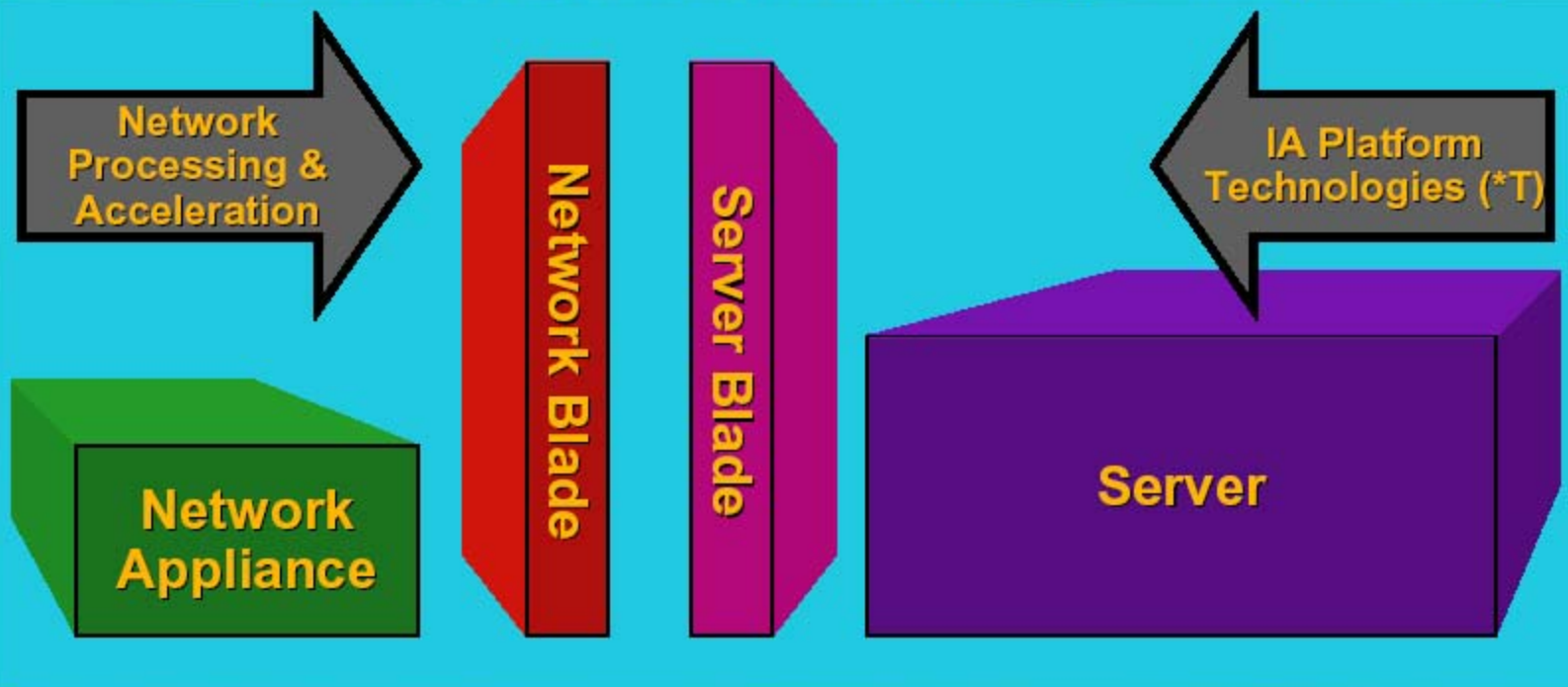
Overview of IXP2400 Architecture

- Intel 2nd generation network processor
- designed for network processing
 - goal to optimize for *fast path*, and accommodate *slow path*
 - programmable – fast deployment of new services, upgrades, protocols...
 - needs external control processor

Synergistic Co-Evolution of IA Platforms

In the timeframe of the end of the decade...

