Complex Network in a Chip

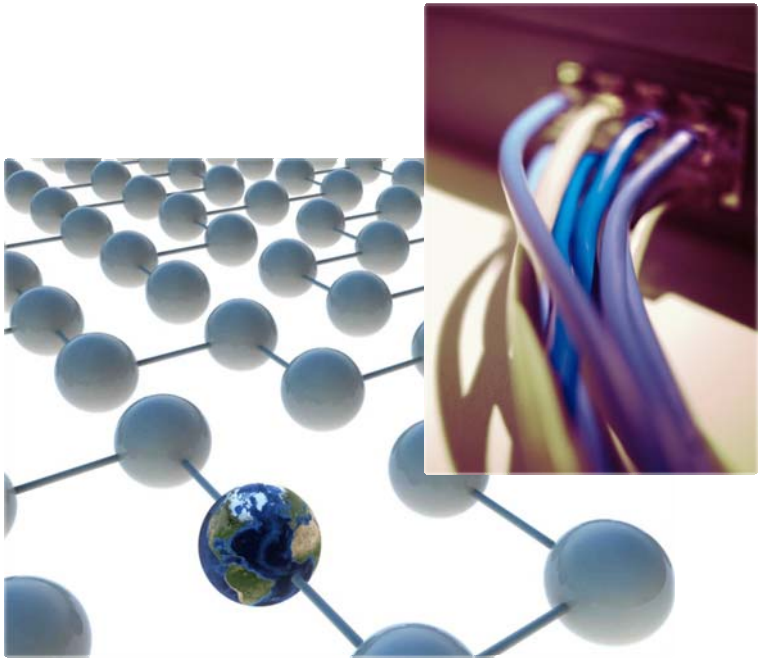
Sungho Kang

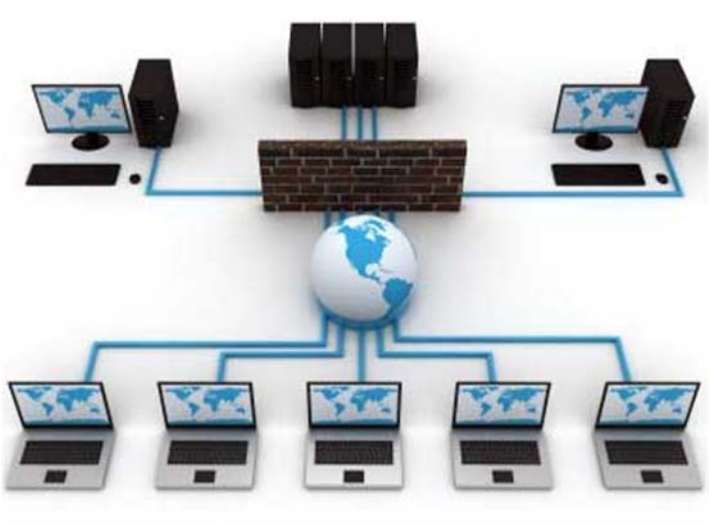
Computer Systems & Reliable SoC Lab.

Contents

- Introduction
- Network Processors
  - Background
  - Routers
  - Network Intrusion Detection Systems
- Network on Chip: NoC
  - Background
  - NoC Design
  - NoC Implementations

Network and SoC





Computer Network is constructed using SoCs; network interface cards, bridges, switches, routers.

Network and SoC

Communication subsystem on SoC can be constructed by Bringing systematic networking methods.

Network Processors

Background

Routers

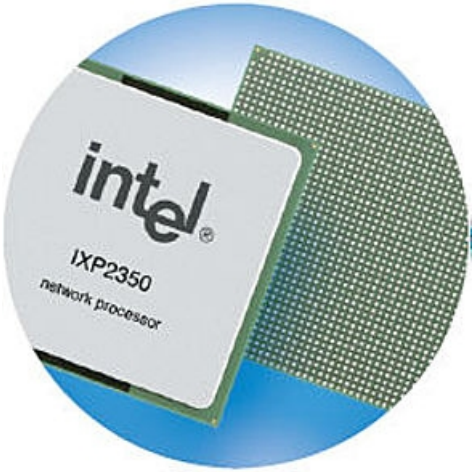
: Address Lookup

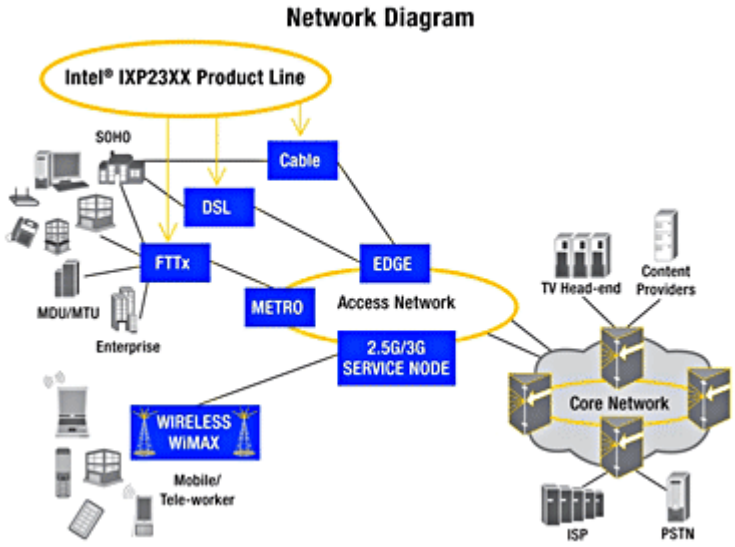
Network Intrusion Detection Systems (NIDS)

: Deep Packet Inspection (DPI)

Network Processors

- Integrated circuit which has a feature set specifically **targeted at the networking application domain**.
- Software programmable devices
  - have generic characteristics similar to general purpose central processing units that are commonly used in many different types of equipment and products.





Network Processors

- Network processors are used in the manufacture of many different types of network equipment such as:
  - **Routers**, software routers and switches
  - Firewalls
  - **Intrusion detection systems (NIDS)**
  - Intrusion prevention systems
  - Network monitoring systems.

Network Processors

Background

Routers

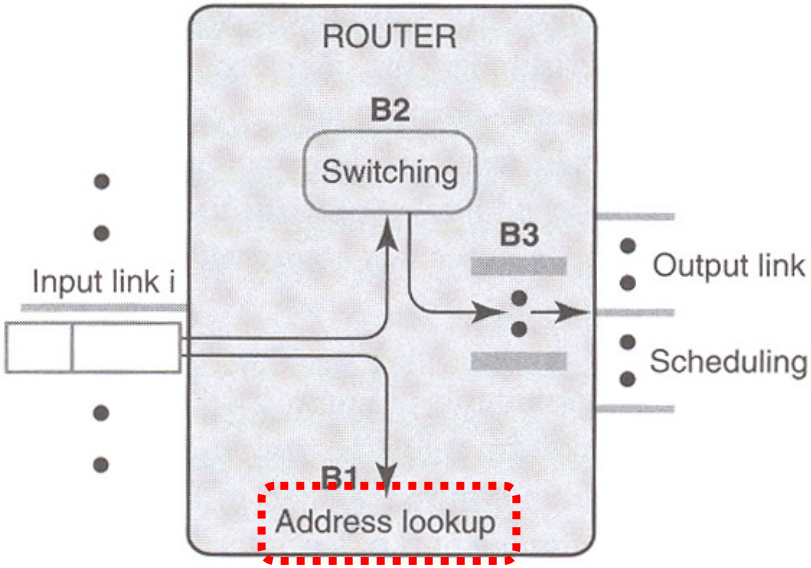
: Address Lookup

Network Intrusion Detection Systems (NIDS)

: Deep Packet Inspection (DPI)

Routers

- Router
  - Device that interconnects two or more computer networks, and selectively interchanges packets of data between them.
  - Each data packet contains address information that a router can use to determine to where transfer the packet.



