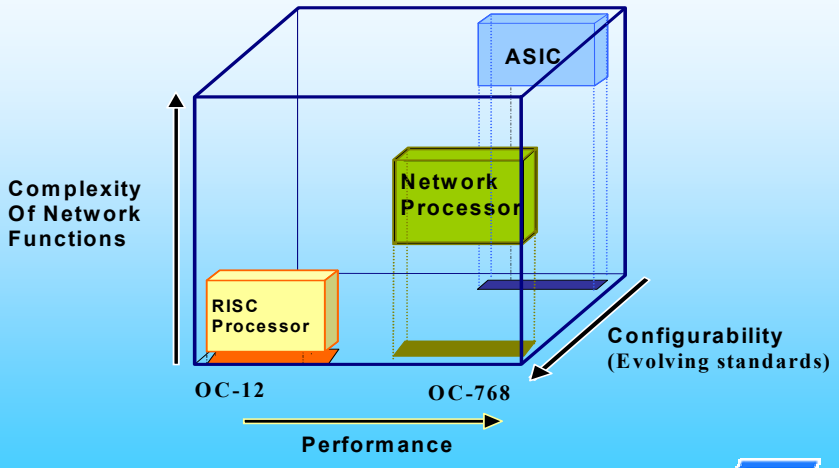
Outline

- Motivation
- Network Processor Complexity
- Methodology and Architecture

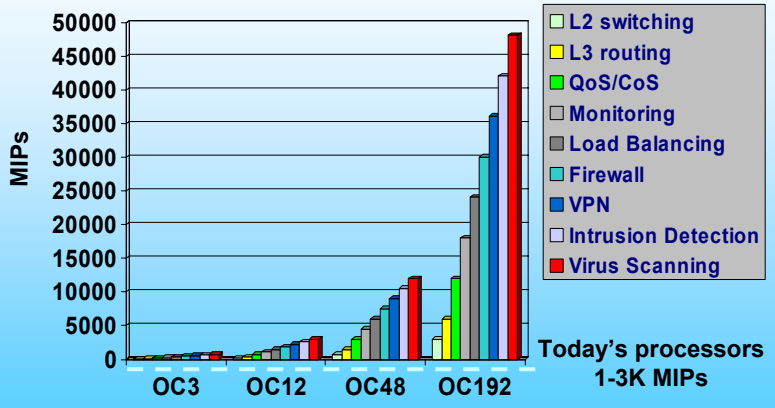
Motivation

- Speed Requirement
- Communication Requirement

Need for Network Processor



MIPS Requirements for Network Processing



Need for highly concurrent SoC architectures

*Sterling Research Report, 2000

Why special-purpose NP?

- Processing Time budgets

Media	*Cell/Packet size*	*Packets/Sec*	*Time/Packet*
10 Mb Ethernet	64 - 1518	14.88k - 800k	67.2-1,240 uS
100 Mb Ethernet	64 - 1518	148k – 8k	6.72 – 124 uS
Gb Ethernet	64 - 1518	1.48M – 80k	672nS – 12.4 uS
OC-3	53	~300k	~3.3 us
OC-12	53	~1.2M	~833 nS
OC-48	53	~4.8M	208 nS
OC-192	53	~19.2M	52 nS
OC-768	53	~76.8M	13 nS

Requirements for a Network Processor

- Requirements for OC-768 network processing
  - 114 million packets/sec (44 bytes/packet)
  - Processing time < 9ns/packet
  - Assumption: forwarding + classification = ~500 instructions
  - Requirement: 57 GIPs
  - Need for multiple GHz processors
  - Packet Classification
    - Lakshman and Stiliadis Proceedings of ACM SIGCOMM, Sept. 98
  - 50 memory accesses/packet
  - Requirement: 5.7 x 109 memory accesses/sec
  - Need for multiple memory components
- Need for multi-processor/distributed memory architecture
- Need for concurrent, high-speed on-chip communication

Requirement .....

- Requires huge computing power
  - ~5.7GIPS for OC-192
  - . . . and getting worse
- Requires huge memory bandwidth
  - data comes in at 10Gbps (OC-192) and 40Gbps (OC-768)
- Inherently parallel
  - frame doesn't depend on previous or next one
- Data-driven
  - driven by data (operand) availability
  - asynchrony

Network Processor Complexity

- Functional Complexity
- Architecture Complexity
- System Design Complexity
- Verification Complexity

Functional Complexity

- State-of-the art Functions of general-purpose processors:
  - Well known properties
  - Existing processors are well defined
  - Simulation with established benchmarks
- Network Processors are application-specific processors
  - Application space known ...
  - However, very complex set of functions:
    - packet classification, forwarding, scheduling
  - Properties to verify not all known
  - Evolving standards
    - Can test suites be developed?

