Virtual Acceleration Engine

Michael J. Miller
CTO
Virtualization

❖ Virtualization is a replication of hardware resources via software
❖ Virtualized resources ease deployment and improve HW utilization in datacenters and across networks
  ▪ Compute, storage, networking, applications
❖ First there was time sharing of CPUs & OS
  ▪ DEC, CDC, IBM, Unix, Tymeshare, VMWare, ...
❖ Then storage
  ▪ EMC, NFS, iCloud, SharePoint, DropBox, Box, ...
❖ Next was networking
  ▪ VPN, VLAN, NFV, OpenFlow, SDN, OVS, ...
❖ Now applications
  ▪ Containers, Open Virtual Formats (OVF), ….
❖ Flexibility is great, BUT it comes at a cost!
  ▪ Virtualizing can impact HW efficiency, throughput, latency and power
  ▪ Opportunity for flexible hardware acceleration
Hardware Acceleration
Addresses Bottlenecks, However…

❖ **Hardware accelerators speed up execution of well-defined tasks**
  ▪ Overcome compute, memory latency and bandwidth challenges
  ▪ Move the data and compute resources closer together
  ▪ Heterogeneous solutions with GPU and FPGA in datacenters today

❖ **Today’s challenges**
  ▪ SW that is dependent on HW acceleration is limited in portability
  ▪ CPU & DRAM perform poorly accessing random unstructured data
  ▪ Getting the data to the HW adds overhead and latency
  ▪ Hardware accelerators are often an after thought ➔ less optimal performance

❖ **Today’s solutions**
  ▪ High level: OpenCL on FPGAs and GPUs, CUDA on GPUs, etc.
  ▪ Very narrow: TCP Offload Engines

❖ **What about accelerating other embedded tasks?**
  ▪ Embedded data search (network addresses), packet classification, data analytics, anomaly detection, flow tracking analysis, security analysis, etc.
Embedded Packet Filtering Function in Various Network Interface Stacks

- Same virtualized Packet Filter function everywhere
  - Very desirable to have unified control software across all platforms
  - Provides for different cost, performance and capacity
Same virtualized packet classifier function and API at each level
- Different latency, throughput and capacity
- Minimizes total cost of development/ownership to manage all levels
Definition: Virtual Accelerator Engine (VAE)

- **Supports trend towards “virtual everything”**
  - Open Virtual Switch, Virtual Machines, Network Virtual Function, SDN

- **Abstract Virtual Function**
  - Defined precisely at a functional level
  - Given the same inputs, the same results will be produced
  - Transparent to implementation
  - Maps well into hardware
  - Embeddable (makes use of natural boundaries)

- **Common API and RTL Module Interfaces**
  - API provided as a software library model
  - Same across all implementations of the VAE
  - Supports adaptation layers to existing higher level application code API

- **Scalable implementation**
  - Lowest level performance & highest capacity: software on CPU core
  - Highest performance with FPGA (or ASIC) plus MoSys Si
  - Can be ported to future or alternative hardware solutions
Virtual Accelerator Engine Scalability
Software Programmable, Hardware Performance

- **VAE example opportunities in:**
  - Matching, filtering, classification, sorting, searching, structured computation

- **Scalable across many platforms**
  - From “C” → FPGA IP core w/BRAM → FPGA w/BE2 → FPGA w/PHE
  - Same high level software interface across all platforms
  - Same RTL Module Interface for FPGA and ASICs

30M ops/s per core
- CPU
  - Cust Code
    - “C” VAE IF
    - “C” VAE
    - BRAM
- FPGA
  - Cust RTL
    - VAE IF
    - VAE
    - BRAM

300M ops/s
- CPU
  - Cust Code
    - “C” VAE IF
    - “C” VAE
    - DRAM
- FPGA
  - Cust RTL
    - VAE IF
    - VAE
    - BRAM

600M ops/s
- CPU
  - Cust Code
    - “C” VAE IF
    - “C” VAE
    - DRAM
- FPGA
  - Cust RTL
    - VAE IF
    - VAE
    - GCI
    - SerDes
    - BE2
    - Memory

1.2B ops/s
- CPU
  - Cust Code
    - “C” VAE IF
    - “C” VAE
    - DRAM
- FPGA
  - Cust RTL
    - VAE IF
    - VAE
    - GCI
    - SerDes
    - BE2
    - Memory
    - HBM

3B ops/s
- CPU
  - Cust Code
    - “C” VAE IF
    - “C” VAE
    - DRAM
- FPGA
  - Cust RTL
    - VAE IF
    - VAE
    - GCI
    - SerDes
    - BE2
    - Memory
    - HBM
    - GCI
    - SerDes
    - PHE
    - VAE + Memory
MoSys Programmable HyperSpeed Engine (PHE) Device
Accelerating Algorithms + Data Structures

- **Monolithic Silicon**
  - High density 1T-SRAM 128MB + ECC
  - 32 high speed RISC cores

- **Tightly coupled cores & memory**
  - Direct connect via cross point Switch
  - No cache or TLB ➔ no miss variability

- **Optimized Instruction Set**
  - Hash, Compressed Trie etc.
  - Packed bit fields
  - 24b x 24b Multiplier

- **High parallelism**
  - Up to 8 way thread cores
  - 8 way threaded SRAM
  - 16 way Thread 1T-SRAM

- **Low latency memory access**
  - 6ns to 25ns
  - Up to 4x faster than DRAM
  - 2 Level hierarchy possible

Maximum Metrics
- 256 KB + ECC
  - @ 8 x 1.25GHz ➔ 12Bts & 8ns Latency
- 32 x 1.5GHz ➔ 48Bips & 256 Threads
- 128MB + ECC
  - @ 16 x 312MHz ➔ 6Bts (read) ➔ 6Bts (write) & 25ns Latency
Benefits of VAE