Routers

- Three Main Bottlenecks in a Router
  1. Lookup
  – FIB (forwarding information base)
    - A set of prefixes with corresponding output links
    - Longest prefix match
    - A dedicated processor per input link interface (Cisco's GSR family)
    - A dedicated chip with some degree of programmability (Juniper M-160)
    - Network processor for more programmability (web load balancing)

Routers

- Three Main Bottlenecks in a Router

2. Switching

– Transfer the packet from link I to output link J
  - N parallel buses
  - Hard to schedule the switch
  - Matching available inputs and outputs every packet arrival time

3. Queuing

– If output link is congested, packet has to be placed in a queue
  - First-In-First-Out Queue
  - More sophisticated scheduling for fair bandwidth allocation and delay guarantees

Address Lookup

- Major function in packet forwarding
  - Lookup the destination address of incoming packets in forwarding table
- Next-hop information from forwarding table
- 32b IPv4 address divided into network part (prefix) and host part

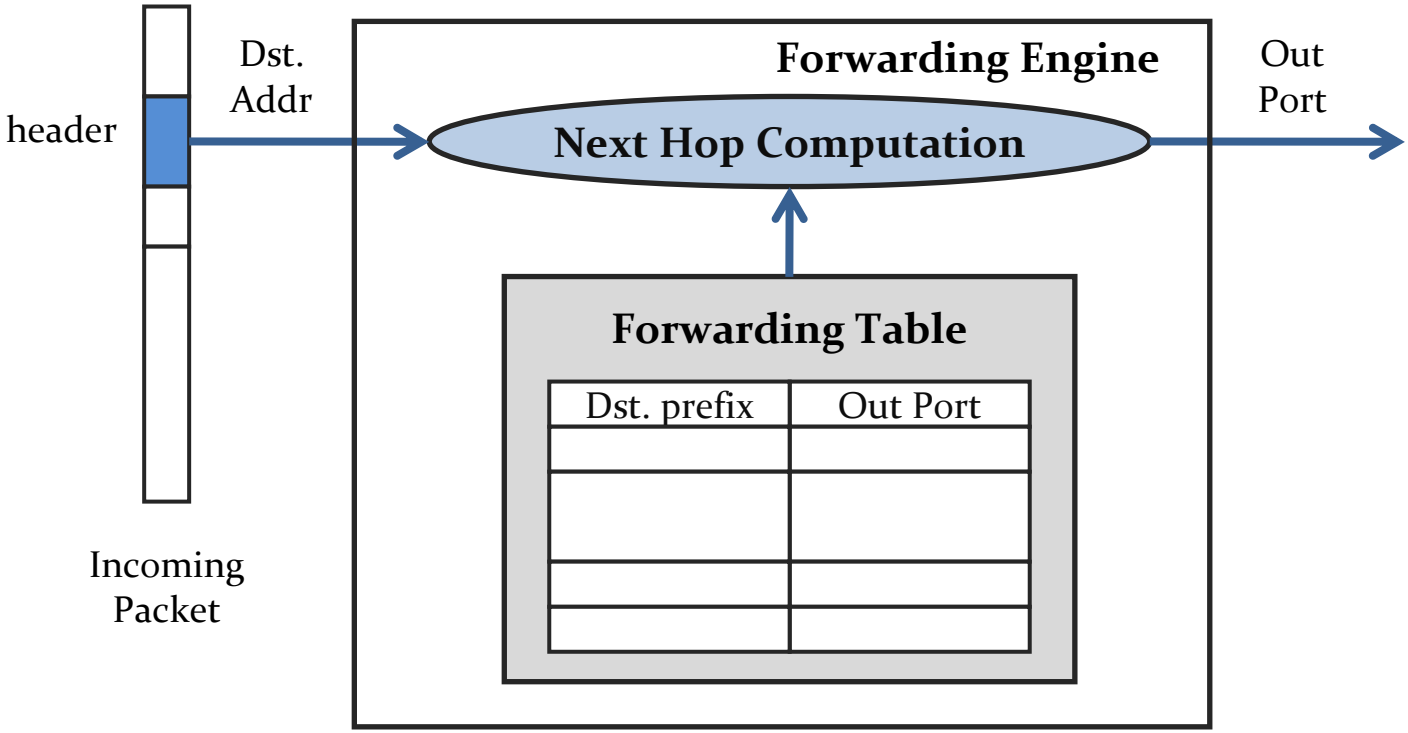
Address Lookup

- Class-based Addressing Scheme
  - Class A (8b), Class B (16b), Class C (24b)
  - Standard techniques for exact match are used for IP route lookup
    - Perfect hashing, binary search, standard CAM
  - Inflexible and wasteful of address space
  - To make better use of scarce resource
    - Bundle of class C network address given out instead of class B
    - Result in massive growth of routing table

Address Lookup

- Classless Interdomain Routing (CIDR)
  - Arbitrary aggregation of network address to reduce routing table entry
  - Better use of available address space
  - More efficient mechanism required to perform the IP route lookup
  - Address prefix is represented by <address/prefix length>
  - Prefix length indicates the number of network part in the address
  - Search is done in a longest matching manner to find the most specific route
  - Longest prefix match harder than exact match
    - Destination address of arriving packet does not carry with information of the prefix length
    - Search all prefix lengths

Address Lookup



The prefix length of incoming packet is not fixed,
and hence **Longest Prefix Match** is required.

Address Lookup

- Metrics for comparison of lookup algorithm
  - Lookup speed
    - Explosive growth of link bandwidth requires faster IP lookups
    - Ex) 10Gb/s link carry 31.25 Mp/s (assume 40bytes packet)
  - Storage requirement
    - Small storage means high memory access speed and low power consumption
    - Important for cache-based software algorithm and SRAM-based hardware algorithm
  - Update time
    - Algorithm should be able to perform 1k update/s to avoid instability
  - Scalability
    - Size of forwarding table will increase at a speed of 25k entries/year
    - Ability of an algorithm to handle large forwarding table
  - Flexibility in implementation

Address Lookup Scheme 1

- Trie-Based Scheme: Standard Trie Structure
  - Each node contain two pointer
    - 0-pointer(left pointer)
    - 1-pointer(right pointer)
  - Node X at level h
    - Set of all route prefixes that have same first h bits
    - Depending on the value of (h+1)bit, pointer of node X points to the corresponding sub-trie if it exists
  - Based on the value of each bit of the destination address of packet, lookup algorithm determines the next node to be visited
  - Next hop information of longest matching prefix is maintained while the trie is traversed

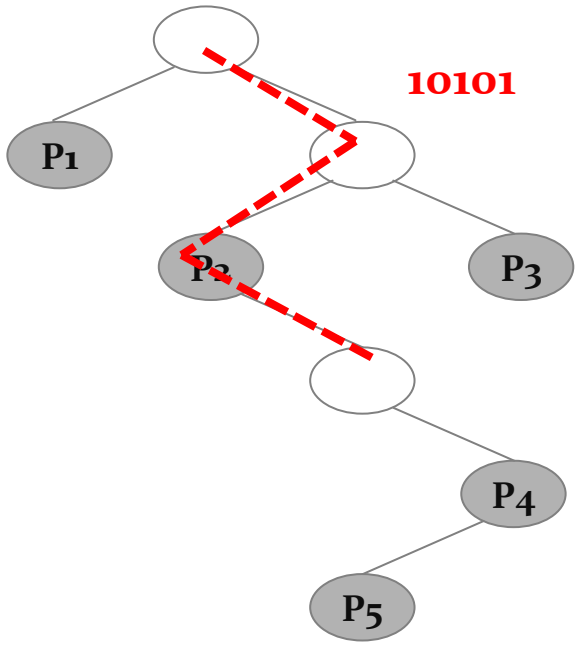
Address Lookup Scheme 1

- Trie-Based Scheme: Standard Trie Structure
  - Forwarding table Example
    - Each entry consists of a route prefix and the corresponding next-hop information
    - This example is used to illustrate most of the schemes

