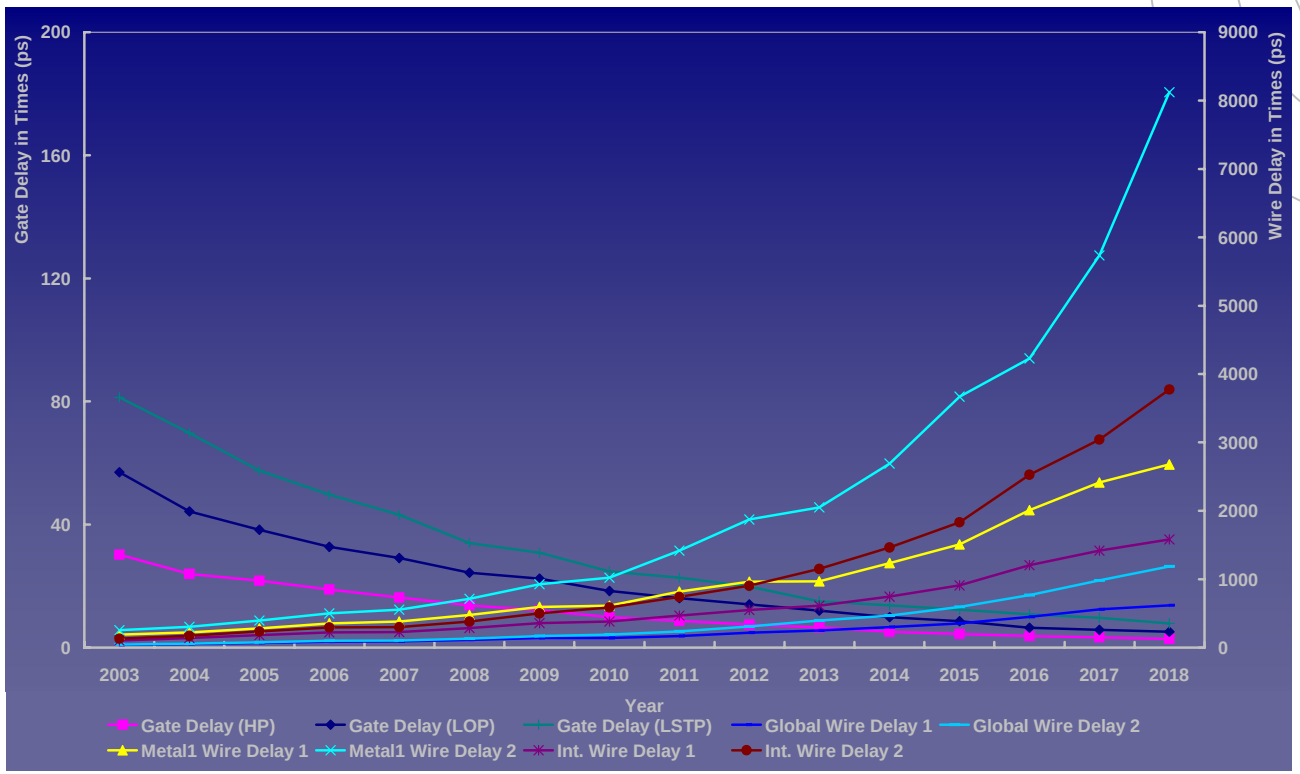
A Generic Network Interface Architecture for a Networked Processor Array (NePA)

Seung Eun Lee, Jun Ho Bahn,
Yoon Seok Yang, and Nader Bagherzadeh
EECS @ University of California, Irvine

Outline

- Introduction
- Network-on-Chip
- Related Works
- Generic Network Interface
  - Related Works
  - Networked Processor Array (NePA) Architecture
  - Generic Network Interface
  - Programming Sequence
  - Modular Wrapper for a Slave IP Core
  - Case Studies: Memory/ Turbo Decoder IP Cores
- Summary

Introduction



- In 2018, the interconnection delay is estimated to be 1000 times greater than gate delay [ITRS]
- The interconnection network among multiple IPs becomes another challenging issue in System-on-Chip (SoC) design

