Rx Stack Accelerator for 10 GbE Integrated NIC

F. Abel, C. Hagleitner
IBM Research – Zurich
Switzerland

F. Verplanken
IBM Systems & Technology Group
La Gaude – France
Discrete vs Integrated Network Interface Controller

- **Discrete NIC (dNIC)**
  - Peripheral ASIC device
  - Marketed as:
    - Ethernet Controller
    - Converged Controller

- **Integrated NIC (iNIC)**
  - Sun Niagara 2
  - Freescale → QorIQ™ family
  - IBM → PowerEN™

![Network Interface Card (NIC)](image)

- Converged Network Adapter (CNA)
- LAN On Motherboard (LOM)

![Diagram](image)

- **Higher performance, lower latency**
- **Lower power consumption**
- **Significant cost reduction**
- **Alter the general-purpose nature of the computer complex**

410 mm² (1.43 billion transistors)
Outline

- Hardware context of this work
  - PowerEN™ / Host Ethernet Adapter

- Functional requirements

- Architecture of the Rx Stack Accelerator
  - Data path
  - Packet parser
  - Packet handler

- Results
  - Implementation
  - Performance

- Summary and conclusions
Hardware Context
Overview of the Host Ethernet Adapter (HEA)

- 4×10 GbE state-of-art Ethernet controller featuring:
  - I/O virtualization support
    - Through 128 queue pairs
    - Internal layer-2 switch for partition-to-partition data traffic
  - Flexible queue selection and scheduling assist
  - Rx and Tx protocol acceleration
  - Low-latency through direct processor bus attachment and cache injection
  - Multi-core scaling,
  - Interrupt and receive coalescing assist,
  - 9 KB Jumbo frame support,
  - Memory address protection
  - . . .

- Equivalent functionality and performance levels as modern discrete NICs

- Small footprint (13 mm²) and low power (2.6 W)
RXACC
Functional Requirements
Target Domain Requirements

- **PowerEN™ targets network-facing applications**
  - Must cope with a large spectrum of protocols (13+), traffic characteristics and policies
    - E.g. routers, firewalls, intrusion-prevention systems and network analytics
  - Must be able to adapt to changes
    - Handle other existing or emerging protocols
  - Virtualized I/Os → Must sustain 16 Gb/s (internal layer-2 switch)

- **Ethernet: A trucking service for application data**
  - Protocol layered architecture (i.e., encapsulation) → 2000+ permutations
Distribution of the Protocol Stacks

- DIX, 1153
- VLAN, 1183
- IPv6, 1263
- MPLS2, 1510
- L4, 2118
- IP4, 940
- IP4t, 197
- IP6, 1263
- IP6t, 186
- L4, 2118
Offloaded Service Requirements

- **WHY**
  - TCP Rx+Tx processing → ~3000 instructions (assuming zero-copy and checksum offload)
  - 10 GbE → Occurrence of 64 B packets → 67.2 ns (14.8 Mfps)
  - A generally accepted rule of thumb → 1 GHz / 1 Gb/s

- **WHAT** (business as usual)
  - 'per-byte' operations
    - check-summing
  - 'per-packet' operations
    - header processing: VLAN, MAC, flow identification, QoS determination, discard, errors, traffic steering

- **HOW** (different)
  - Flexible way → Programmable parsing and processing (rule-based)
  - Meta-data descriptors → Save 300-400 instructions per frame
    - E.g., Protocol stack signature (31b), Protocol stack offsets (64b)
Rx Stack Accelerator
Rx Stack Processing

- **Can be decomposed in three major tasks:**
  - *(t1) Parsing*
    - Identify protocol stack + position of protocol fields
  - *(t2) Data extraction*
    - Locate and retrieve data to be processed
  - *(t3) Processing*
    - Execute instructions based on identified rules
      - Filtering (MAC, VLAN), VLAN extraction,
      - Rx queue assignment, checksum verification,
      - flow determination, discard, counters increments, …

- **Main characteristics exhibited by protocol processing applications [Jantsch1998]**
  - *(c1) Intensive use of pattern matching*
    - especially on headers
  - *(c2) Complex and control dominated flow*
    - many nested if-then-else and case structures
  - *(c3) Intensive use of irregular memory accesses*
    - various sizes and patterns
RXACC Architecture

A Transport Triggered Architecture (TTA) w/ 5 transport buses (1 byte/bus)

Cut-through architecture
→ low latency
→ efficient irregular mem. access (c3)

ILP architecture
→ instruction-level parallelism
→ parallel, independent, and pipelined coprocessors (FUs)
→ performance scalability
→ hardware efficiency
→ hardware modularity
→ “ultimate” RISC architecture
  - only 1 instruction
  🔷 “move data”

Prog. FSM + Balanced Routing Table search algorithm (B-FSM)
→ efficient pattern-matching (c1)
→ efficient one-cycle multiway branching (c2)

Frame “signature”
→ saves 300-400 instructions (avg.)