Functional Complexity

- Segmentation and Reassembly (SAR)
- Protocol Recognition and Classification
  - Identify frames based on information such as protocol, destination/source address, etc
- Queuing and Access Control
  - Queue frames awaiting further processing (prioritization)
- Traffic Shaping and Engineering
  - Meet delay/jitter requirements
- Quality of Service (QoS)
  - Tag frames for processing in subsequent devices

Source: Agere, Inc

Architectural Complexity

- Network processing is a dataflow problem
  - Locality inter-packet is poor. uP cache does not help.
  - A lot of pointer-chasing which requires
    - Cache thrashing
    - uP stalls during these indirections
- IPC dramatically reduces because of memory latencies.
- Caches exploit locality. Data structures accessed per packet exhibit poor temporal locality.
- Time budget requirement per packet is too high for regular microprocessors.

Architectural Complexity

- The faster the network port the likelier for more unrelated streams.
- A lot of alignment issues.
- Branch prediction ineffective
  - > 90% taken for DSP
  - 50/50 for some network applications

Architectural Complexity

- Network has two conflicting requirements *programmability and speed*.
  - Network processors must support those two requirements where the traditional micro processors can't.
- Current Network Processors have relied on duplicating/copying the ASIC paradigm on a chip.
  - Either copying some off-the-shelf processors with a few additions and tying them together the same old fashion way.
  - Or making some minimal modification for product differentiation purpose.
  - Besides, many of the current Network Processors are very difficult to program.
- System houses demand platform solutions from manufacturers. They can no longer afford point product solutions.

Architectural Complexity Computations

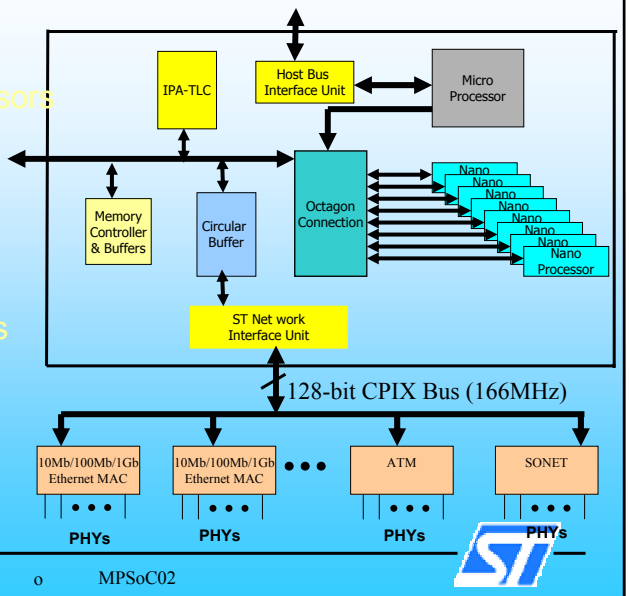
- Provide Specialized Network Instructions to achieve more with less instructions.
- Fuse several appropriate primitives to enhance performance as it is done in the case of Multiply Accumulate
- Add more predicate to reduce branch penalties

Architectural Complexity Computations

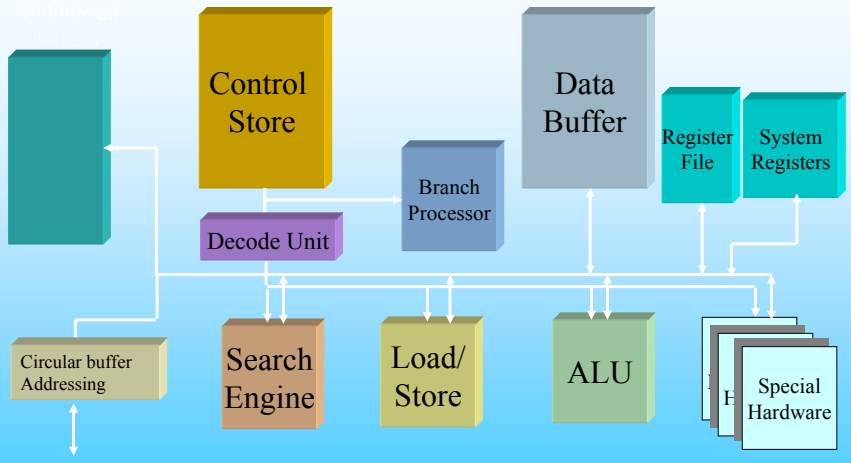
- Use more computational processing units as needed In:
  - pipeline fashion
  - Parallel fashion