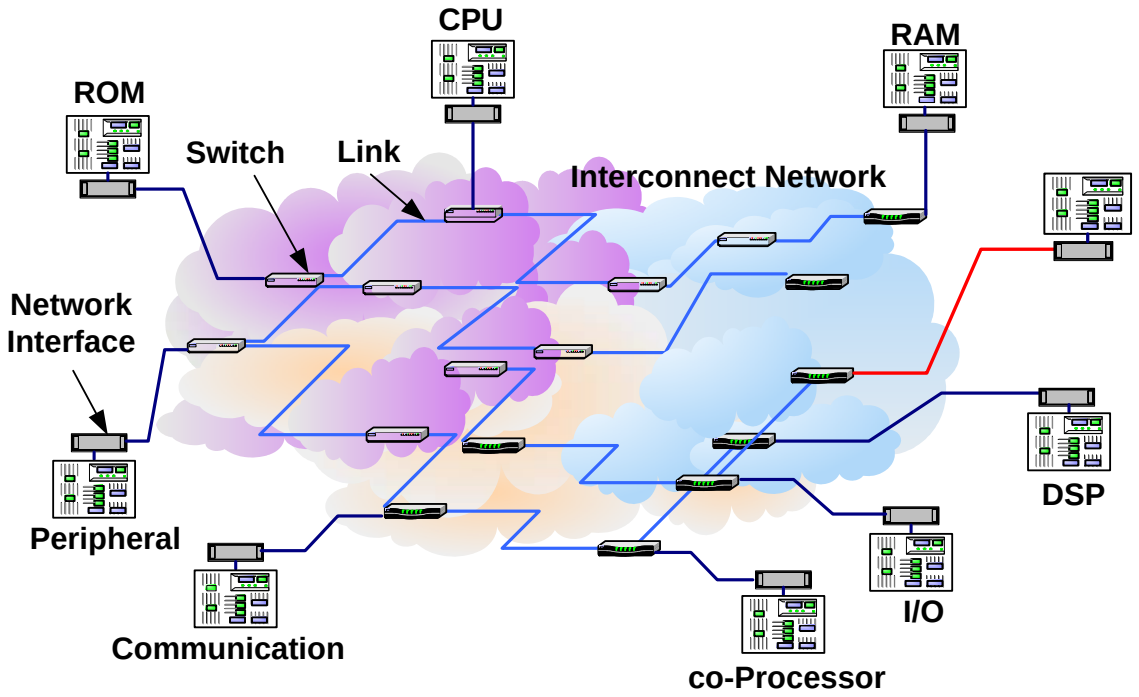
from ITRS 2004 Report

Introduction (cont'd)

- Current Trends in VLSI Technology
  - Requirements
    - Computation intensive applications
    - Highly integrated + low power
  - Increasing # of computing resources in SoC
    - CPUs, DSPs, ASPs
  - System platforms
    - MPSoC (Multi Processor System-on-Chip) or CMP (Chip Multi Processor)
    - Homogeneous/Heterogeneous processors
    - Similarity in a small scale distributed computer system

- Interconnection?

Network-on-Chip Interconnection



The use of switching based technology

- Have been extensively used for computer network
- Communication between IPs can be packet based

The key efficiency of NoC

- Communication resources are SHARED !

Network-on-Chip (cont'd)

- Network-on-Chip (NoC) Architecture
  - Network-like interconnection
    - Insertion of routers
    - Shortened wiring requirement
    - Alleviating scalability and freedom from the limitation of complex wiring
  - Difference from computer network technology (Internet TCP/IP)
    - Simple and light-weight modification
    - low power requirement for mobile applications
    - Performance and cost
- Different interface specification of integrated components raise a considerable difficulty for adopting NoC techniques