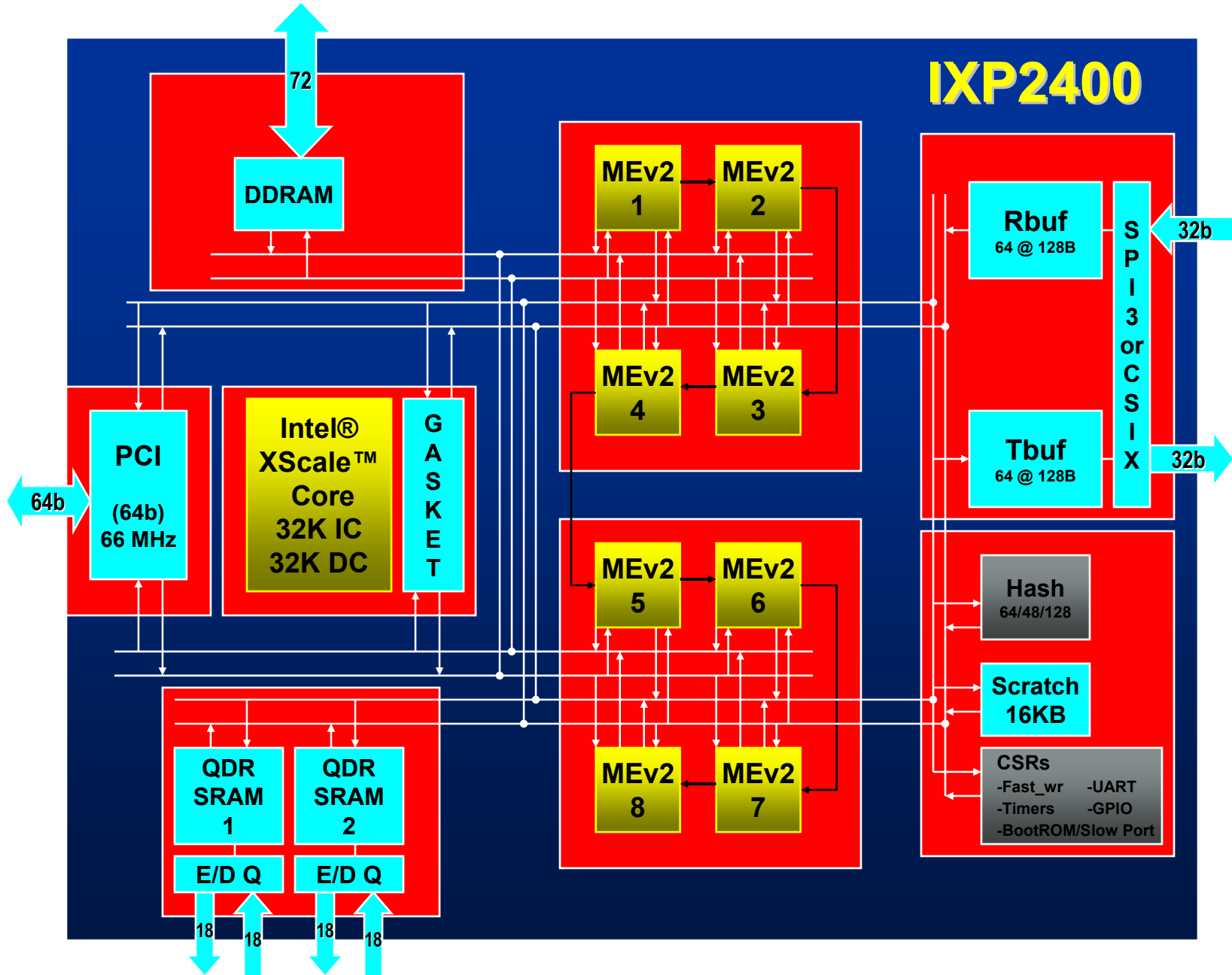
- 0.5 watt IA processors available for the smallest appliance
- 10X performance increase with many-core (double digit cores)



IXP2400 chip highlights

- 8 microEngines (uE)
 - RISC 600MHz processors with instruction sets targeting networking-related operations
 - for fast path
- XScale core
 - standard embedded CPU (ARM family) running Linux
 - slow path
- Media and Switch Fabric (MSF) Interface
 - to Rx/Tx packets from physical layer and/or switch fabric, i.e. main data path interface
 - separate 32b Rx and Tx busses @ up to 125MHz w/ 8kB buffers
 - various configs up to 4Gpbs (ours 3x1GbEthernet)
 - connects to external devices and is configurable for different bus protocols

- DRAM controller
 - for packet storage (up to 2GB, but access slow ~300 cycles, 64bit wide)
- SRAM controller
 - store packet queues, lookup tables..., ~150 cycles, 32bit)
- Scratchpad memory (16k)
 - on chip, for parameters, communication b/w uEs...
- PCI bus (64bit/66MHz)
 - connect to “host” – external control processor, or other PCI-compliant devices (e.g, chain IXPs together)
- Other units
 - Hash, control registers, other peripherals (timers, interrupt controller...), perf. monitors...