Network Processor Architecture

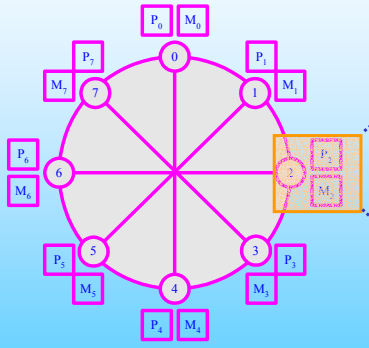
- Multiple Nano-processors
- Complex on-chip interconnects
- High-speed memory components
- High-speed Interfaces



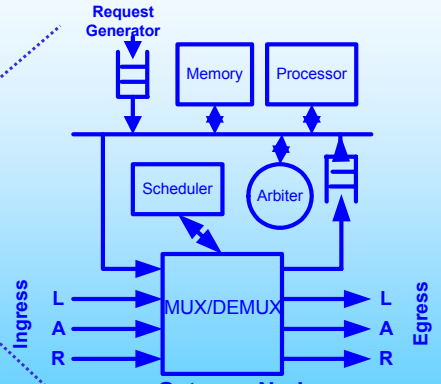
Nano-Processor Programming Model



Octagon On-Chip Communication

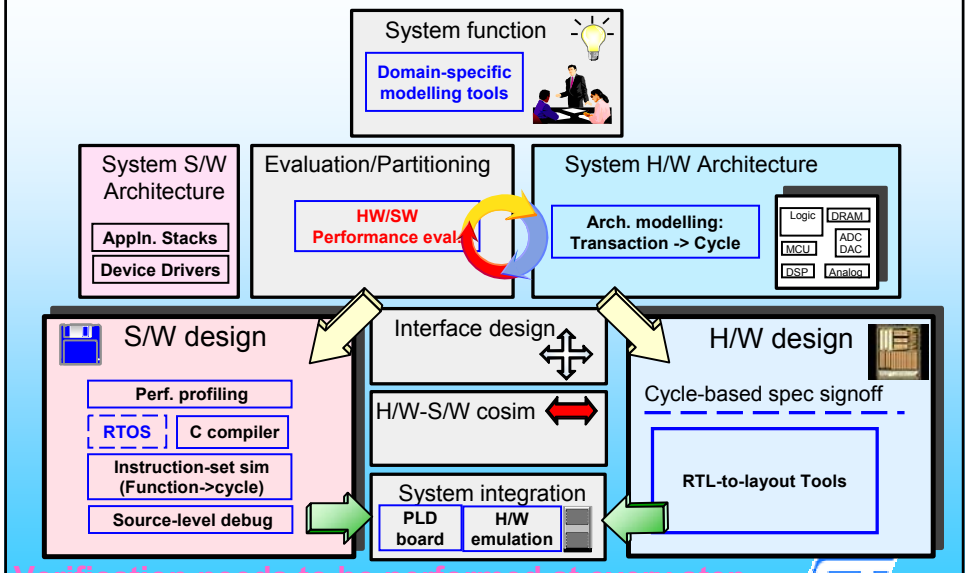


Network Processor using Octagon



Octagon Node Model

System-level Design Complexity



Verification needs to be performed at every step individually and collectively

Design Validation Challenges

Due to:

- Functionality Complexity
- Architecture Complexity
- Embedded Application Software Complexity
- Design Methodology Complexity