Generic Network Interface

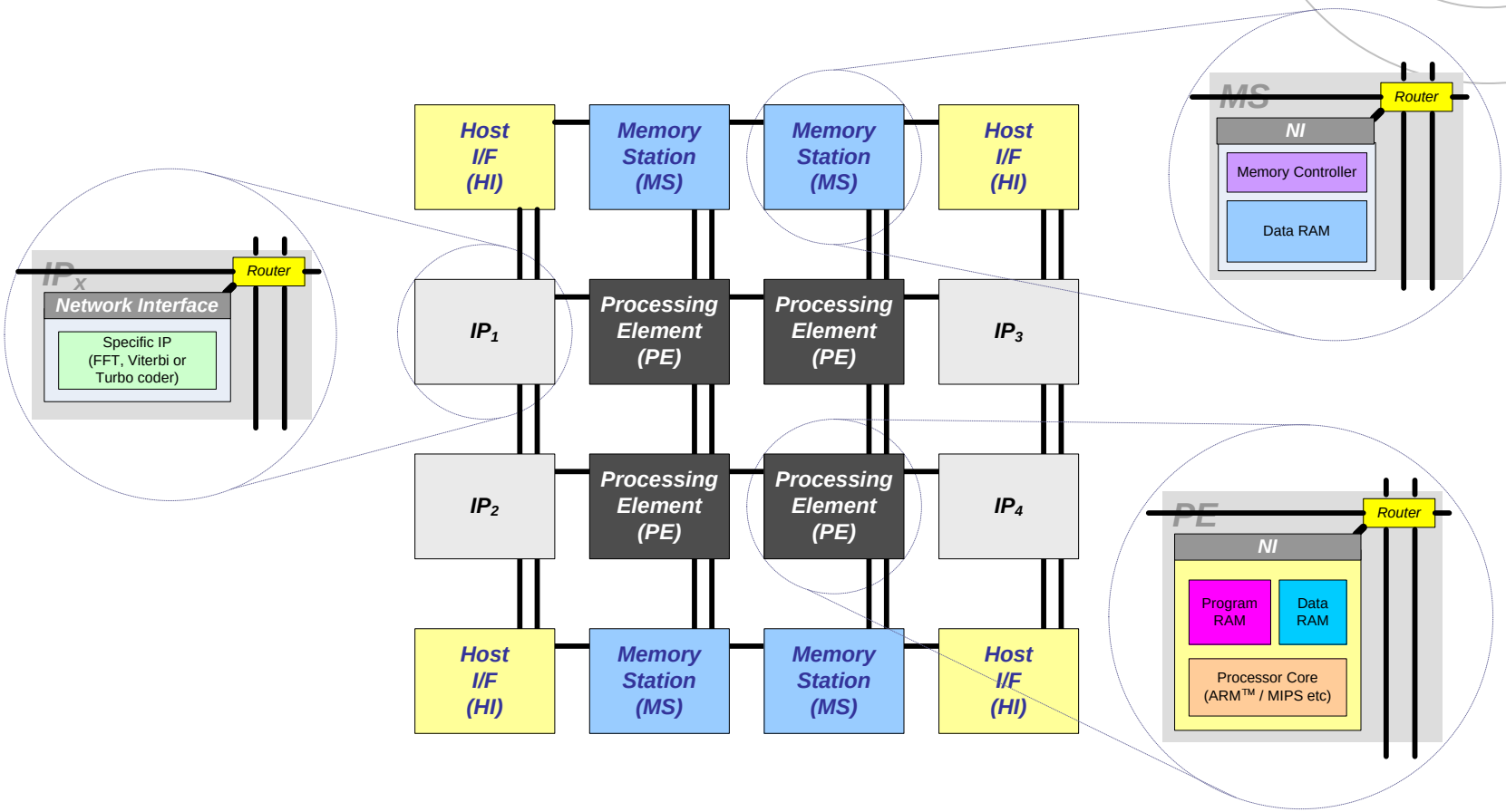
- The reuse of IP cores in plug-and-play manner can be achieved by using a generic network interface (NI)
  - Reduce design time of new system
  - Translate packet-based communication into a higher level protocol
  - Decouple computation from communication
  - Hide the implementation details of interconnection

Related Works

- Different Packetization strategy
  - Software library, on-core and off-core implementation
  - A hardware wrapper implementation has the lowest area overhead and latency
- NI for standard Interface such as OCP, DTL and AXI
  - Improve reuse of IP cores
  - Performance is penalized because of increasing latency
- Generic architecture and automatic generation of interface
  - Existing researches limit the embedded IP cores to CPU (ARM7 and MC68000)
- The designs of wrapper for application specific cores still lack generic aspects