Name	Prefix	Next Hop
P1	0*	H1
P2	10*	H2
P3	11*	H3
P4	1011*	H4
P5	10110*	H5

Address Lookup Scheme 1

- Trie-Based Scheme: Standard Trie Structure
  - 1-b trie structure
    - Node structure
      - Next-hop pointer (Next-hop-ptr): Next hop information
      - Left pointer (Left-ptr) / Right pointer (Right-ptr): Link to subtrie
    - Performance
      - Worst case time complexity for lookup O(W)
      - Storage requirement O(NW)
      - Time for update O(W)
    - Simple and extensible to a wider field for its low storage requirement
    - Worst case lookup time can be large
      - e.g. W=128 for IPv6 address



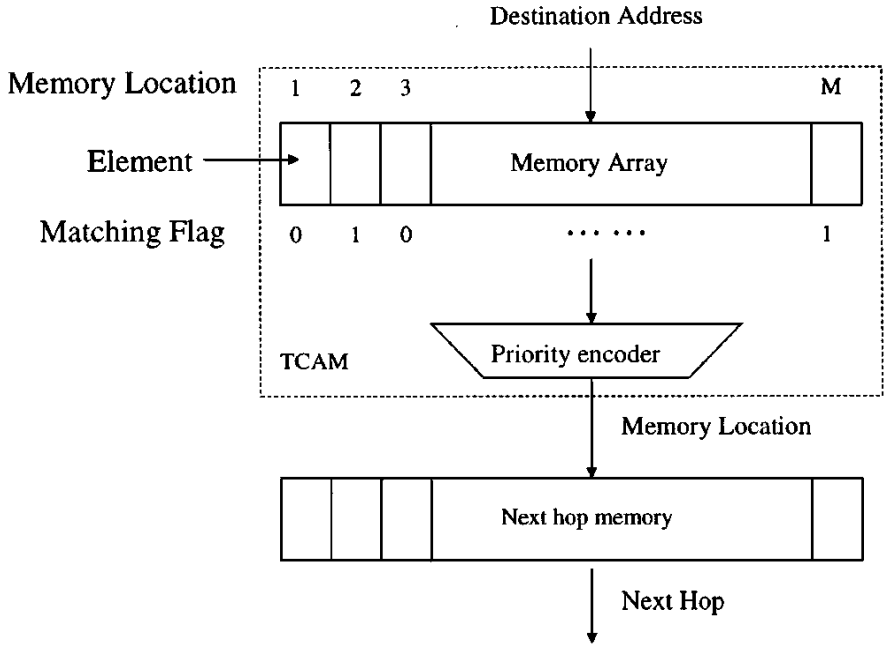
Next-hop-ptr (if prefix)	
Left ptr	Right ptr

Address Lookup Scheme 2

- Hardware-Based Scheme: Ternary CAM
  - CAM: Specialized memory that performs parallel comparison internally
  - Ternary CAM: TCAM
    - Store W-bit field as (val, mask) pair
    - If multiple match is found
      - Priority encoder is used to select highest priority (longest match)
    - Forwarding table stored in decreasing order of prefix length

Address Lookup Scheme 2

- Hardware-Based Scheme: Ternary CAM
  - Commercially available TCAM integrate 9M bits into a single chip



Address Lookup Scheme 2

- Hardware-Based Scheme: Ternary CAM
  - Commonly used in commercial routers
  - Provides virtually-instantaneous hardware search and update
  - Cost 4-5 times as much as a conventional RAM of the same size
  - Less memory capacity than RAMs of the same size

SRAM	$30	8Mb @ 200 MHz	4-6 tr/bit
TCAM	$70	2Mb @ 100 MHz	10-12 tr/bit

Network Processors

Background

Routers

: Address Lookup

Network Intrusion Detection Systems (NIDS)

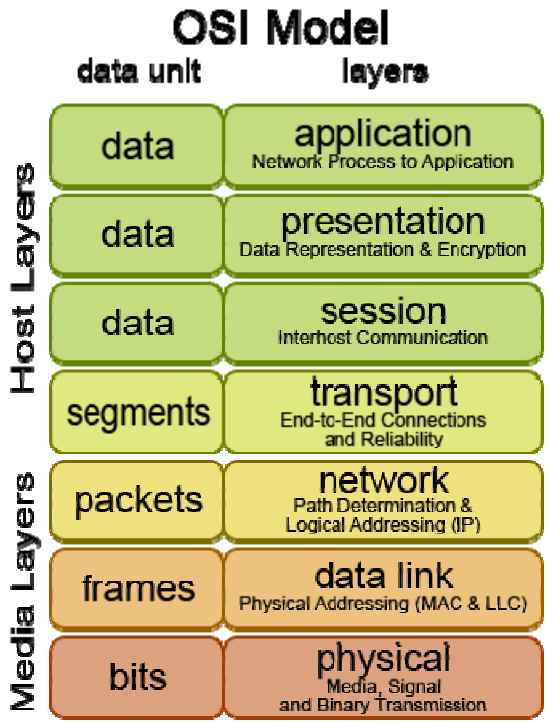
: Deep Packet Inspection (DPI)

NIDS

- Network Intrusion Detection System: NIDS
  - Intrusion detection system that tries to **detect malicious activity** such as denial of service (DoS) attacks, port scans or even attempts to crack into computers by monitoring network traffic.
    - Log intrusion information or save raw packets.
    - Send an alert to an administrator using email or another method.
    - Interfere with the attack.
  - Supports **deep packet inspection**

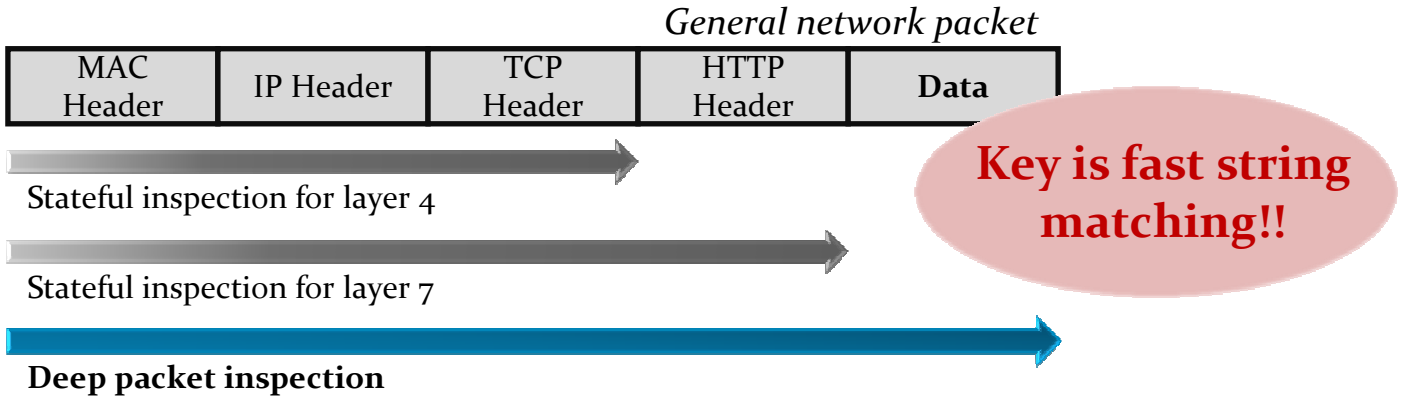
Network Layers

- OSI Model
  - an abstract description for layered communications and computer network protocol design.