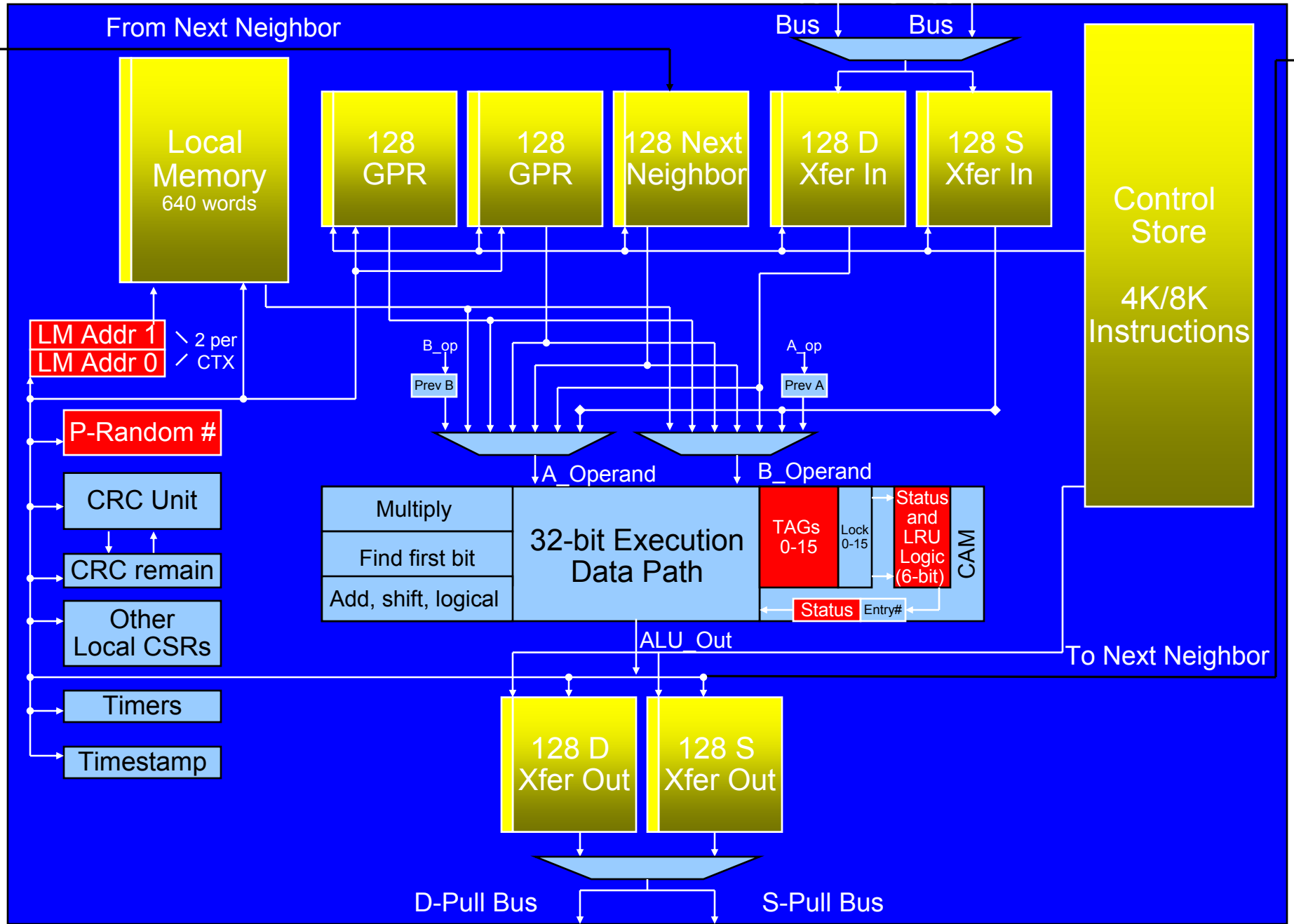
* John Morgan presentation at IXA University Summit04

Microengine MEv2

- control store
 - for instructions... 4k 40bit instruction per uE
- 8 hardware supported threads
 - hide memory access
 - separate register sets (PC) – fast ctx switch
- registers
 - 256 GPR (32/thd), 512 Xfer (4x16/thd – [D|S]x[R|W]), 128 NN (16/thd)
 - addressing context relative or absolute (GPRs)
 - local memory 640x32bits
- CAM – 16 entry cache