Functional Complexity

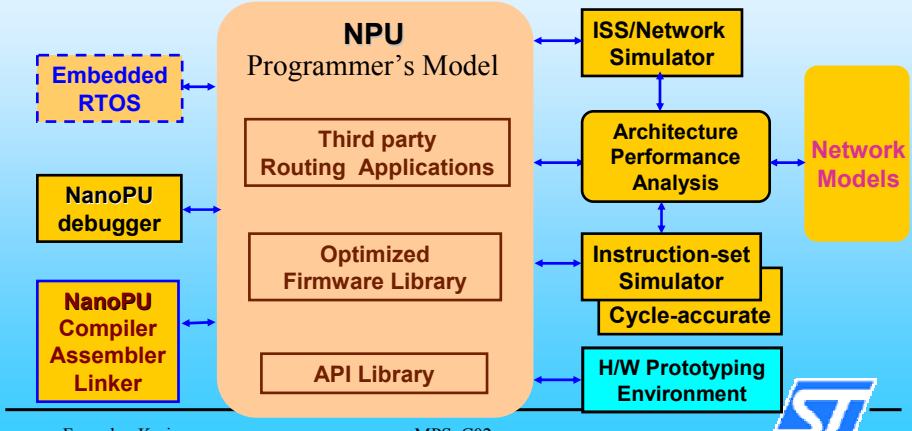
- State-of-the art Verification/Validation of general-purpose processors:
  - Property checking of well-established properties
  - Validation test suites of known processor functionalities
  - Simulation with established benchmarks
- Network Processors are application-specific processors
  - Application space known ...
  - However, very complex set of functions:
    - packet classification, forwarding, scheduling
  - Properties to verify not all known
  - Evolving standards
    - Can test suites be developed?

Architectural Complexity

- State-of-the art in verification/validation:
  - processor: formal and simulation-based techniques for a single processor
  - hw/sw co-designs: co-simulation of single processor-based co-designs
- However, network processors/ASICs are very complex hardware/software co-designs
  - Multiple embedded processors
  - Multi-threading, parallel processing, pipelining
  - Mix of homogenous and non-homogenous processors
    - nano-processors and control processor
  - Multiple co-processors/hardware accelerators
    - for packet forwarding, packet classification, queue management

Software Complexity

- Complex set of application, firmware, and development software
- Need for comprehensive set of software debugging tools
- Need for real-time verification through hardware prototyping environments



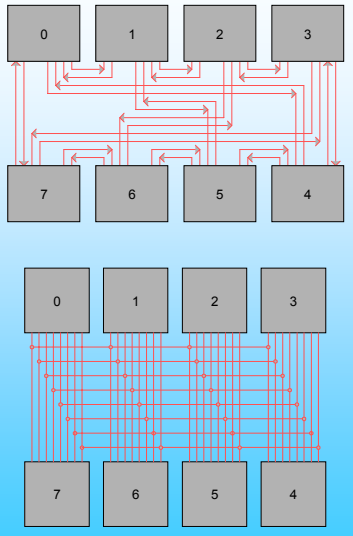
Test Challenges

- Use of GHz clocks, multiple clock domains
  - To meet OC-192/OC-768 speeds
  - Need for at-speed test
- Ultra-Deep Sub-micron technologies
  - Need test for noise problems
- Deployment in optical networks
  - Need for mixed-signal/mixed-domain test

Need for At-Speed Test of GHz Chip

- May lead to excessive test cost
  - v. high cost for GHZ external tester
- Self-test a viability
- LFSR-based self-test may not be well-suited for multi-processor SoC
- Need to look at alternative self-test methodologies

Complex, Long On-Chip Interconnects



Octagon

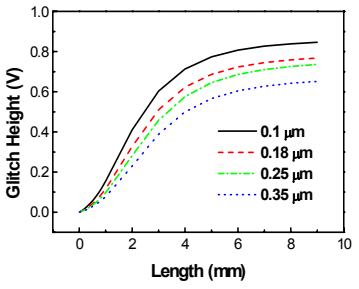
Nodes	Horizontal Links #/max length(mm)	Vertical Links #/max length(mm)
8	12/8	12/0.156
15	24/8	24/0.156
22 (shown)	36/8	36/0.156

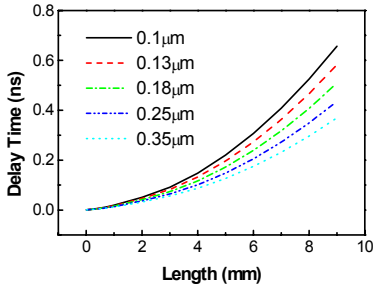
Crossbar

Nodes	Horizontal Links #/max length(mm)	Vertical Links #/max length(mm)
8	8/8	32/0.108
15	15/16	120/0.192
22	22/22	242/0.276

- Very large number of wide buses
  - 24 32-bit busses for 8-node Octagon
  - 40 32-bit busses for 8-node Crossbar
- Interconnects very long
  - proportional to # of nano-processors