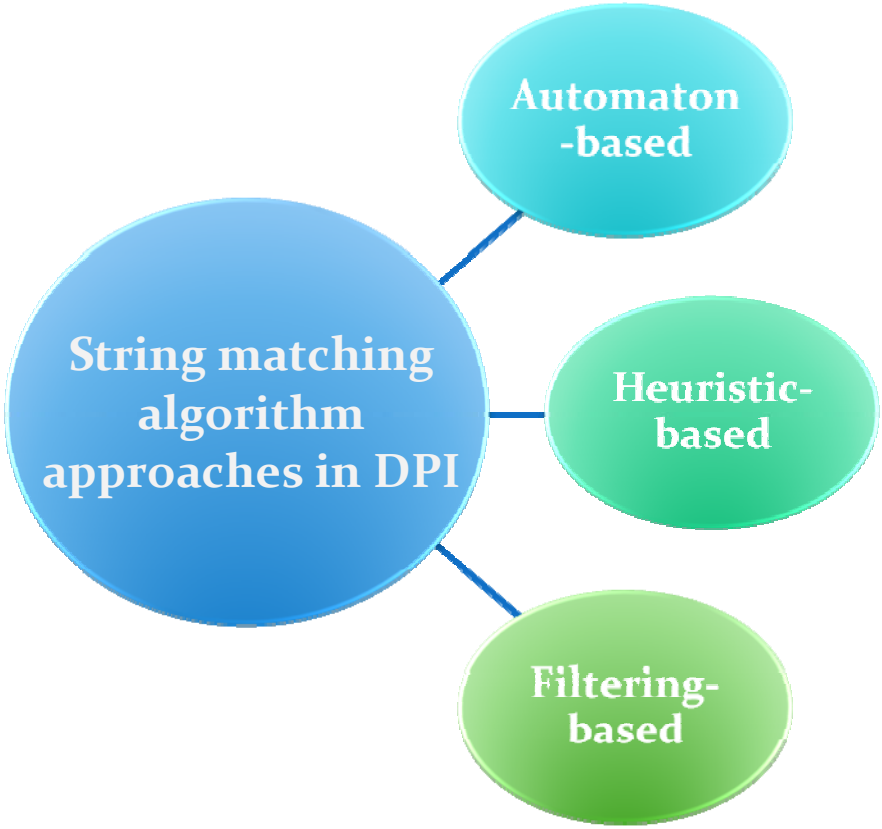
Deep Packet Inspection

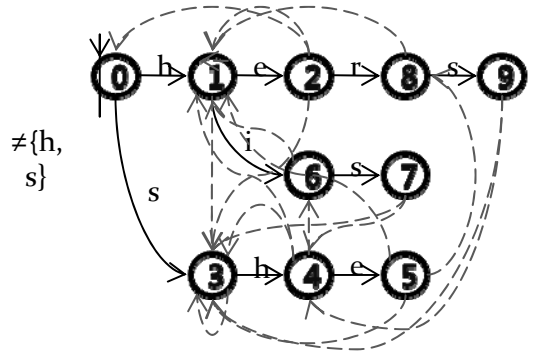
- Deep packet inspection: DPI
  - A form of computer network packet filtering
  - Examining the data and/or header part of a packet as it passes an inspection point
  - Searching for protocol non-compliance, viruses, spam, intrusions or predefined criteria to decide if the packet can pass or if it needs to be routed to a different destination, or for the purpose of collecting statistical information.

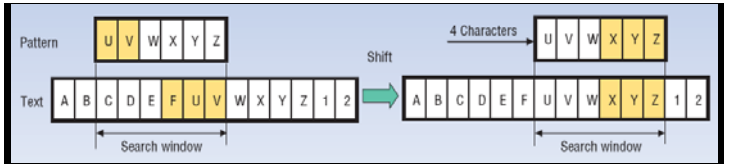


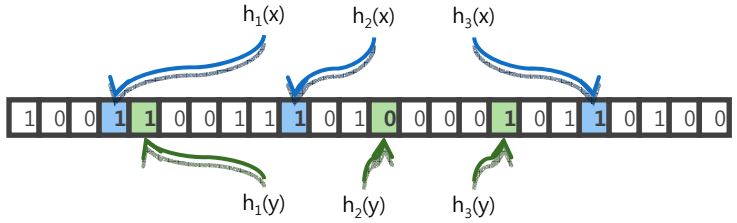
Deep Packet Inspection

- String Matching Algorithms









String Matching

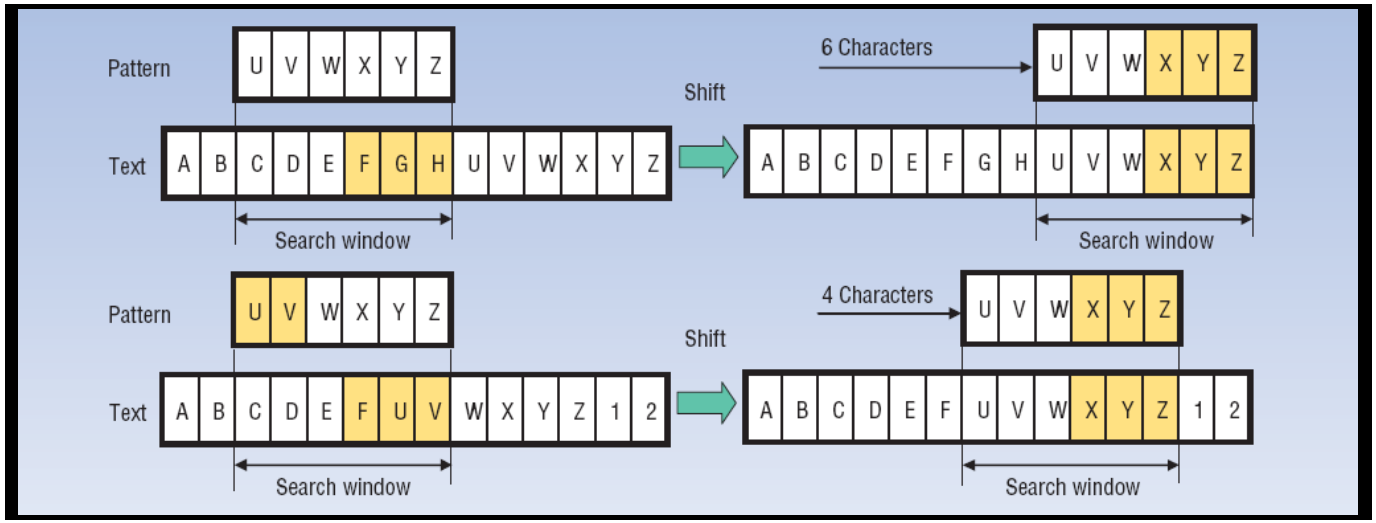
- Automaton-Based Approach

	Deterministic Finite Automaton (DFA)	Nondeterministic Finite Automation (NFA)
Feature	Only one transition to a next state for each pair of state and input symbol	Several possible next states for each pair of state and input symbol
Strength	Lower time complexity	Less space for pattern storage

  - Reducing data-structure space
    - Reducing sparse transition table size
    - Reducing number of transitions
  - Inspecting multiple characters simultaneously