❖ Provides a wide range of performance/capacity: 100 to 1 possible
  ▪ Enables a wider range of product SKUs
  ▪ Easier adoption of HW acceleration
  ▪ Migration path for the future Si and Hardware

❖ Reduced time to deployment of new features
  ▪ Software definable function without the cost of RTL design timelines

❖ High level platform portability
  ▪ Higher layers of code are not limited to specific hardware
  ▪ Use available HW VAE or software version of VAE
  ▪ Enables “graceful fall back”

❖ SW investment is protected
  ▪ Insulated from hardware implementation by common API

❖ Programmers can take advantage of HW without knowing details
  ▪ Today’s software engineers are are focused on a higher level than firmware or RTL
  ▪ Allows programmers to focus on the bigger picture
Virtualized Graph Memory Engine
Abstract Graphs Can Be Used For Many Problems

RegEx

String Match
Patricia Tree

Longest Prefix Match

<table>
<thead>
<tr>
<th>ID</th>
<th>Prefix</th>
<th>Nextseq</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₀</td>
<td>10*</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₁</td>
<td>001*</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₂</td>
<td>1011*</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₃</td>
<td>0000*</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₄</td>
<td>0001*</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₅</td>
<td>0019*</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₆</td>
<td>00000</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₇</td>
<td>001011</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₈</td>
<td>00111*</td>
<td>NH₅</td>
</tr>
<tr>
<td>P₉</td>
<td>000010</td>
<td>NH₆</td>
</tr>
<tr>
<td>P₁₀</td>
<td>000111</td>
<td>NH₆</td>
</tr>
<tr>
<td>P₁₁</td>
<td>001000</td>
<td>NH₆</td>
</tr>
<tr>
<td>P₁₂</td>
<td>011000</td>
<td>NH₆</td>
</tr>
</tbody>
</table>

MoSys, Inc. Copyright 2019
Virtualized Graph Memory Engine
Block Diagram

- Graph Memory Engine is composed of:
  - **Graph Walker** including the Graph Memory
  - **Computation Engine** which computes the next Edge as a function of input vector
Graph Memory Engine API

- Edges connect two nodes \((n, m)\) with a unique edge value for a given node \(n\)
- Multiple graphs can be stored in the memory

```python
add_node(n, comp_edge_operation)  # add node \(n\) with next action
add_edge(n, m, edge_value)       # add an edge from \(n\) to \(m\)
add_default_edge(n, m)           # add a default edge from \(n\) to \(m\)
has_node(n)                      # test if node \(n\) exists
has_edge(n, edge_value)         # test if there is an edge from \(n\)
adjacent(n, m)                   # test if \(n\) and \(m\) are connected
list = neighbors(n)              # list of adjacent edges to \(n\)
list = edge_values(n)            # list of all edge values
func = comp_edge_function(n)     # returns compute edge function
action = action(n, edge_value)   # returns action associated with edge
action = default_action(n)       # returns action for default edge
delete_edge(n, edge_value)       # delete edge
delete_node(n)                   # delete node
```
MoSys Packet Classifier Platform utilizes an Adaption Layer

- Various Adaption Layers for building and maintaining search graphs
- Allows for options in function, throughput, latency and capacity
Example Multi-Field Match

Test For Rule 100

Test rule 100 @ tcan entry 100

Results

One 0 Results in 7 cycles ===== result entries: [] and disjoint results: [261, 74, 0, 100]
default hash table accesses => 3

MoSys, Inc. 2019. Proprietary
MoSys Packet Classification Platform

Example Control Planes

API Model

Adaption SW

Graph Memory Engine

Silicon Support

Hardware Data Plane Platforms

$n$-tuple, ACL, Ternary

User Designed

TCAM Compiler

Future (TBD)

GME in “C”

GME in FPGA RTL w/BRAM/HBM

GME in FPGA RTL + MoSys BE

GME in FPGA RTL + MoSys PHE

User Designed

User Designed

User Designed

Supplied by MoSys
MoSys “Cheetah” Accelerator Dev Kit
2 x 100G Network Accelerator

❖ Clone of Xilinx VCU1525 Card
  ▪ XCVU9P Xilinx FPGA (other assembly options possible)
  ▪ 4 x 16GB DDR4 DIMMS
  ▪ 2 x QSFP28

❖ MoSys PHE 1.5GHz
  ▪ 1 Gb of 1T-SRAM + 32 RISC cores

MoSys BE3 or PHE
Future VAEs: Data Analytics?

Profile Snap Shots Over Time

Distribution of Napier Flow Sizes
Spring 2000 (7,800,000 Events in 24 hours, May 21 & 23)

Shannon Entropy, ML Random Forest of Trees

Up to 36B Features/s

Anomaly Alert
Summary

❖ **Virtual Accelerator Engines provide:**
  - Software at hardware speeds
  - Wide range of performance/capacity: 100 to 1 possible
  - OEM platform portability
  - Preservation of software investments
  - Future proof to improvements in hardware acceleration

❖ **MoSys will be creating application specific platforms**
  - Using VAE technology for scalability
  - For applications that are dominated by random memory access and unstructured data
  - Require high throughput and low latency

❖ **Applicable for embedded functions:**
  - Searching, Filtering, Matching, Sorting, Security, Data analytics, Compute on Sparse or Random data

❖ **Equipment such as:**
  - Embedded Switching/Routing, Smart NICs, Security appliances, Application Severs, HPC, 5G edge, defense/aerospace, test/measurement equipment, datacenter acceleration
Thank You,
Questions?