# CMP SCI 635: Modern Computer Architecture

## Evaluation of the Raw Microprocessor:

## An Exposed-Wire-Delay Architecture for ILP and Streams

Name: Kunjal Panchal

Student ID: 32126469

Date: 17th Sept, 2019

Paper: 4 – Experience [Agarwal04]

Strengths:

1. The researchers tested Raw's ability to exploit all forms of parallelism, including ILP, DLP, TLP, and Stream parallelism. Specifically, they evaluated the performance of Raw on a diverse set of codes including traditional sequential programs, streaming applications, server workloads and bit-level embedded computation. And results were enough to say that Raw, with its different hardware organization, can truly be a viable candidate for the future of multithreaded computation heavy applications.

2. For each of the application classes, we can see that Raw is able to perform at or close to the level of the best-in-class machine.

3. It's ability to provide glueless peer-to-peer connection. The off-chip devices can connect through routing through on-chip network to communicate.

4. Raw ISA being backward compatible and bridging the gap between what hardware can offer and what software can actually use, which is unlike of conventional ISAs. Thus, it can have more functional units and efficient pin utilization. This makes Raw scalable.

Weaknesses:

1. To exploit Raw capabilities fully, we need a new ISA for it. There are already many established ones and telling programmers to learn and code in new ISA in the competing environment is infeasible and less likely to happen.

2. Raw has different specialized compilers for both sequential and ILP heavy applications, even then, it performs at suboptimal level (as low as 0.5x) when given sequential application programs with varying degree of ILP. This just means that a highly specialized tool won't work any better than a generalized tool when confronted with a slightly different situation. So, was the cost of effort and time worth making such hardware and compilers?

3. "*To go from corner to corner of the processor takes 6 hops, which corresponds to approximately six cycles of wire delay. To minimize the latency of inter-tile scalar data transport, which is critical for ILP, the on-chip networks are not only register-mapped but also integrated directly into the bypass paths of the processor pipeline.*" Although this is a strength of the architecture we are discussing on this

paper, we should also consider how this will picture out when using 100s or 1000s of tiles in a square grid. The cycling delay can go up exponentially. And the more the number of tiles, the greater the requirement of register for mapping (although, that's good for stream-based applications) and the greater the number of bypass paths, which will ultimately increase the pin count even more and the wire heat and latency will be no better. It is independent of transistor count, but not of tile count.

4. Dealing with cache misses is time costing because of no hierarchical caches. After L1 cache miss, the penalty is of 54 cycles and it would have to access DRAM,

Questions/Assertions:

1. "*Raw manages the effect of wire delays by exposing the wiring channel operators to the software…*" doesn't this stand a chance of getting maliciously used to get access to the lowest level of hardware and thus causing memory access violations? If these (gate, wire, pin) details are exploited in user mode, what about memory or paging protections?

2. Is it common to build two different compilers? One for ILP-based applications which might benefit from huge parallelism and other for sequential applications?

3. If we want to measure how an architecture would perform under certain stress conditions or commercial applications; why do we use simulators instead of emulator? Also, is there any difference between RTL Simulator and a logic emulator? Aren't both replicating the hardware as it is? Does emulator account for cycle costs, which RTL Simulator doesn't?

4. "*A tile contains an 8-stage in-order single-issue MIPS-style processing pipeline, a 4-stage single-precision pipelined FPU…*" What's a single-issue MIPS-style pipeline? And why does FPU needs only 4-stages? Compared to the former unit needing 8 stages?

5. Although memory-independent load-stores can be eliminated, what will happen when loads or stores have read or write dependencies? Can there be any speedup achieved there?

6. I couldn't find in the paper if Raw only supports in-order instruction execution, is this the case? Would re-order buffer windows do any good? Can it improve performance, if we look at latency and throughput trade-off?

7. When Raw architecture went commercial and became Tilera, it looked like the multicore future was finally coming. They had a lot of nice dev tools, like an eclipse-based simulator that let you visualize what was going on in the chip; 64 cores were really appealing; but because they didn't have floating point in the first set of chips, people mainly used them for network processing.

8. Can the individual idle cores be turned off? With the increase of number of cores running, there will be larger amount of power consumption too.