clk = 625 MHz

Data Path

Packet Handler

Packet Parser
(programmable)

octaword-in
(16 B)

octaword-out

(180 bits)

meta-data

Src

Dst

Ctrl

FU 1

FU N

Sockets
Data Path
(DP)
Multi-field Packet Inspection & Data Extraction

QQ/VLAN/SAP/SNAP/IPv6/UDP

0  8  16  24  32  40  48  56  64  72  80  88  96  104  112  120

FP

Current fields of interest

Next fields of interest

Off.5
Off.4
Off.3
Off.2
Off.1
Data Path Architecture

- Cut-through architecture
  - Relaxed buffering ($2 \times 16B$) + Low latency
  - Flexible data extraction (any 5 bytes within the 32B window)
  - Entire packet is exposed to the parser → Can inspect and match any data of the frame
Packet Parser (PP)
Packet Parser

Design space:

- **Micro-coded** → limited branch capabilities → multi-GHz operation → power
- **Finite state machine (FSM)** → efficient but inflexible
- **Programmable finite state machine (pFSM)** → high performance and energy efficient

State transition diagram

<table>
<thead>
<tr>
<th>Rule</th>
<th>Current State</th>
<th>Input symbols</th>
<th>Next State</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R44</td>
<td>S22</td>
<td>XXXX_XXXXb</td>
<td>S24</td>
<td>0</td>
</tr>
<tr>
<td>R52</td>
<td>S24</td>
<td>XXXX_0101b</td>
<td>S65</td>
<td>0</td>
</tr>
<tr>
<td>R53</td>
<td>S24</td>
<td>XXXX_XXXXb</td>
<td>S82</td>
<td>1</td>
</tr>
<tr>
<td>R61</td>
<td>S24</td>
<td>XXXX_XXXXb</td>
<td>S36</td>
<td>2</td>
</tr>
<tr>
<td>R112</td>
<td>S24</td>
<td>XXXX_0101b</td>
<td>S44</td>
<td>3</td>
</tr>
</tbody>
</table>

('X' symbol represents a “don't care” bit)

**HW Engine**

*We use the B-FSM architecture [van Lunteren2006]*

'B' stands for Balanced Routing Table search algorithm (BaRT)

Originally designed for longest matching prefix searches (i.e. routing table lookups)
Packet Parser Architecture

Input symbols

Output symbols

Transition Rule Selector

Input

Transition Rule Memory

(F, F', G, G', H')

Address Generator

Data Path Controller

Range Comparator

Match 1

Match 2

Match 3

Match 4

B1 - B3

B1 - B3

-transition

64 x 640 bits

256 rules

(1 rule = 160 bits)

Rule Structure

<table>
<thead>
<tr>
<th>Test part</th>
<th>Result part</th>
<th>H part</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_C</td>
<td>S_N</td>
<td>H sym.</td>
</tr>
</tbody>
</table>

64 x 640 bits

256 rules

(1 rule = 160 bits)
Packet Handler (PH)
Functional Units (FU)
- Independent coprocessors (IP-blocks) → Concurrent operation → ILP → Performance
- Configurable through MMIO
- Implement TTA socket registers:
  - O = operand register(s)
  - T = trigger register
    (note: result registers are not implemented)
### Simplified Instruction Set

- **Frame pointer instructions**
  - e.g. `ffpinc(4)`; // fixed increment
  - e.g. `vfptrc(1)`; // variable increment

- **Move instruction**
  - e.g.: `move(DP(4), CSI.O1);` // unicast destination
  - e.g.: `move(DP(12), HSH.O1 & CSI.O1 & CST.O9);` // multicast destination (socket write sharing)
Performance and Implementation Results
Implementation Results

■ Area (in 45 nm SOI technology)
  - $1 \times \text{RXACC} = 0.7 \text{ mm}^2$
  - $4 \times \text{RXACC} = 21.5 \% \text{ of HEA (entire HEA = 13 mm}^2)$

■ Clock Frequency
  - 625 MHz (27\% of core frequency)

■ Power (estimate)
  - $\text{RXACC} \approx 0.15 \text{ W (entire HEA = 2.6 W)}$

■ Transition Rule Memory Utilization
  - 256 rules → $64 \times (4 \times 160) \text{ bits} = 40 \text{ kbits}$
    • 68 states out of 128 (53 \%)
    • 221 rules out of 256 (86 \%)
  - Transition rule memory + ECC logic = 15 \% of RXACC
RXACC Bandwidth

Protocol stacks do not fit (missing bars)

Theoretical limit of 10 GbE media (Inter-frame gap=12B - Preamble=8B)
Summary & Conclusions

- **Processor compute complex + integrated network I/O complex**
  - IFF high-computation performance (1) + high-chip density (2) + low-power (3) + flexibility (4)
  - RXACC delivers (1) + (2) + (3) + (4)

- **(1) Performance**
  - 15 Mfps, 20 Gb/s (at “relaxed” clock frequency → 625 MHz)
  - Saves hundreds of CPU cycles per frame

- **(2-3) Area and power efficiency**
  - 0.7 mm² (45 nm SOI), 0.15 W

- **(4) Flexibility**
  - 2000+ protocol permutations
  - Programmable parsing and processing → Rule-based
    - Can parse new emerging standards (e.g. SDN)
    - Can inspect header and payload
    - Pragmatic approach w/ one code-set per application

- **RXACC = TTA + pFSM = novel Application Specific Processor**
  - Key enabler for integrated NICs
  - The architecture has headroom to scale towards 40-100 GbE
Rx Stack Accelerator for 10 GbE Integrated NIC

Thank you