Significant DSM Noise Potential





- Most affected:
  - Long Interconnects in High speed/low voltage DSM Socs
- Network Processors are susceptible since
  - Use of Ghz, v. long, nano-meter interconnects

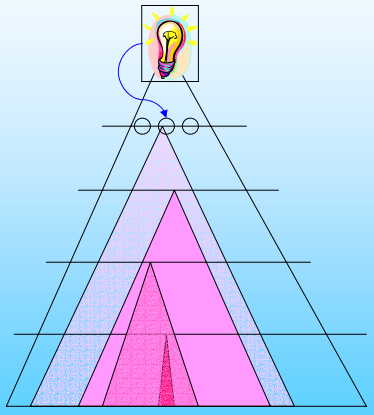
Mixed-Signal/Mixed-Domain Testing

- Optical Networks: Integration of digital, analog, and optical components in network chips

Methodology & Architecture

- Abstraction Layers
- New Pyramid of Design
- Putting it all together

Abstraction Layers



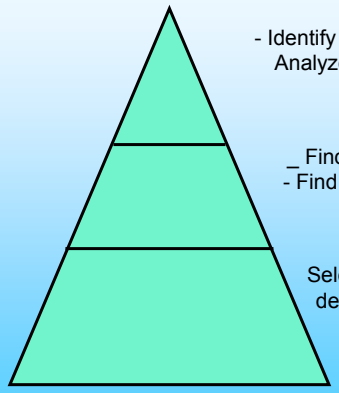
Abstraction Layer	Modeling Type
Functional	Functional
Architecture	Processes
RTL / u-Arch	Cycle-based
Gate-Level	Boolean
Silicon	Circuit

The New Pyramid of Design

Application

Scheduling

Architecture



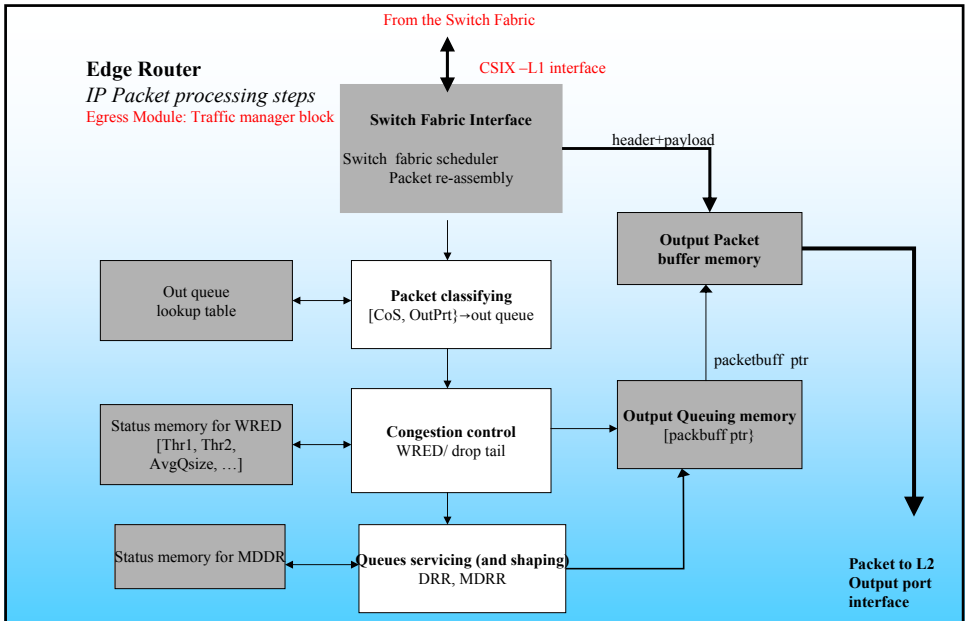
- Identify The Application and Analyze all its component