MicroEngine v2

*

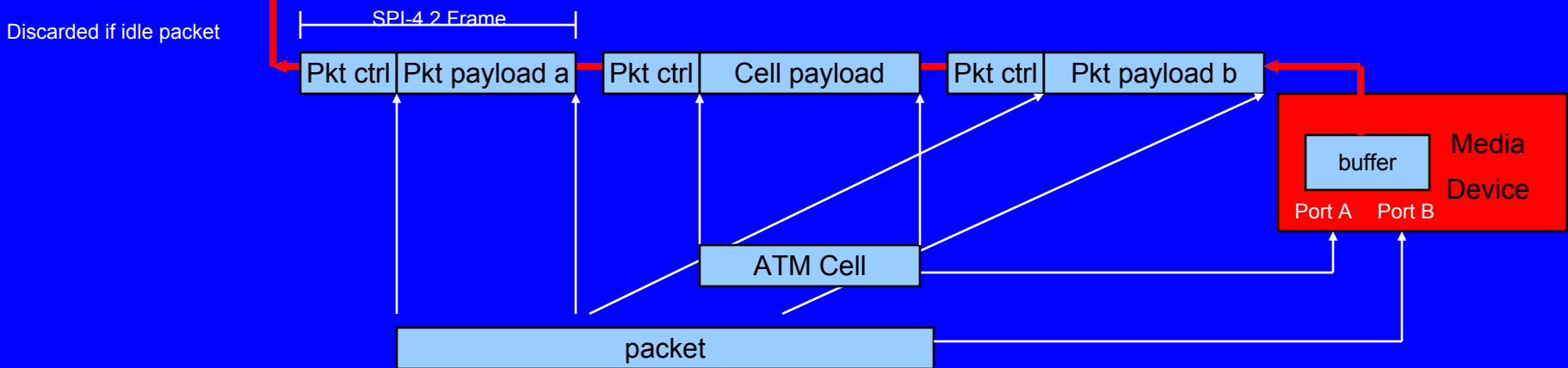
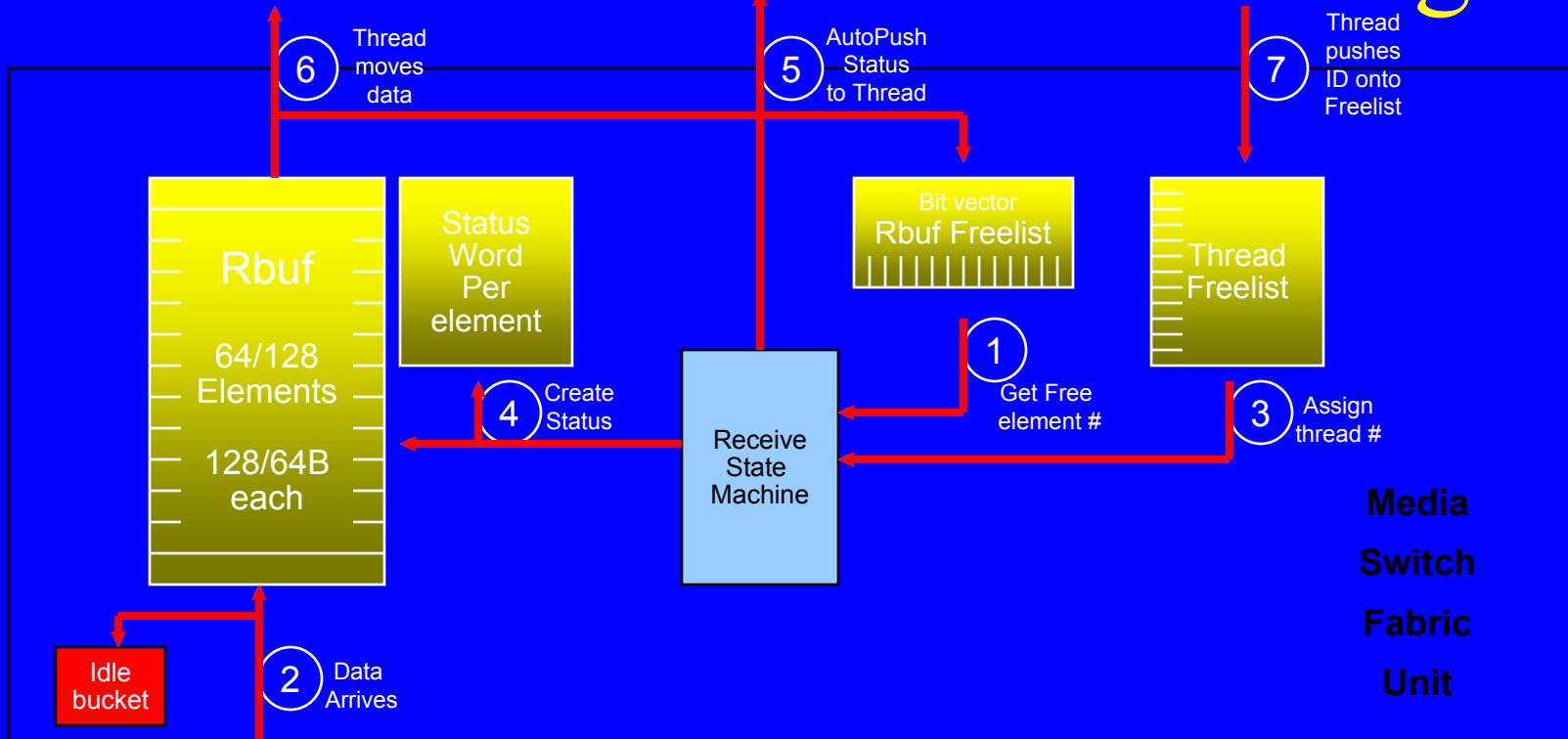