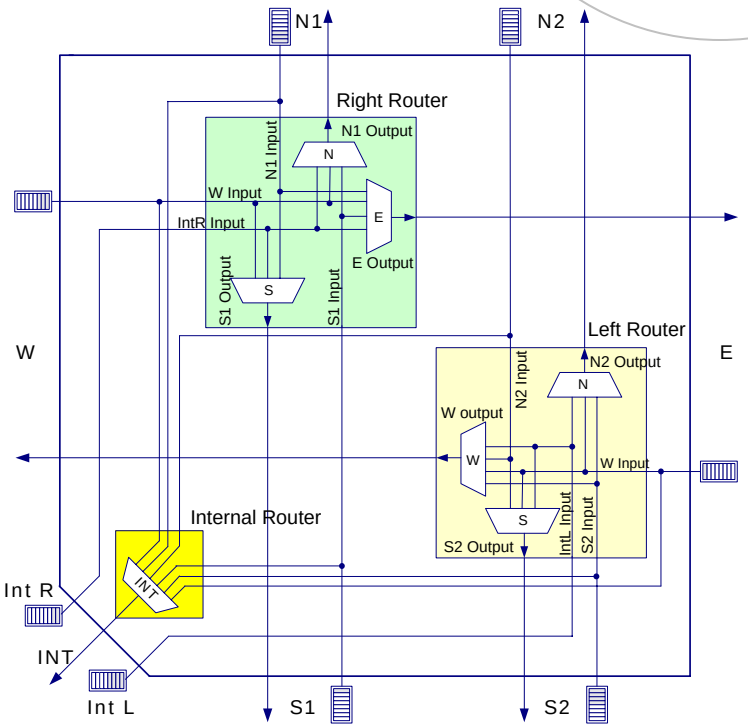
NePA Architecture

- System Platform

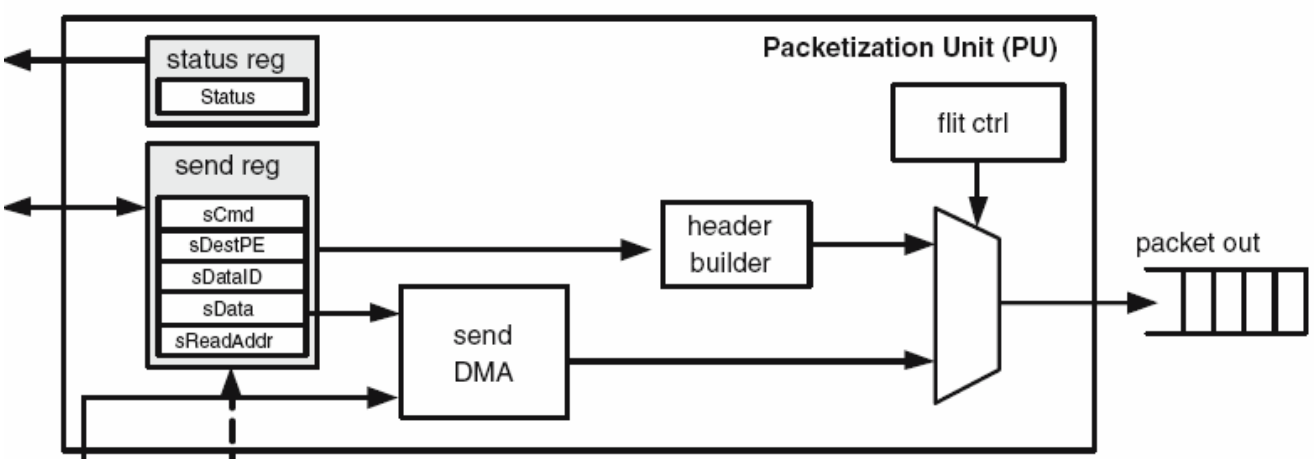


NePA Architecture:
High-Performance Router Architecture

- Interconnect throughout FIFO between neighboring PEs
  - Simple Interconnect Wiring
- Minimal (shortest) adaptive routing
  - Livelock-free
- Point-to-point single or block transfer
- Two disjoint sub-networks for the west-to-east and east-to-west traffics
  - Network avoids a cyclic dependency
  - Resulting in deadlock-freedom
- Prioritized packet delivery

