Dynamic Hardware Plugins

Reconfigurable Hardware for High-Performance Programmable Routers

Jayakrishnan K Nair

Introduction

- With rapid evolution of Programmable Networks, there exist a growing need for scalable router architectures.
- Computationally intensive and diverse applications and streaming data are ever on the rise.
- Next-Generation Routers have hundreds of ports envisaged.
- Routers must have higher per-flow processing capacity, flexibility and performance at a reasonable per-port cost.

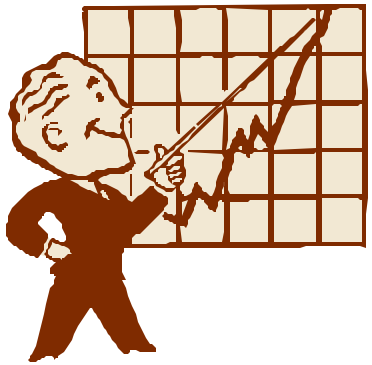
Existing Solutions – Enough?

- Software Processing Environments containing RISC cores
  - Good flexibility and low costs– good if processing need is limited
  - Per-Flow Processing capacities allotted in software
  - Performance not sufficient to meet today's requirements
- ASIC for high-end data processing (at line speeds of Gb/s)
  - Extracts good computational parallelism in hardware
  - Limited Flexibility for deploying new protocols and applications
  - Longer Design cycles and higher cost
- Can we combine these two complimentary approaches?

Technology Push

- Emergence of high performance reconfigurable hardware
  - FPGA – provides flexible hardware platform
  - Available configs of up to 1 million application logic gates at 200MHz
  - Over 100 KB of on-chip memory

- Terrific rate of recent technical progress (Moore overtaken!)
  - Clock frequency doubled in one year
  - On-Chip memory quadrupled in the same time
  - I/O bandwidth quadrupled in 2 years
  - Gate count increased over 10 times in same time



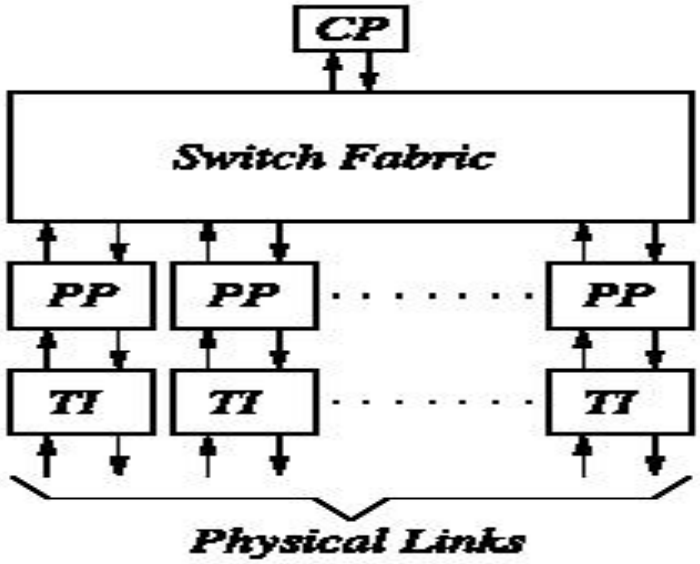
Related Ideas

- Scalable Software Processing Environment using elements with multiple RISC cores
- Dynamic switching of FPGAs in and out of datapath – flows routed through chains of applications as required
- Applications may be requested by incoming packets or deployed at session setup via signaling protocols