IPC

- messaging via circular queues (rings)
 - scratch/SRAM operations for ring management
- signaling
 - 15 signals per context
 - context can poll a signal explicitly
 - use to guard critical sections

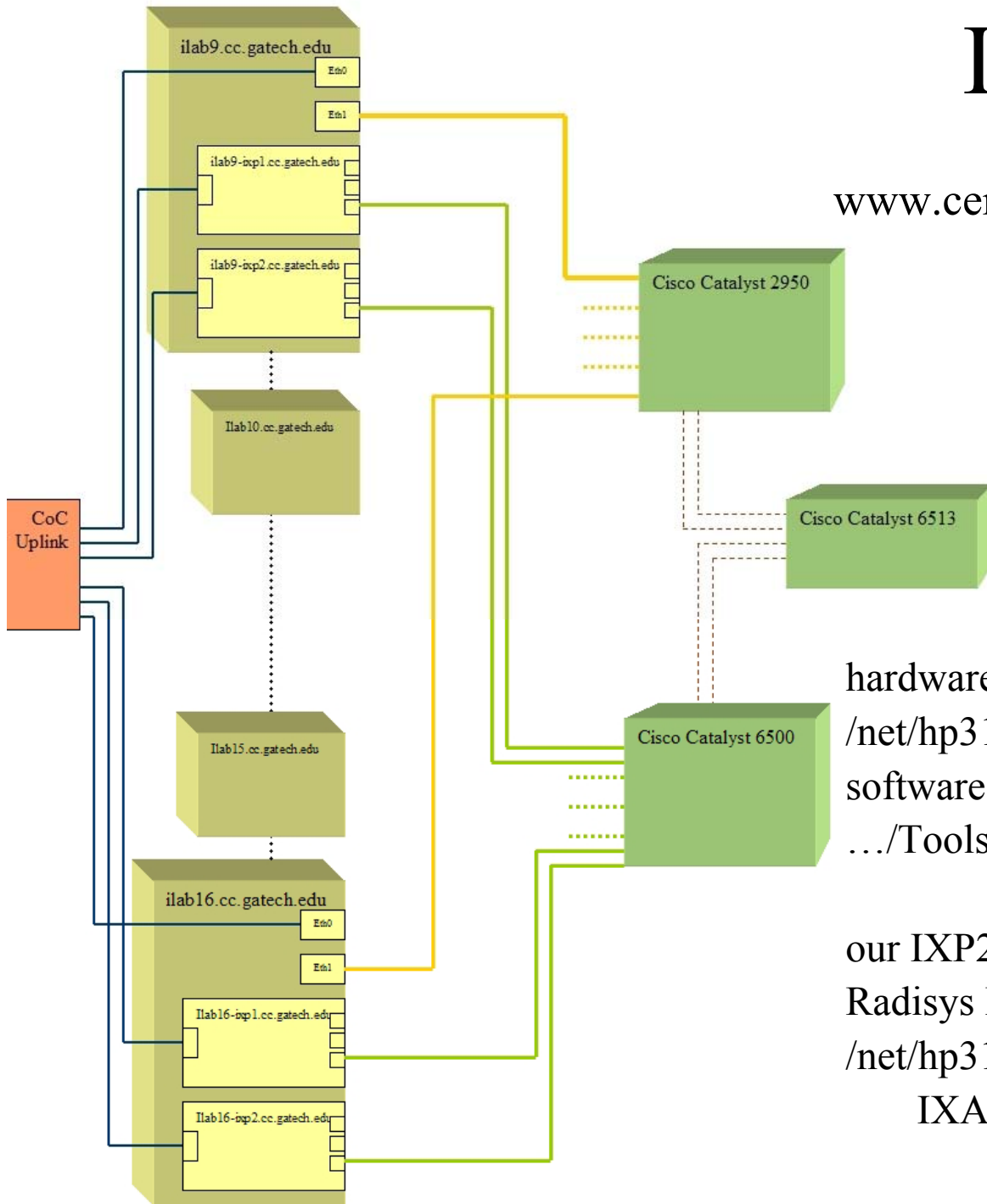
Media / Fabric Receive Logic:

