Fabrics – Why We Love Them and Why We Hate Them

#OFADevWorkshop

Dave Dunning
Intel Corporation
Why we love Interconnect Fabrics, or, “I can design transportation systems”

• They move bits around…
  – Communication between resources
  – Start at some “address” and go to another “address”
    • On-die, fit everything into a nice gridded pattern, at an intersection, go towards your destination
    • Between die, travel along wires, when you get to a fork in the road, pick the path that goes to your destination

• How hard can it be to design?
  – Just copy our transportation systems, but make it better
Fabrics – Often the most “scrutinized” shared resource in a system

• The technology seems simple, can be visualized
  – Switches, arbiters, buffers, wires, tables, counters…
  – Distributed, usually organized, repetitive
  – “Features” can be “invented” by anyone with any background
  – Make them work the way I want to use them

The most common sentence uttered by a Fabric Designer:

“Stop helping me!”
Why we hate interconnect Fabrics, or Pretending packets are cars is just wrong

- Contexts, threads often don’t share resources well
  - Packets can’t back up
- Different flows often don’t mix or coexist very well
  - Order of arrival often matters
- Balancing the use of resources is often difficult
  - What seems like a good idea sometimes isn’t
Chip Scaling

- Moore’s Law – 2x transistor density increase every 2 years
- Dennard Scaling (MOSFET scaling) – Power density stays constant
Chip Scaling – How this affect Fabrics; A hardware View

• **WAS:**
  – Compute a scarce resource
  – Simple memory hierarchies
  – I/O was the bottleneck
  – Form factors (FF) – chip counts, chip I/O, connectors
  – Cost: Designing, building, # chips, wires, connectors

• **Key Metrics**
  – Performance (BW, freq.), cost

• **Less Key Metrics**
  – Latency, FF, power

• **IS:**
  – Compute inexpensive
  – Complex memory hierarchies
  – Off-chip I/O faster than on-chip
  – Form factors (FF) – packaging, cooling external I/O interfaces
  – Cost is components, integration/packaging, power

• **Key Metrics**
  – Performance (BW), Cost, Power, FF

• **Less Key Metrics**
  – Latency, frequency
Chip Scaling – How this affect Fabrics; A software View

• **WAS:**
  - Throw the HW “over the wall”, SW will find the performance
  - Maximize compute utilization
  - Users don’t worry about Memory, let the OS handle it
  - Cost: Time to released code, limited “legacy” code validation

• **Key Metrics**
  - Performance, cost, user interfaces

• **Less Key Metrics**
  - Power (system management)

• **IS:**
  - SW complexity due to sharing, parallelism
  - Compute abundant
  - Efficient memory usage to save power
  - Cost: Time to market, backwards compatibility design time

• **Key metrics**
  - Performance, power, user interfaces, cost

• **Less Key metrics**
  - None – SW rules the systems
Process (Chip) Scaling; Building Fabrics

- **On-die bandwidth**
  - Lots of traces available
  - Tools must mature

- **Energy (to move bits)**
  - Linear with distance
  - Energy to move bits is your enemy

- **Latency**
  - RC product
  - Increases exponentially with distance

---

*Cross die wires will need to be buffered*  
*Logic (switches) at clock cycle intervals*  
*Local BW is cheap, cross die BW will cost time and energy; optimize for locality*
Process (Chip) Scaling Is Helping You, but less so moving bits

<table>
<thead>
<tr>
<th>22 nm</th>
<th>pJoules</th>
<th>8 Bytes</th>
<th>Description</th>
<th>pJ/bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP Mul</td>
<td>6.4</td>
<td>A = B * C</td>
<td>8B/Operand</td>
<td>0.10</td>
</tr>
<tr>
<td>FP Add</td>
<td>8.1</td>
<td>A = B + C</td>
<td>8B/Operand</td>
<td>0.13</td>
</tr>
<tr>
<td>FMA</td>
<td>10.5</td>
<td>A = B * C + D</td>
<td>8B/Operand</td>
<td>0.16</td>
</tr>
<tr>
<td>Xbar Switch</td>
<td>0.86</td>
<td>8B per port</td>
<td>12 ports</td>
<td>0.01</td>
</tr>
<tr>
<td>On-die Wire</td>
<td><strong>11.20</strong></td>
<td><strong>8B per 5 mm</strong></td>
<td>50% toggle</td>
<td>0.18</td>
</tr>
<tr>
<td>Phys Reg File</td>
<td>1.2</td>
<td>8B R/W</td>
<td>2KB, 3 ports</td>
<td>0.02</td>
</tr>
<tr>
<td>SRAM</td>
<td>4.2</td>
<td>8B R/W</td>
<td>Small (8KB)</td>
<td>0.07</td>
</tr>
<tr>
<td>SRAM</td>
<td>16.7</td>
<td>8B R/W</td>
<td>Large (256KB)</td>
<td>0.26</td>
</tr>
<tr>
<td>In pkg DRAM</td>
<td>192</td>
<td>Stacks</td>
<td>64B accesses</td>
<td>3.00</td>
</tr>
<tr>
<td>Off Pkg DRAM</td>
<td>640</td>
<td>DDR</td>
<td>64B accesses</td>
<td>10.00</td>
</tr>
<tr>
<td>In pkg Wire</td>
<td><strong>19.2</strong></td>
<td>≤ 20 mm</td>
<td></td>
<td>0.30</td>
</tr>
<tr>
<td>Off pkg wire</td>
<td>128</td>
<td>≤ 200 mm</td>
<td></td>
<td>2.00</td>
</tr>
<tr>
<td>In Cab wire</td>
<td>320</td>
<td>≤ 2 m</td>
<td></td>
<td>5.00</td>
</tr>
<tr>
<td>Optical</td>
<td>640</td>
<td>≥ 2 m</td>
<td>Cost and area</td>
<td>10.00</td>
</tr>
</tbody>
</table>

Power = Energy * Frequency + Leakage

Will scale well with process and voltage
More difficult to scale down
Will scale well with process, less well with voltage
Can move to 2-4 pJ/bit range; Depends on demand, volume, leading to cost
Most challenging technologies to scale going forward

Copyright © 2015 Intel Corporation
Pulling it Together in the Future

• Formerly scarce resources are now plentiful
  – Compute, logic, on-die memory, wires
    • Technology can support very large bandwidths
  – Moving bits will dominate the power consumed
  – Memory: DRAM still scaling, lagging logic, other technologies may mature
  – Electrical wires remain the low cost choice, optical provides distance and minimizes cables, but requires power and $
    • Optical 5 – 15x higher $ per Gb/s than electrical for distances <2 meters

• Specialized building blocks will proliferate
  – Simpler, faster, lower energy, power gated (off) when not used
  – Sea of resources
  – OS problem, managing, sharing the resources
A Software View

• SW engineers must become system engineers
  – HW engineers have hidden many system challenges from SW engineers
    • SW engineers need to rethink what they are willing to “pay” for features
    • Do you want heavy weight Oses?
    • Memory management, Scheduling, Execution model
    • Methods to exploit parallelism
    • General purpose vs. specialization

• Standardization
  – Standardize when time-to-market is reduced
  – Cost reduction due to increased volume
  – Independent of implementation
Conclusions

• Communication Packets are not cars
• Process scaling Moore’s Law continues to provide more transistors/silicon area
  – Compute, logic trends to inexpensive
  – Communication trends to more expensive
• More specialization (accelerators, fixed function)
• SW evolving from performance to locality-based system management
• Standardize for time to market, cost between die, unclear on-die