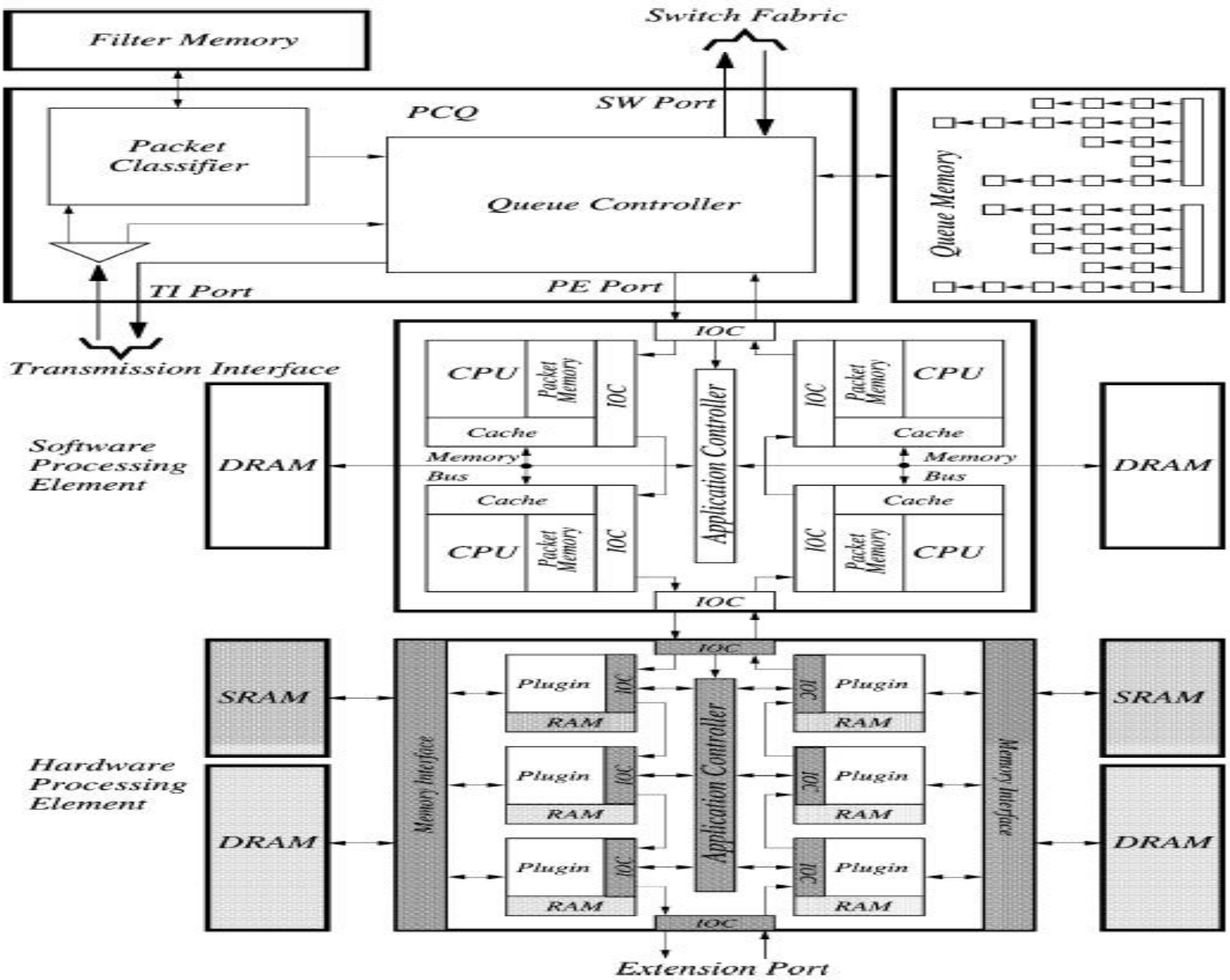
Dynamic Hardware Plugins

- Reconfigurable Hardware for providing flexible processing environments in multi-port routers
- Dynamically loads multiple hardware applications (plugins) into a single device
- Parallel Processing of Plugins for high per-flow performance
- Dedicated On-Chip Logic and memory resources for each plugin, and arbitrated access to off-chip resources

Programmable Routers



- Programmable router architectures must have compatible performance to current routers – link speeds ~2-10 Gb/s
- Scalable Multi-Stage Switch Fabric – 10 to 1000's of ports
  - PP – Port Processor – flow classification, forwarding,queuing
  - TI – Transmission Interface – converts data from link into std format
  - CP – Control Processor – external control for PPs. Manages flow classification data structures, filters and flow identifiers