*



ILAB setup

www.cercs.gatech.edu/projects/npg/ilab/

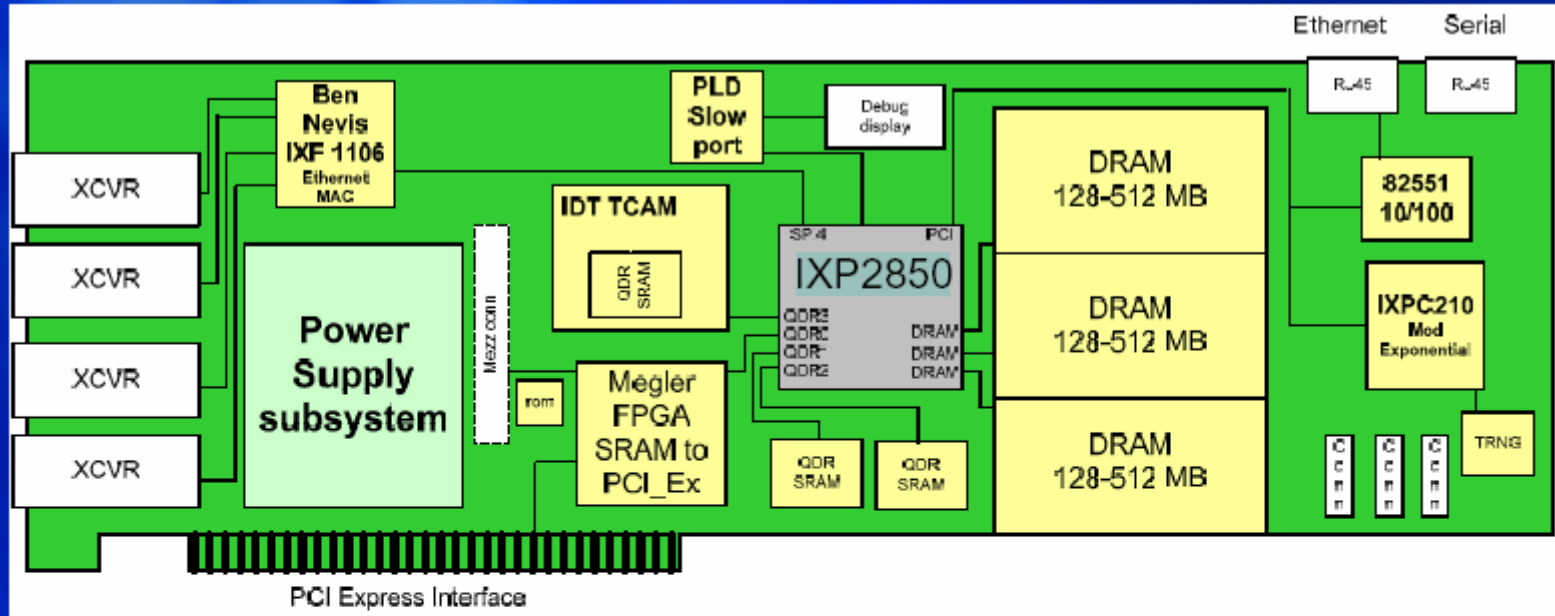


hardware manual:
[/net/hp31/ixpdev/sdk3 or 4.1/Docs/IXP2400/](http://net/hp31/ixpdev/sdk3%20or%204.1/Docs/IXP2400/)

software tools:
[.../Tools/...](http://net/hp31/ixpdev/Tools/)

our IXP2400 Boards:
Radisys ENP-2611
[/net/hp31/ixpdev/CD-Images/
IXA_Education_Wkstn3.1/Docs...](http://net/hp31/ixpdev/CD-Images/IXA_Education_Wkstn3.1/Docs...)

IXP285x / IXCP210 IA PCI Express Card



- **Prototyped Technology Evaluation Boards**

- PCI Express full form factor / Targeting 60 Watts power budget / (4) lane design
- 12 Volt power supply

- **Variations**

- Low power / low cost SKU achieved by not populating optional memory sites and using slower speed NPU and Math coprocessor