String Matching

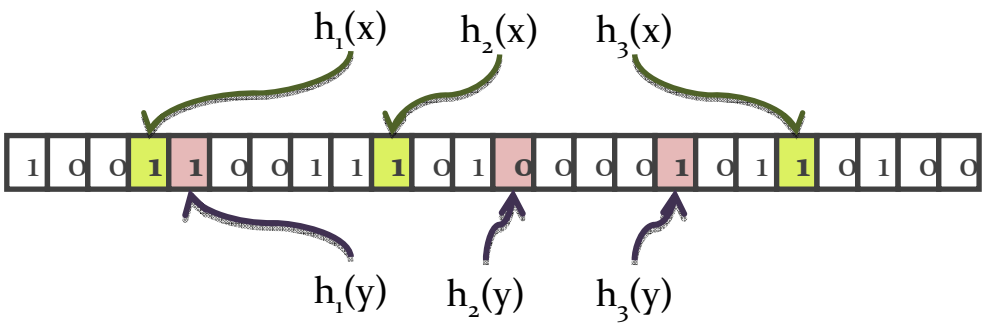
- Heuristic-Based Approach
  - Accelerating by skipping characters not in a match
  - Not preferred due to its vulnerability to algorithmic attacks
  - Needed to consider size of pattern set, block distribution, cache locality, and verification frequency



String Matching

- Filtering-Based Approach
  - Text filtering with the Bloom filter
  - Reducing false-positive rate
  - Handling parallel queries
    - Handling Sequentially with a set of hash functions
  - Handling queries in different lengths
    - Breaking a long pattern into short patterns

Pattern x, y
Three hash functions h_1, h_2, h_3



- *All the hash functions of pattern x are set in 1. Pattern x might be in the pattern set.*
- *Not all the hash functions of pattern y are set in 1. Pattern y is not in the pattern set.*

String Matching

- Pros and Cons

Approach	Pros	Cons
Automaton-based	• Deterministic linear execution time • Direct support of regular expressions	• Might consume much memory w/o compressing data structure
Heuristic-based	• Can skip characters not in a match • Sublinear execution time on average	• Might suffer from algorithmic attacks in the worst case
Filtering-based	• Memory efficient in the bit vectors	• Might suffer from algorithmic attacks in the worst case

Hardware DPI Engine

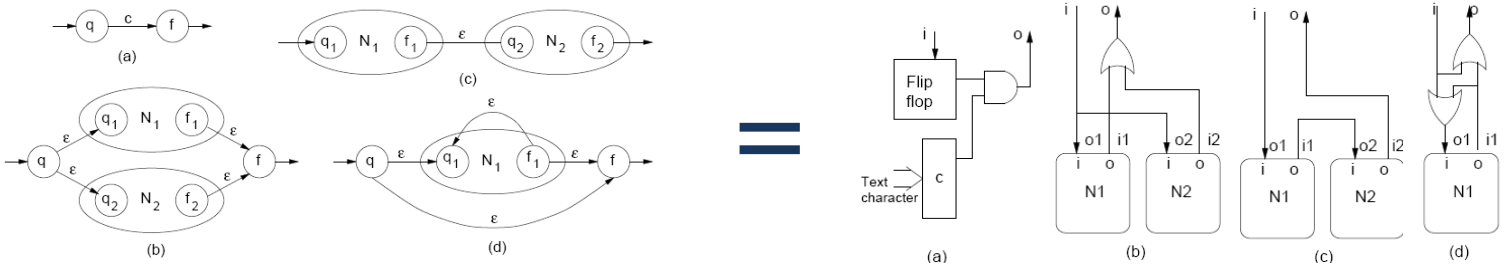
- Features & Requirements
  - Support high level of parallelism
  - Easy configuration
  - High efficiency

- Implementation
  - Finite Automation(FA) based
    - Non-deterministic Finite Automation(NFA) based
    - Aho-Corasick Algorithm(DFA) based
  - Bloom Filter based
    - Use the set of the hash tables

Hardware DPI Engine 1

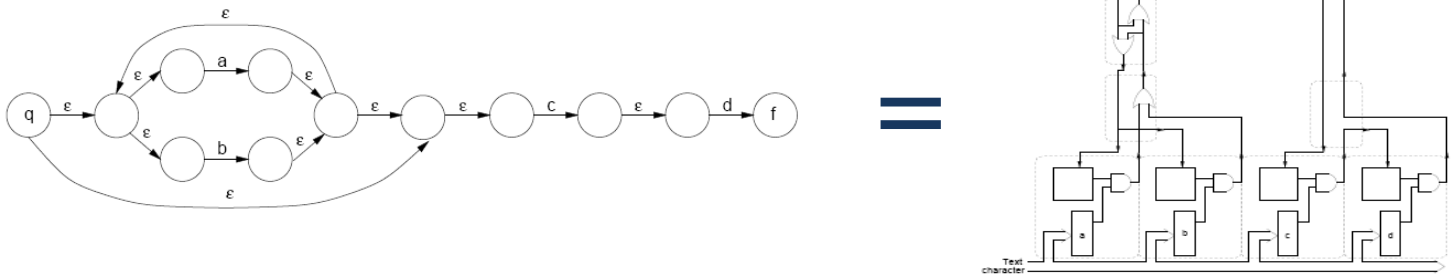
- NFA-Based DPI Engine

Basic Block Construction



a) single character b)r1|r2 c)r1r2 d)r1*

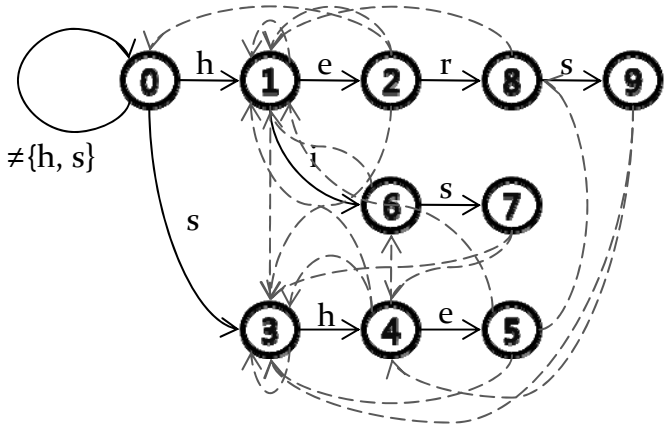
Example: NFA for ((a|b)*)(cd)



Computer Systems & Reliable SoC Lab.
Yonsei University

Hardware DPI Engine 2

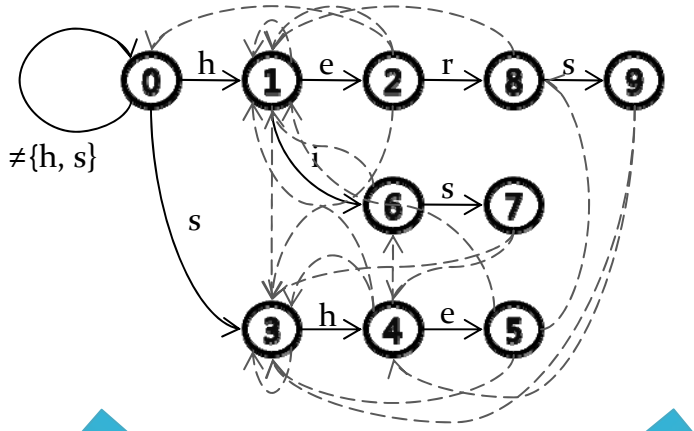
- AC Algorithm-Based DPI Engine
  - DFA generation based on AC algorithm
    - 1) building up a tree of all the strings
    - 2) inserting failure edges
  - Hardware implementation Issues
    - A large number of possible out edges that are directed out of each and every node
    - The serial nature of the state machines



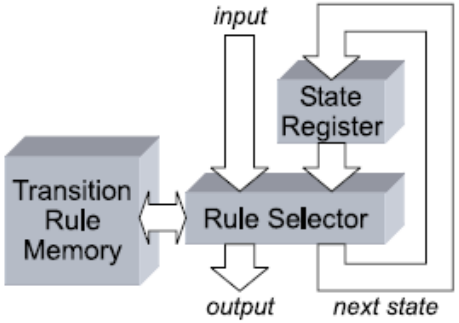
i	output(i)
2	{he}
5	{she, he}
7	{his}
9	{hers}