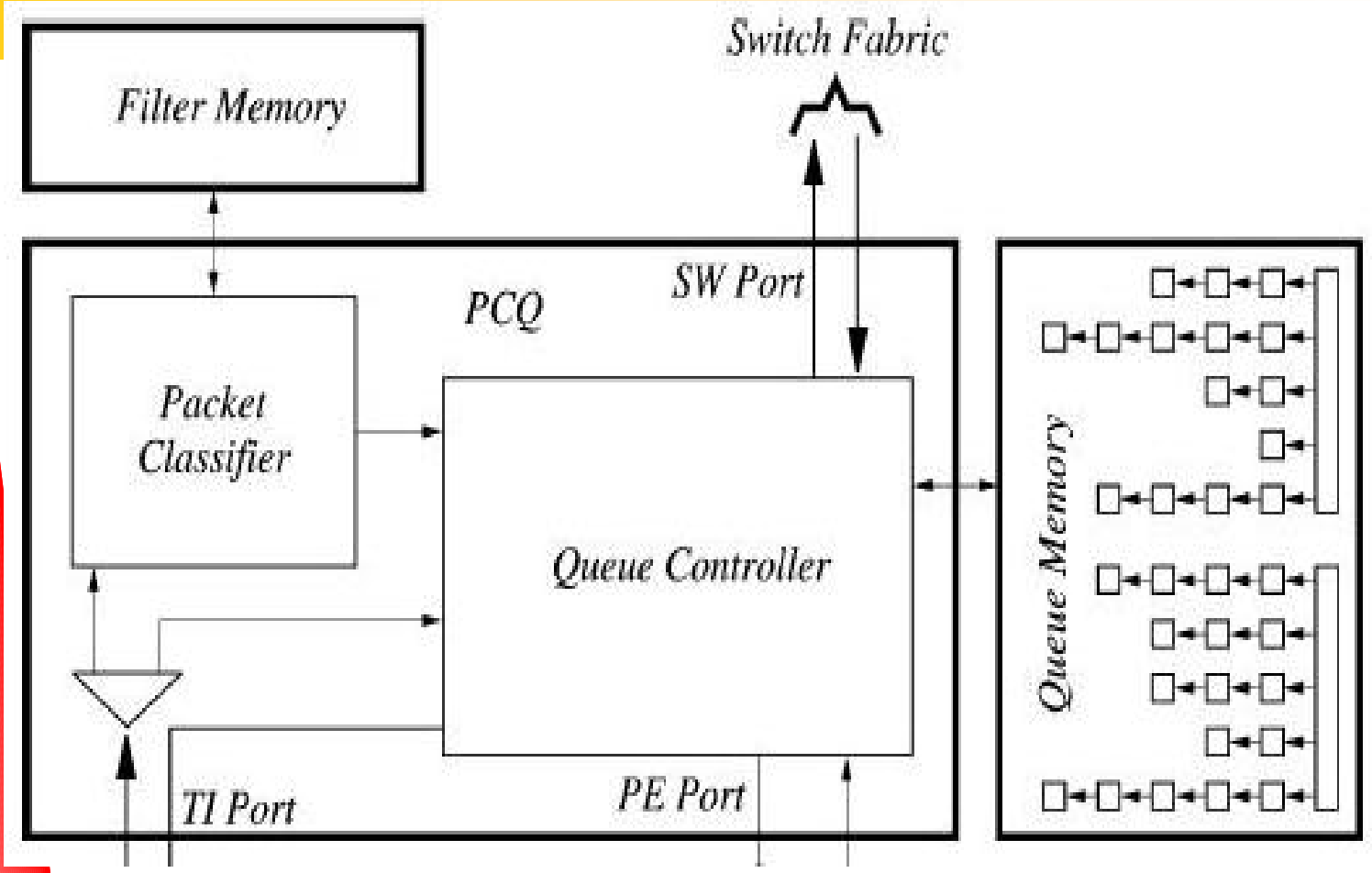


Port Processor Architecture

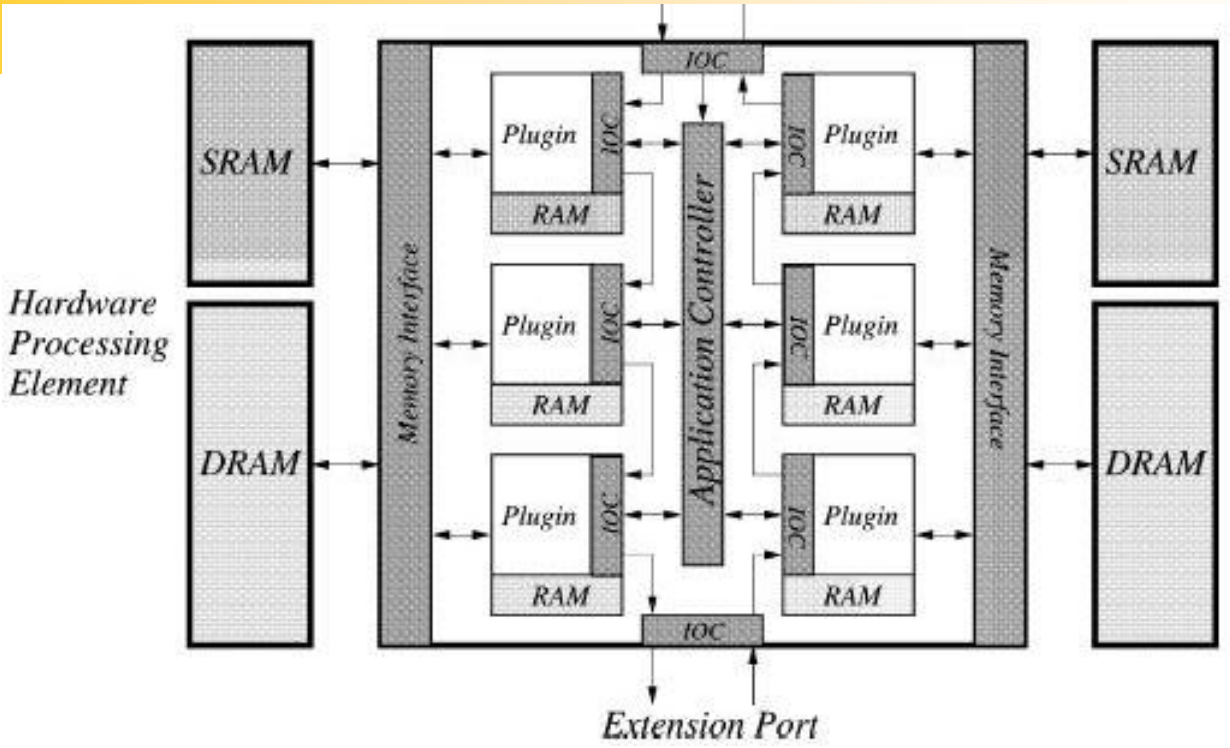
Port Processor Architecture

- PCQ – manages the flow of data through the device ports
  - TI port – Transmission Interface Port
  - SW port –Switching Fabric Port
  - PE Port – Processing Element Port
- Packet Classifier in PCQ assigns a Flow ID using a packet classification algorithm such as Pruned Tuple Space Search
- The Queue Controller places them on appropriate queues
  - Passive Packets are simply placed on the right output queue
  - Packets for processing are mapped to the right application and sent for processing to the PE port
  - Processed Packets are placed on the right output queue

PCQ Architecture



DHP Architecture



- The hardware processing element of PP is implemented in DHP.
  - Hardware Plugins – Dynamically Reconfigurable Components
  - Infrastructure – Static Control and Datapath components

Infrastructure

- Common Services to all hardware plugins
  - Supports dynamic, modular applications
  - Standard Interface to plugins - analogous to an API
- Input Output Controllers (IOC)
  - IOCs are the interfaces connecting components to the slotted ring
  - Ring architecture chosen for higher clock frequencies
  - One IOC for each plugin
  - IOCs interface upstream and downstream elements
  - Flow Control and Congestion Control achieved through bit vectors
  - Each IOC signals congestion by setting its allotted bit in the Flow Control Vector