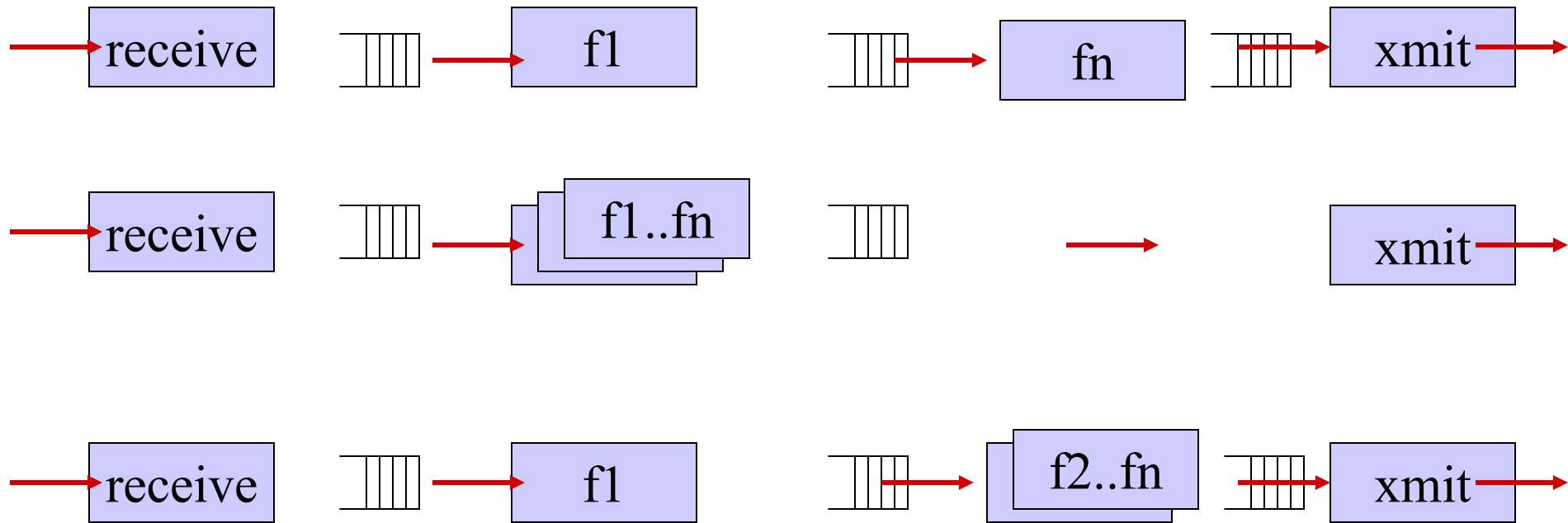
External Interfaces:

- (4) 1 Gbs Enet ports
- (1) 10/100 Enet port / (1) Serial port for debug
- (3) channels RDRAM supporting up to 256 MB per channel / 2 channels optional
- (3) channels QDR supporting up to 24 MB of QDR II SRAM / optional
- Up to 32 MB Strata-Flash
- 9 Mb TCAM / optional
- Mezzanine connector for additional functionality
- True Random Number Generator for security applications (250k words/sec)