Hardware DPI Engine 2

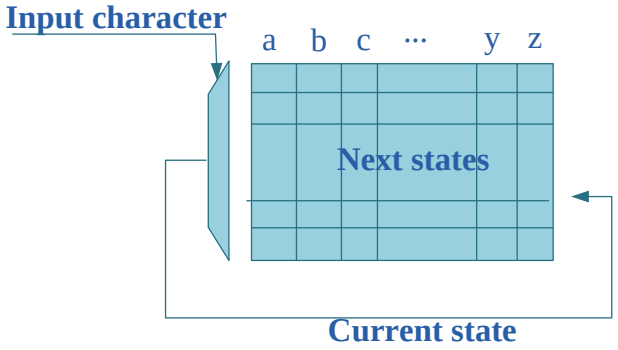
- AC Algorithm-Based DPI Engine



Storing transition information



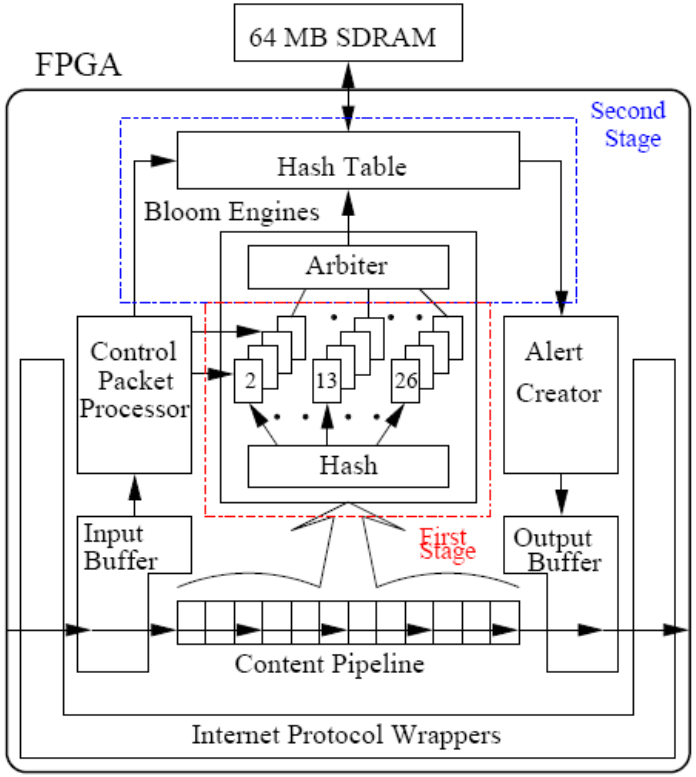
Storing states information



Computer Systems & Reliable SoC Lab.
Yonsei University

Hardware DPI Engine 3

- Bloom Filter-Based DPI Engine
  - Bloom Engines
    - To search for multiple signature widths, multiple bloom engines are instantiated that each operate in parallel
  - Hash Table(in the second stage)
    - To eliminate false positives generated from bloom filters



Network on Chip (NoC)

- **Background**
  - On-Chip Bus vs. NoC
- NoC Design
- NoC Implementations

Background

- Evolutions of system Architecture

Application Convergence

Parallel processing on a chip using many small processors

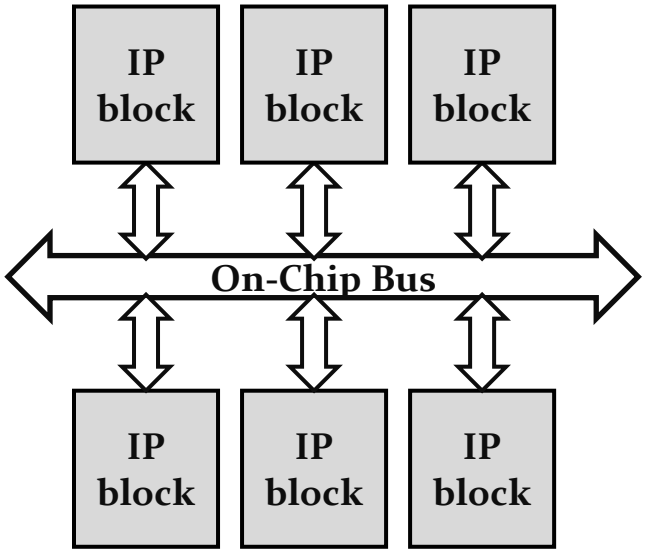
Gates cost relatively less than wires.

IP reuse to reduce time-to-market

Driving evolutions of On-Chip Communication

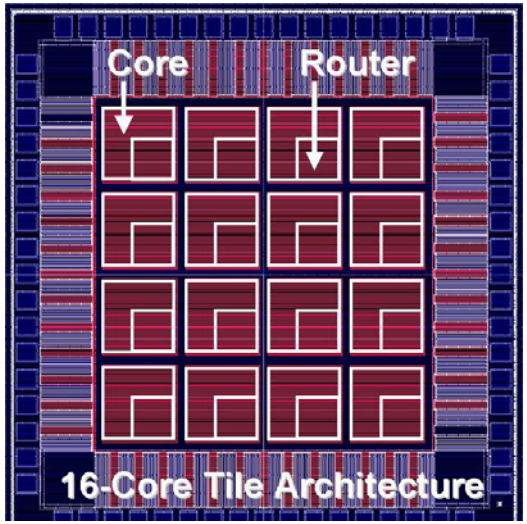
Background

- On-Chip Buses
  - Traditional way to implement the communication subsystem of System-on-a-Chip (SoC)
  - Limitations
    - Lack of scalability
    - Difficult to reuse
    - High design cost



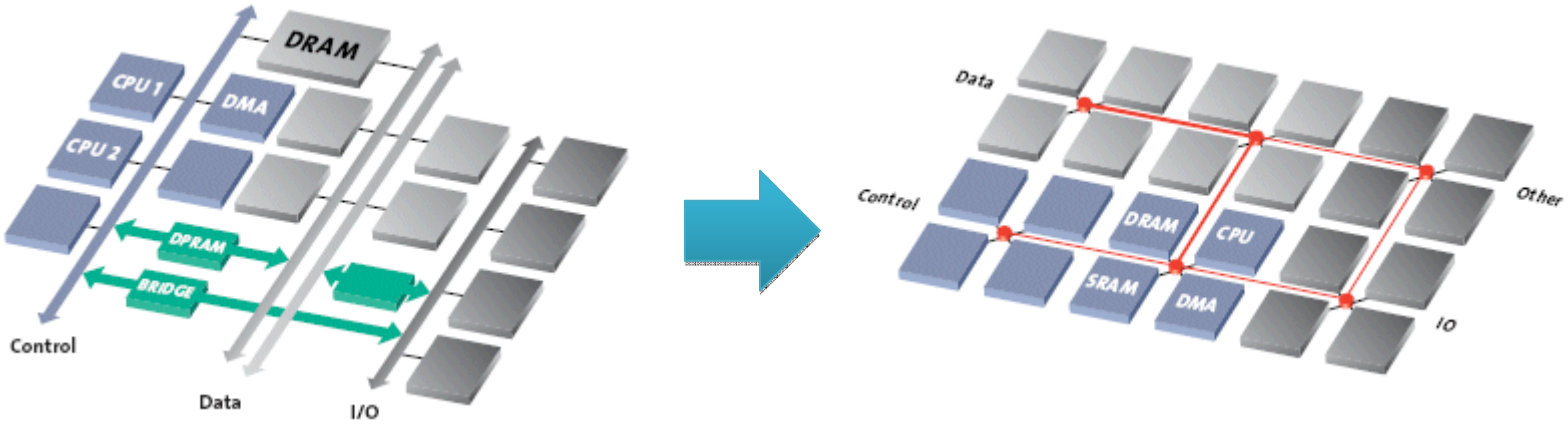
Network on Chip

- Network on Chip: NoC
  - New approach to design the communication subsystem of SoC
  - Brings networking theories and systematic networking methods to on-chip communication
    - Layered and scalable architecture
    - Flexible and user-defined network topology.