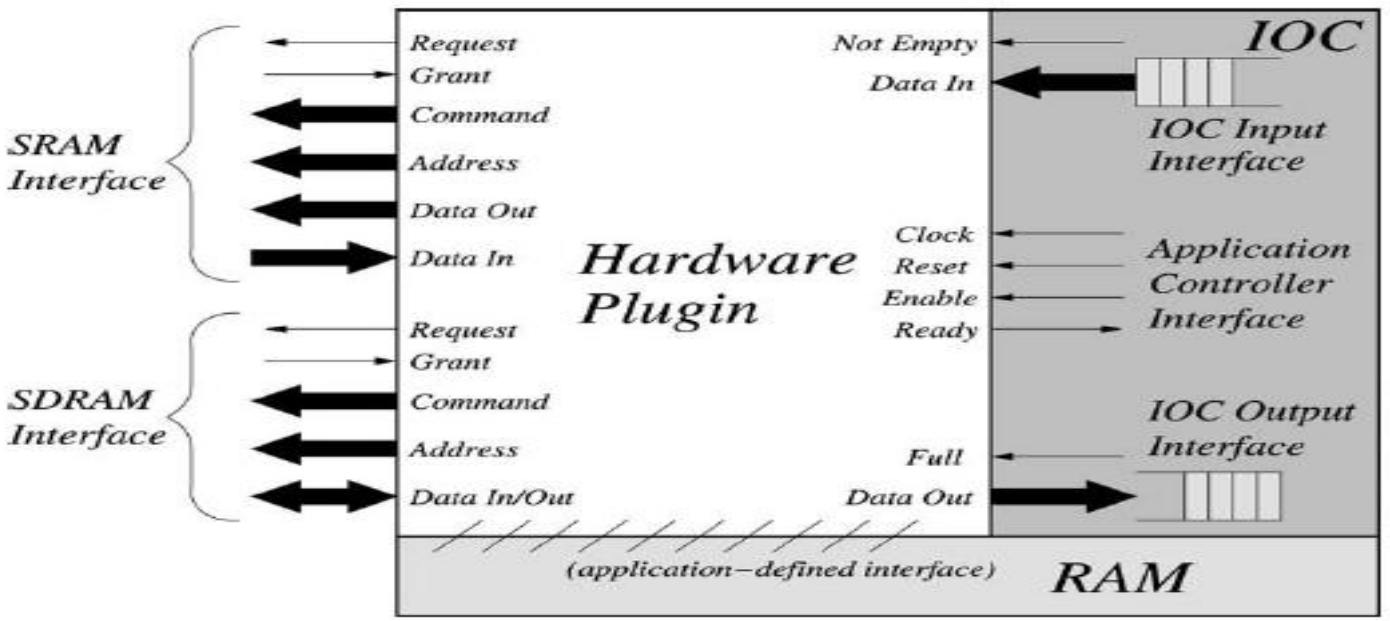
Application Controller

- Manages the dynamic reconfiguration of hardware plugins
- New Applications arrive as bitfiles that specify the logic operations, signal routing and on-chip memory configuration
- Assembles, buffers, checks integrity and then loads the bitfiles into the plugins for reconfiguration
- Asks an application, by handshaking, to stop accepting new packets and to finish, before being replaced by the new one
  - Prevents Data and flow state losses
  - Flow states can be saved off-chip if the application is replaced by a revision.

Memory Interfaces

- Memory interfaces arbitrate access among hardware plugins
  - Supports parallel utilization of off-chip resources
  - Insulates applications from device-specific timing constraints
- Two types of off-chip memory resources:
  - SRAM Banks – For per-flow state storage and low-latency accesses
  - DRAM Banks – For memory-intensive applications
- Architecture scalable to support DRAM and SDRAM technologies