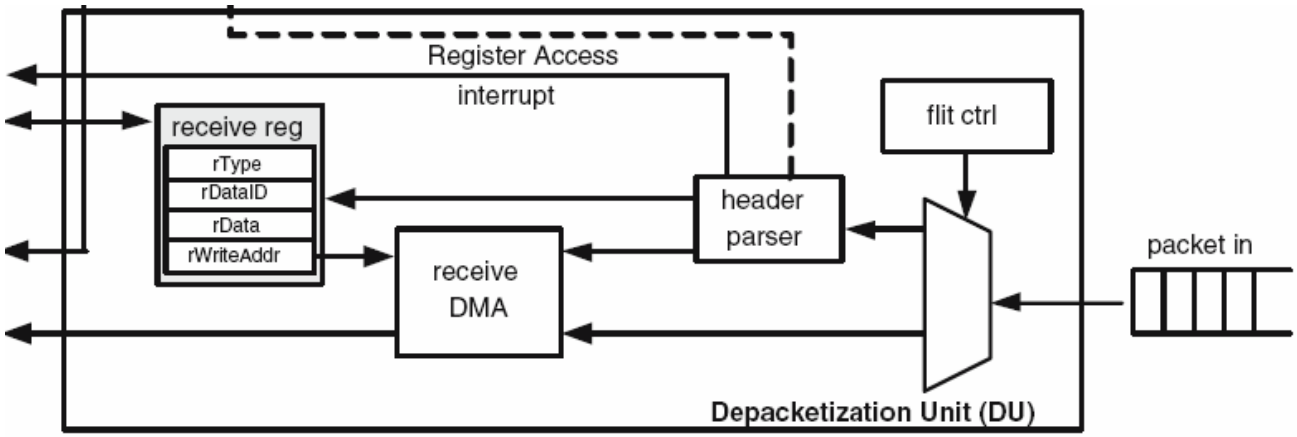


Network Interface Prototype:
Packetization Unit



- Build the packet header and converts the data into flits
  - Header builder: form the head flit based on the information provided by registers
  - DMA controller: generate control over the address and read signal for the internal memory automatically
  - Flit Controller: wrap up the head flit and body flits into a packet

Network Interface Prototype:
Depacketization Unit



- Receive data from interconnection network
  - Flit Controller: select head flit from a packet and pass it to the header parser
  - Header parser: extract control information from the head flit and assert an interrupt signal to the OpenRISC core
  - DMA controller: writes the body flit data into the internal memory automatically

Network Interface Prototype:
Programming Sequence

- Sending SINGLE Packet
  - All required parameters are set to the associated registers
  - Writing command register generate a complete packet
- Sending BLOCK packet
  - *sDataReg* represents the number of data
  - *sReadAddrReg* indicates the start address of data in memory
- Receiving SINGLE/BLOCK packets
  - Parameters are accessed by interrupt service routine
  - Accessing *rDataReg* completes the procedures for current packet
  - For BLOCK packet, OpenRISC sets the corresponding write address (*wWriteAddrReg*) for internal memory access