On-Chip Bus vs. NoC

- On-Chip Bus vs. NoC
  - Synchronous bus limitations lead to system segmentation and tiered or layered bus architectures.



Traditional synchronous bus **Homogeneous NoC**

On-Chip Bus vs. NoC

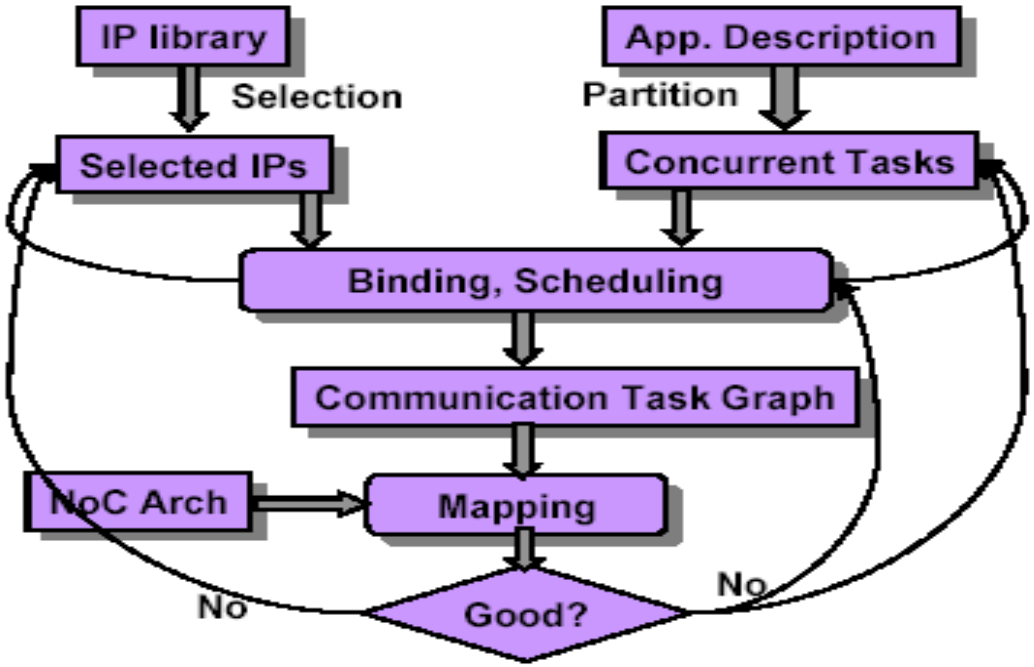
Bus Pros & Cons			Network Pros & Cons
Every unit attached adds parasitic capacitance, therefore electrical performance degrades with growth.	-	+	Only point-to-point one-way wires are used, for all network sizes.
Bus timing is difficult in a deep submicron process.	-	+	Network wires can be pipelined because the network protocol is globally asynchronous.
Bus testability is problematic and slow.	-	+	Dedicated BIST is fast and complete.
Bus arbiter delay grows with the number of masters. The arbiter is also instance-specific.	-	+	Routing decisions are distributed and the same router is reinstanciated, for all network sizes.
Bandwidth is limited and shared by all units attached.	-	+	Aggregated bandwidth scales with the network size.
Bus latency is zero once arbiter has granted control.	+	-	Internal network contention causes a small latency.
The silicon cost of a bus is near zero.	+	-	The network has a significant silicon area.
Any bus is almost directly compatible with most available IPs, including software running on CPUs.	+	-	Bus-oriented IPs need smart wrappers. Software needs clean synchronization in multiprocessor systems.
The concepts are simple and well understood.	+	-	System designers need *reeducation* for new concepts.

Network on Chip (NoC)

- Background
  - On-Chip Bus vs. NoC
- **NoC Design**
- NoC Implementations

NoC-Based System Design

- NoC Design Flow



NoC-Based System Design

1. Backbone Design
   - A development platform for all NoC-based design
   - It encapsulates topological and communication issues such as channels, switches, and network interfaces
   - Its design focus is the network communication resources
     - e.g. switches and interfaces, and NoC system services and performance of different region topologies
   - Definition of region requires that potential applications are analyzed and modeled

NoC-Based System Design

2. Platform Design: creates a computation platform for an intended application area.
   - Main activities: scaling of the network, definition of regions, design of the resource nodes, and definition of the system control.
   - First, understand the target systems' functionality thoroughly.
   - Do not use exact applications as a starting point for architecture requirement definition due to the platform nature.
   - Use of optimized virtual components and knowledge of application-area requirements
   - Characterizes application area domain and architecture, and estimates system quality

NoC-Based System Design

3. System Design
   - Application mapping: application functionalities are mapped to the resources.
     - Must support both dynamic and static mapping
       - Main problems: resource allocation, optimization of network usage, and verification of performance and correctness
       - Several modeling languages should be supported.
   - Main tasks: what to put into the NoC as resources, how to map functionality into those resources (resource selection), and how to validate the decisions.
     - Mappability of algorithms and architectures
     - Analysis of network behavior is a critical part
   - The development and verification environments provides a virtual machine and development environment for software development, and tools for hardware design

Backbone Design

- Resource design problem
  - What is needed inside resources?
  - Internal computation type and internal communication?

- Region definition problem
  - What kind of regions are needed?
  - What kind of interfaces between regions?
  - What are the capacity requirements for the regions?

Platform Design

- Scaling problem
  - How big NOC is needed?
  - What are the application area requirements?

System Design

- Application mapping flow problem
  - What kind of languages, models and tools must be supported?
  - How to validate and test the final products?

NoC Application Development

- Mapping problem
  - How to partition applications for NOC resources?
  - How to allocate functionality effectively?
  - Is the performance adequate?
  - Is the resource usage in balance?
- Optimization problem
  - How to perform global optimization of heterogeneous applications?
  - How to define right optimization targets?
  - How to utilize application/resource type specific tools?

NoC Application Development

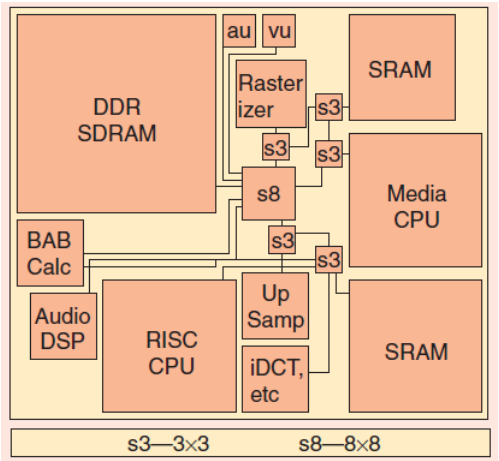
- Validation problem
  - Are the constraints met?
  - Are there communication bottlenecks or power consumption hot spots?
  - How to simulate 10000 GIPS system?
  - How to test all applications?