Hardware Plugins

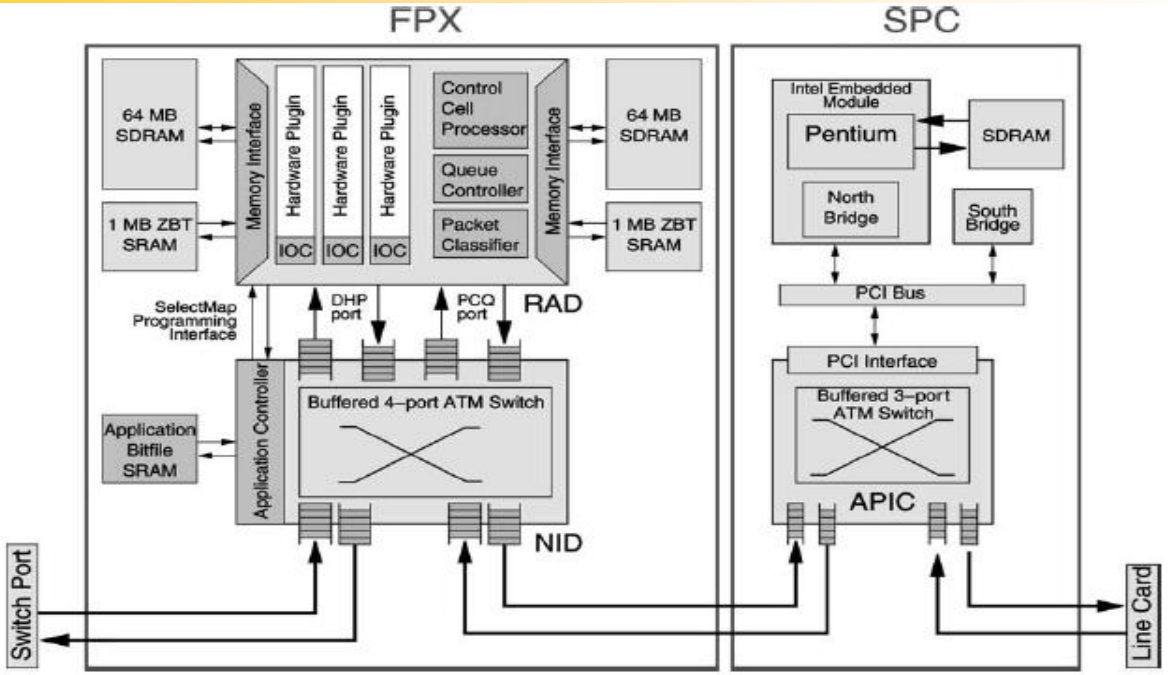


- The physical structures dynamically configurable to implement various networking applications
- Resources: Logic Gates, Look-up Tables, Flip-Flops, MUX and DEMUX, Signal Routing Matrices, Queues and Multiport Memories
- Implements Interfaces to SRAM, SDRAM, IOC and Application Controller

Comparison Studies

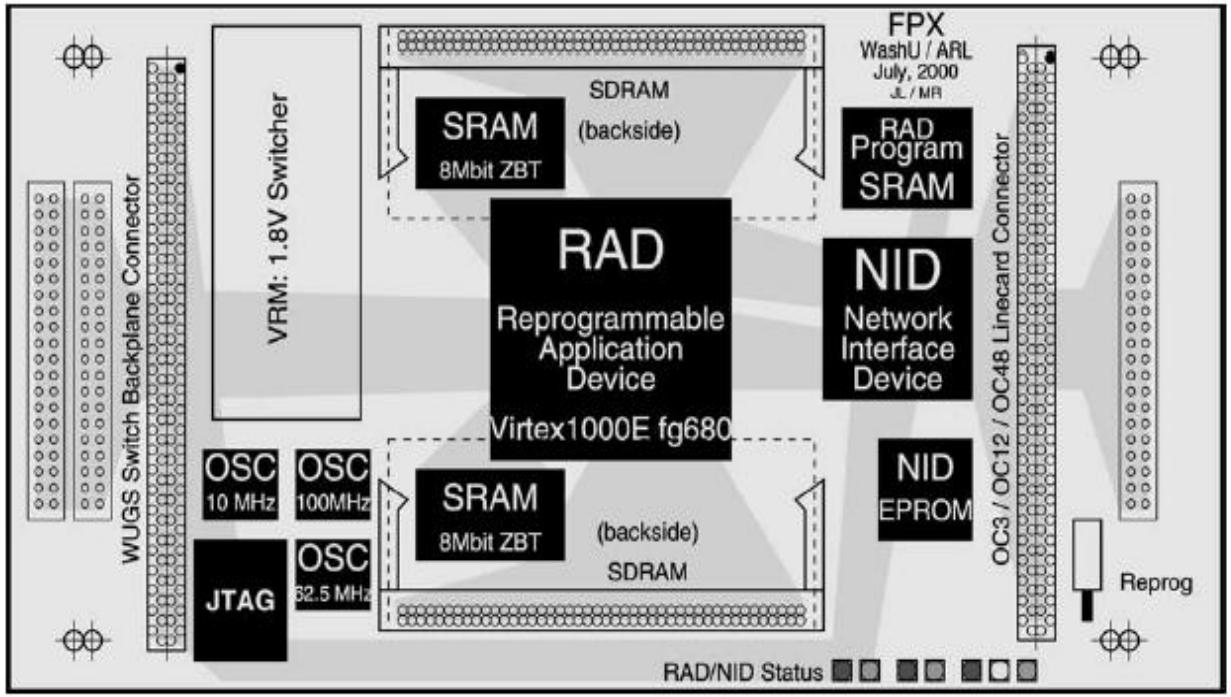
- A computationally-intensive and commonly used application used as benchmark – the Rijndael Algorithm, an Advanced Encryption Standard
- Software Implementation using RISC cores – 31.64 Mb/s
- ASIC Implementation – 5.16 Gb/s
  - Fully pipelined, non-reconfigurable, high cost ASICs - resource-crunching
- An iterative implementation of DHP - 353 Mb/s
  - ~20% use of resources on a Xilinx Virtex 3200E FPGA
  - Device Configuration delay ~5ms
- Five-Stage Partial Pipelined Implementation of DHP
  - Encryption and Decryption as separate plugins
  - ~40% Use of resources in a Xilinx Virtex 3200E FPGA
- Hence it is seen that DHP approach is a very good trade-off in achieving acceptable throughput, flexibility and economical use of resources