IXP Software Development

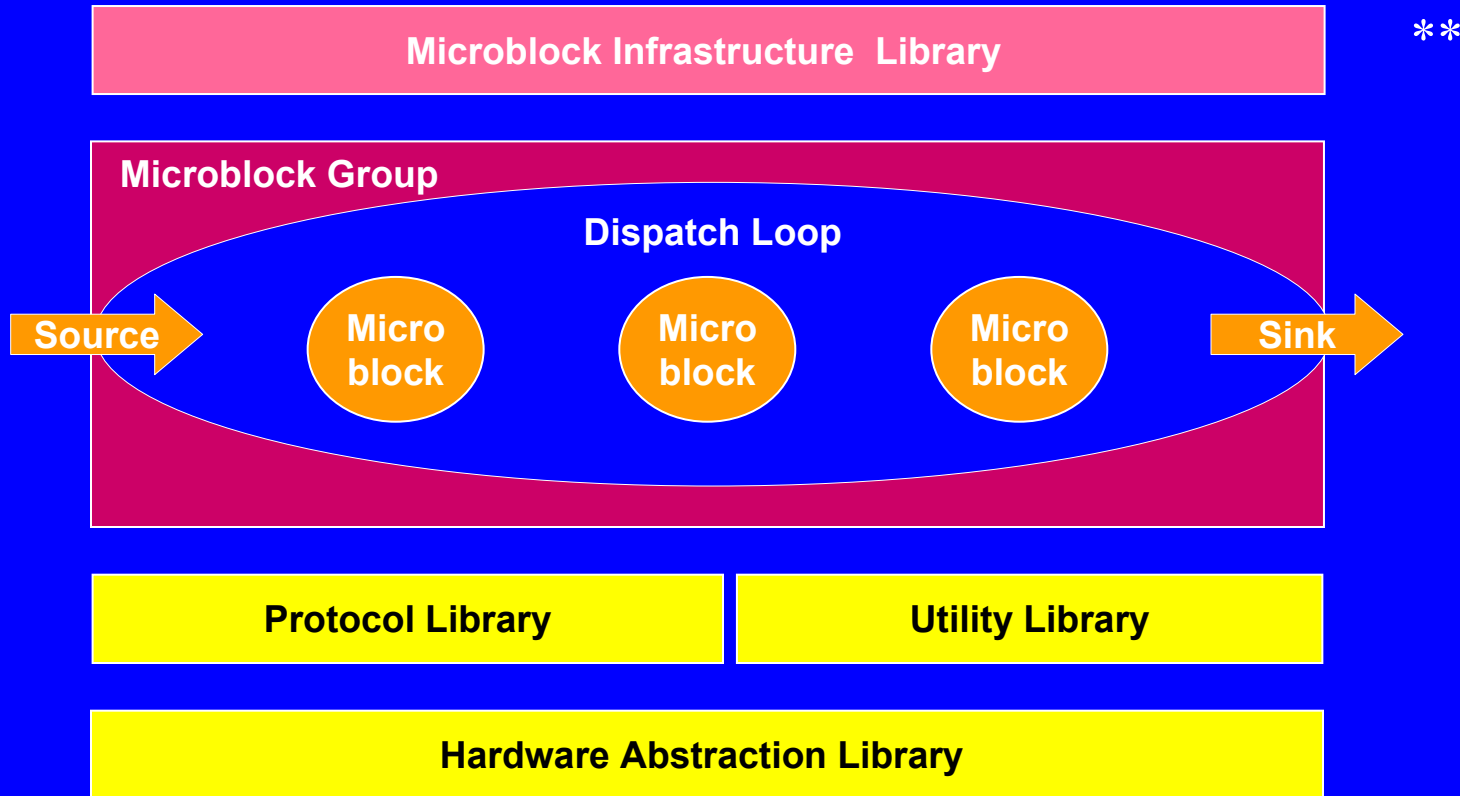
- Two issues:
 - How to structure/design your applications and
 - How to actually conduct the development/
programming

Programming model



issues: size of f_i , cyclecount for f_i ,
state across packets at f_i , state for a packet across all f_i s

Microengine Programming Model



- Hardware abstraction
 - OS-like functionality – APIs for memory manipulations (sram, scratch, dram...), buffer/queue management, MSF access...
- Protocol library
 - header field extraction, validation, update for popular protocols (ipv4, Ethernet...)
- Utility library
 - functions for hash table, CAM accesses, threads API...
- Infrastructure library
 - set up application specific packet meta data, pipeline parameters...
- Software framework
 - .../opt-links/ixp/IXA_?./ ... /dataplane-library,
 - ... microblocks-library
 - EDU_Wkstn ... sample_application – Radisys stuff
 - SDK_Tools/ src/libraby or me-tools

Programming the IXPs

- Rely on tools (Windows)
 - Software Development Kit – Development Workbench + Simulator (Transactor)
 - cycle accurate simulation (or so)
 - traffic generator
 - build DLLs to implement interaction with special components
 - e.g., custom protocols, interactions with XScale or host...
 - Architecture Development Tool
 - initial design and analysis
 - [/net/hp31/ixpdev/exports-sdk4.0/SDK/nassaupr8_noncrypto.zip](http://net/hp31/ixpdev/exports-sdk4.0/SDK/nassaupr8_noncrypto.zip)

Program in

- microC (familiar, portable),

or

- microcode (efficient, best use of platform features)

or

- mix of both

Build and run code with

- workbench tools (hardware mode)

or

- microcode assembler and linker on ilab and XScale
command line utilities on ilab-ixp
 - ssh ilabn -> telnet ilabn-ixp1 -> load, start, stop, sram...

Dispatch Loop Structure in Microcode

**

```
// include files
#include "dispatch.uc"
#include "dl_system.uc"
#include "IPv4_Fwd.uc"
#include "l2_encap.uc"
#include "l2_classify.uc"

Init#:
// The following code is
// executed once
DL_Source_Init []
IPv4_Fwd_Init []
L2_Classify_Init []
L2_Encap_Init []
DL_Sink_Init []
```

```
// The following loop runs for
// every packet
loop#:
// Get a packet from the Scratch
// Ring
DL_Source []
// Perform Layer-2 Classification
// on the packet
L2_Classify []
// call the IP forwarding block
IPv4_Fwd []
// Perform Layer-2 Encapsulation
L2_Encap []
// pass the packet to next
// Microengine
DL_Sink []
BR [ loop# ]
```