Network on Chip (NoC)

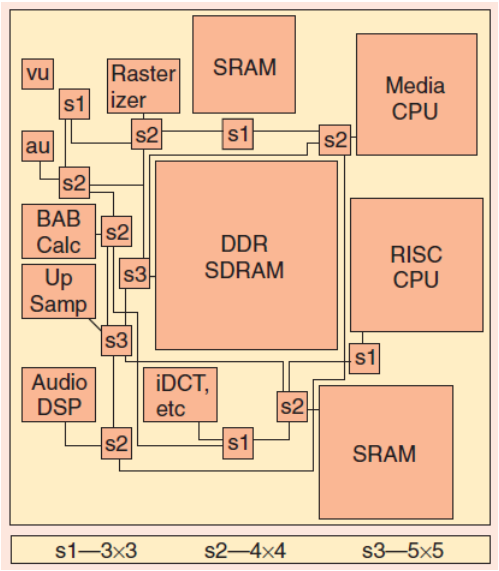
- Background
  - On-Chip Bus vs. NoC
- NoC Design
- **NoC Implementations**

NoC Implementation 1

- *Xpipes*
  - Targeting high performance and reliable communication for on-chip multi-processors
  - Library of soft macros (switches, network interfaces and links) that are design-time composable and tunable so that domainspecific heterogeneous architectures



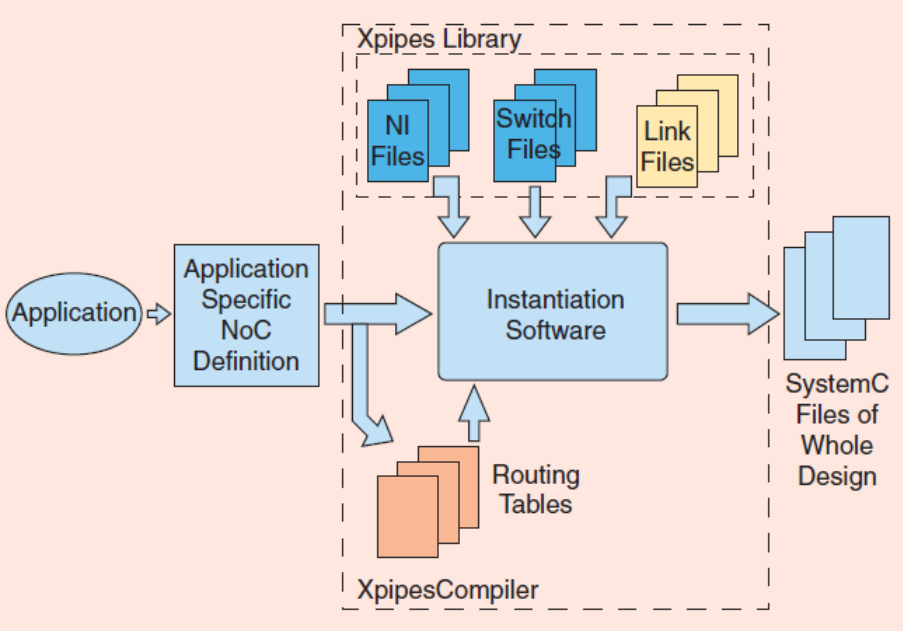
Application-specific NoC



Mesh NoC

NoC Implementation 1

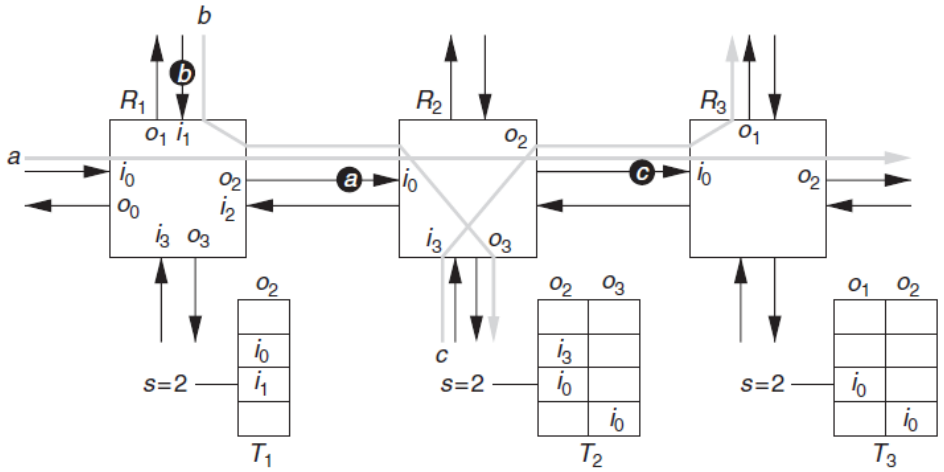
- *Xpipes*
  - *XpipesCompiler*: tool which automatically instantiates a customized NoC from the library of soft network components



NoC design flow using *XpipesCompiler*

NoC Implementation 2

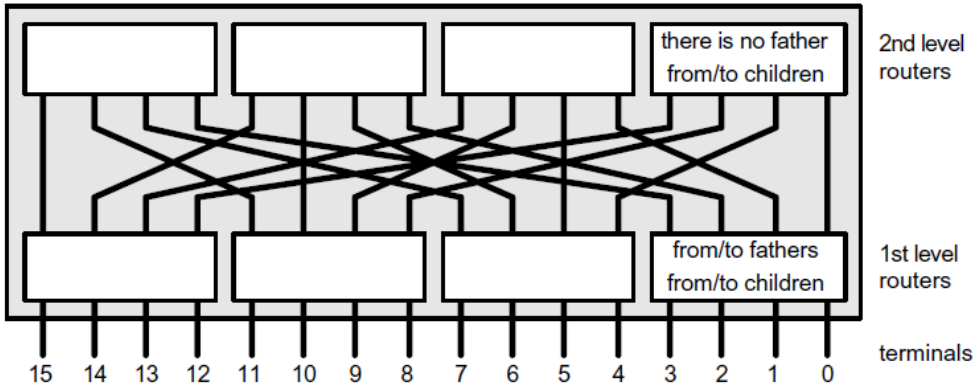
- *Ethereal*
  - Developed by Phillips
  - Aims at achieving composability and predictability in system design and eliminating uncertainties in interconnects, by providing guaranteed throughput and latency services



Contention-free routers

NoC Implementation 3

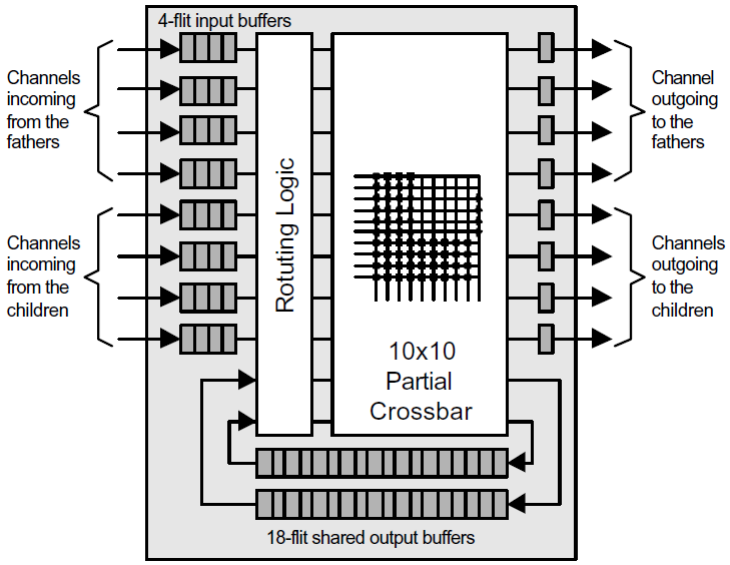
- *SPIN*
  - Fat-tree topology
    - tree structure with routers on the nodes and terminals on the leaves, except that every node has replicated fathers
    - non-blocking network with a performance that scales gracefully with the system size



16-terminal *SPIN* network

NoC Implementation 3

- *SPIN*
  - Basic building block of *SPIN* network: *RSPIN* router.
    - Partial 10×10 crossbar, which implements only the connections allowed by the routing: all the packets flowing down the tree can be forwarded to children and only such packets can use the output buffers when the required output channel is busy.



RSPIN