_ Find parallelable Functions
- Find Pipelinable Functions

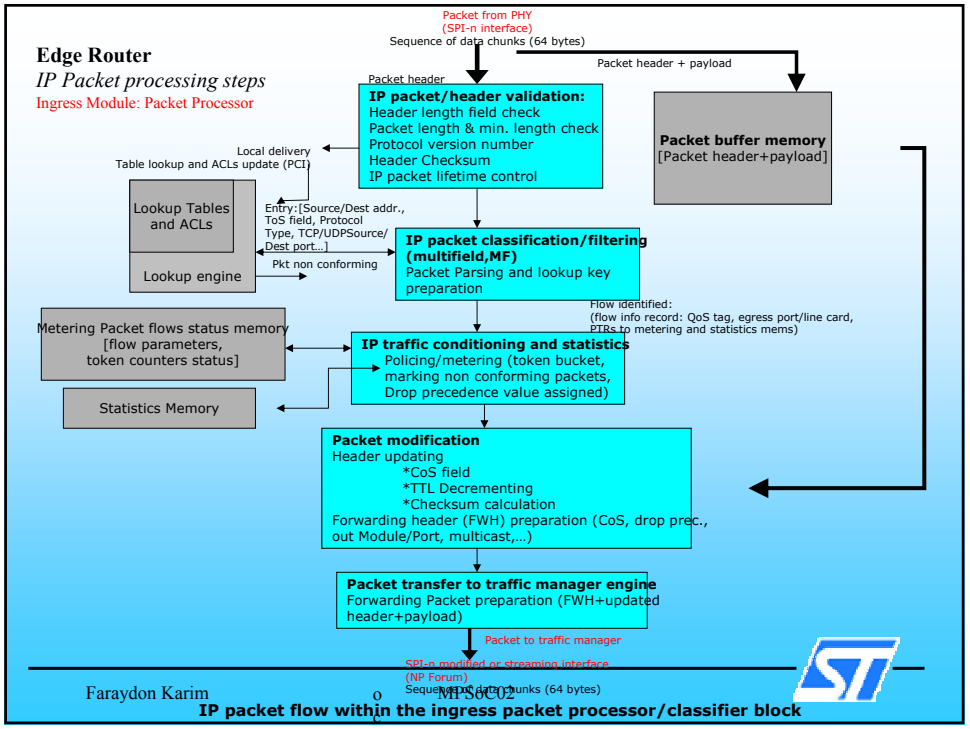
Select The Components for desired Cost/Performance

Application

- Understand the Application
- Draw the Flow of the Application



IP packet flow within the egress module



IP packet flow within the ingress packet processor/classifier block

Scheduling

- Define The Processes
- Divide each to sub-processes
- Find the process and sub-processes that can run independently to events
- Chose the components that can execute these events.

Scheduling -2

- An event is the smallest part of application that can not be sub-divided to run in parallel or meaningfully in sequence
- Name the components that can execute each event according to the cost and performance
- Select the communication devices that can move data between components according to the desired cost and performance

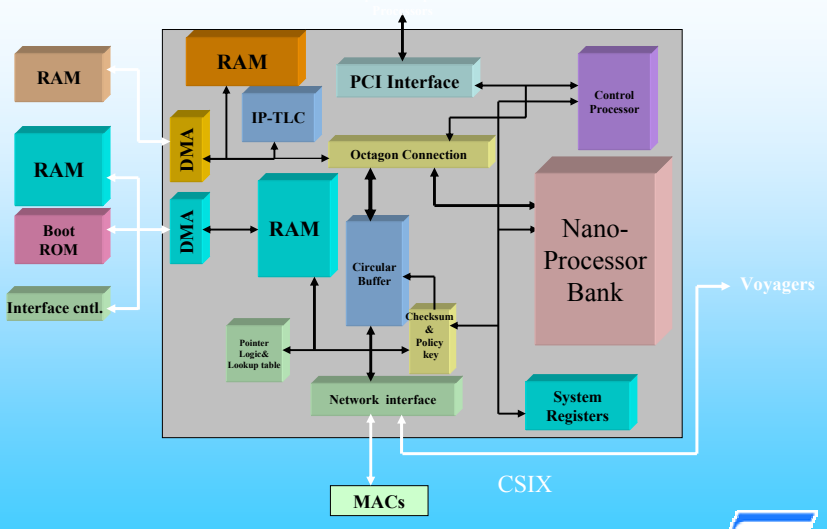
Scheduling -3

- Draw the final events flow
- Measure the desired speeds
- Find several closest flow for the desired goal.
- Select the easiest achievable one

Architecture

- Select the component(s) for the events
- Connect the components with good communication architecture
- Control the move and dispatch of all events
- Re-evaluate the final goals.

NP Architecture



Conclusions

- Systems are getting too complex
- Design bottom up can not satisfy the desired cost/performance
- There must design through several layers of abstractions
- Applications must drive the component selections