Prototype Testbed



- The testbed at Washington University in Saint Louis uses WUGS 20, and 8-port ATM switch as switch fabric, each port is fitted with FPX/SPC
- SPC – Smart Port Card prototypes the Software Processing Element
- FPX – Configurable Hardware processing element and PCQ

Prototype Testbed *(Continued…)*



- Field Programmable Port Extender is used to prototype DHP architecture
- Two FPGAs :
  - NID – Network Interface Device–> Implements Application Controller
  - RAD - Reprogrammable Application Device –> Hardware Plugins

Prototype Testbed *(Continued…)*

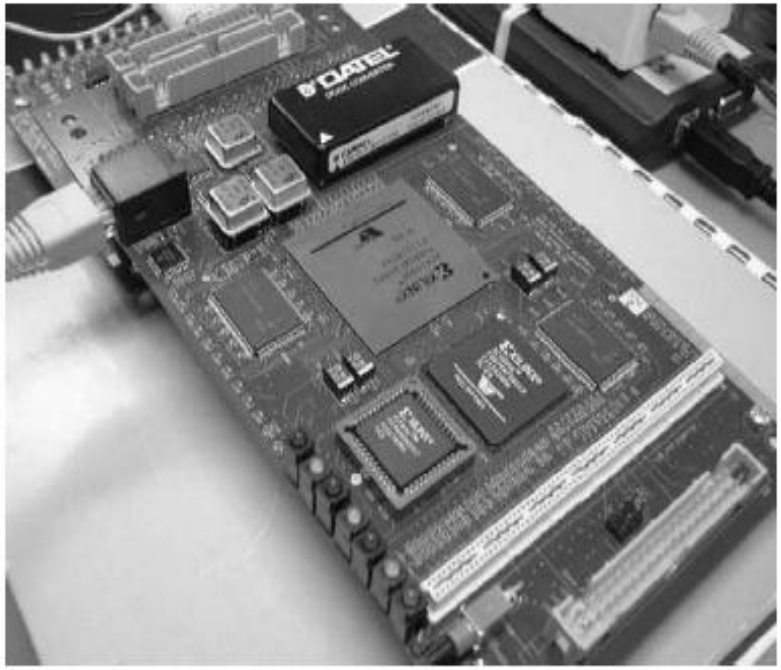


Photo of FPX

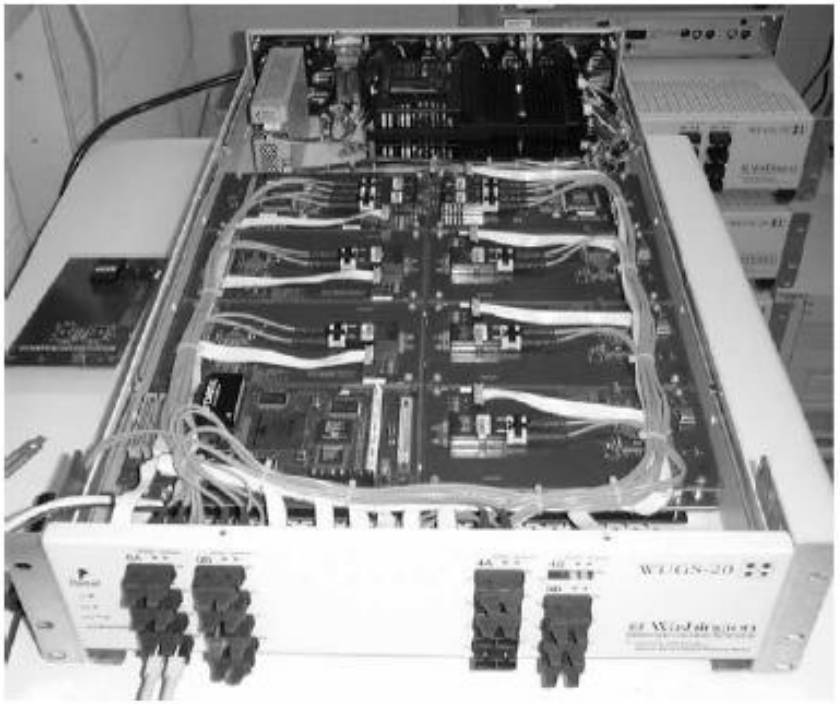
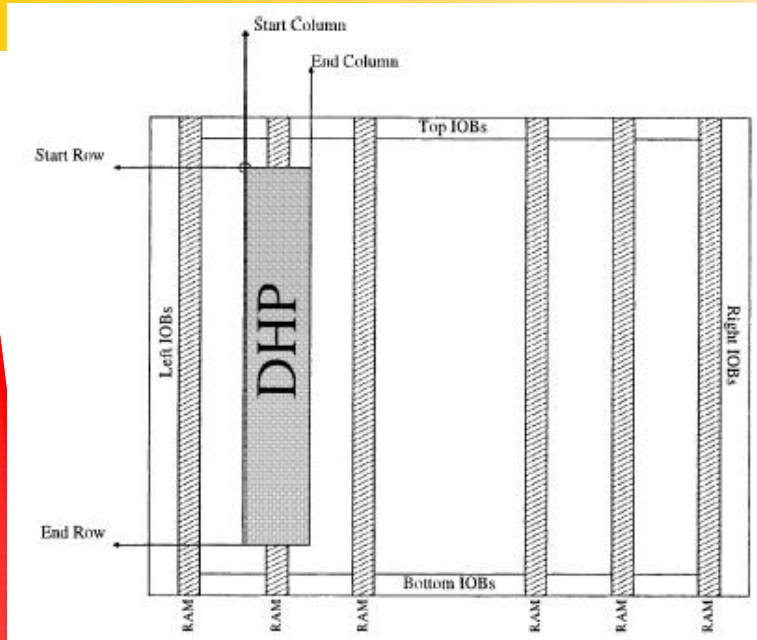
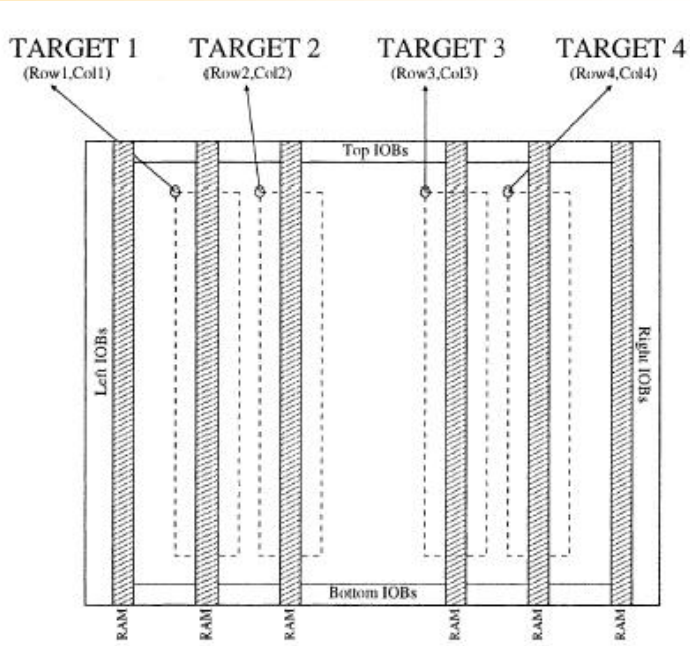


Photo of prototype environment for DHP architecture

PARBIT Tool





- PARBIT is a tool for transforming bit-files for implementing dynamically loadable hardware application
- It uses the Xilinx Architecture, by manipulating bitfiles and block regions of a compiled FPGA to dynamically reconfigure it

Conclusions

- DHP provides a scalable mechanism for building high-performance, multi-port routers capable of robust per-flow processing
- Constant evolution of better and cheaper reconfigurable hardware technology is a strong encouraging factor for the approach
- Provides performance throughputs unachievable by software, while comparable ASICs require far too much resources & design complexity.
- The DHP is a flexible, parallel and scalable mechanism as multiple hardware applications can be loaded into a single device.

Thank You Very Much

Questions?

Comments?