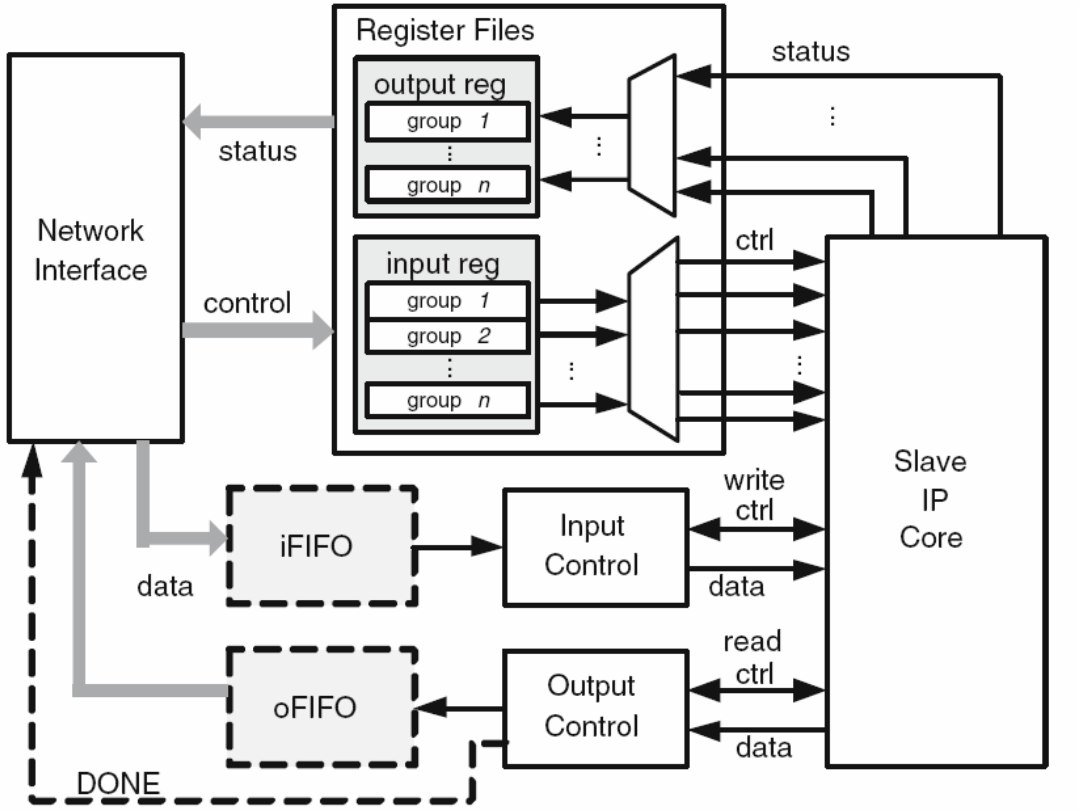
Generic Network Interface
Modification of Packet for NI access

- A slave IP core is not able to write registers in a current NI
- These registers are accessed by other cores using the network

Type	$(\Delta x, \Delta y)$	*Tag*	*Data*
NI access		**Opcode**	**Operand**
SINGLE	0x21	write (*sDestPEReg*)	0x01

  - Opcode and Operand of an instruction are located at Tag and Data field in the SINGLE packet
  - *Type* field indicates that packet contains an instruction for NI
  - Instruction decoder in the header parser fetches opcode and operand from a packet
  - Update internal registers

Generic Network Interface
Modular Wrapper for a slave IP core



- Un-buffered Mode: data is exchanged in data stream without intermediate buffer
- Buffered Mode: data is saved in the intermediate buffer temporarily

Generic Network Interface Modular Wrapper for a Memory

- Maintain data and shared among a number of PEs.
- Assume synchronous SRAM model

- Wrapper design
  - Core type is slave IP
  - There is no control signals for initialization or status monitoring
  - Data interface is realized in the un-buffered mode removing the FIFOs between NI and memory
- Programming sequence
  - The base address is set to the desired value using SINGLE packet
  - Sending BLOCK packet stores data into memory
  - Read operation is done by sending SINGLE packet

Generic Network Interface Modular Wrapper for a Turbo Decoder

- Stand-alone turbo decoder operating block by block process
- Wrapper design
  - Core type is slave IP
  - There are six signals that are used for initialization and mode selection
  - Data interface adopts buffered mode, inserting FIFOs between NI and the core
- Programming sequence
  - Before starting turbo decoding, it is initialized by sending packet which accesses the input control signals
  - Data is sent to the core using BLOCK packet
  - When decoding of one block is completed, NI start to send a packet to the other node automatically

Summary

- Introduced Networked Processor Array (NePA)
- Proposed network interface architecture for OpenRISC core
- Classified the possible IP cores for processing elements
- Proposed a modular wrapper for an embedded IP cores
  - Allocation table was used for the configuration of the modular wrapper
  - Programming model was presented
  - Case studies in memory and turbo decoder cores demonstrated feasibility and efficiency of the proposal