Architecture Goals

- **What?**
  - Handle increasing wire delay
  - Verification of new designs
  - Stream-based multi-media computation

- **How?**
  - Minimize the “ISA gap”
    - Export the low-level details of the architecture to the compiler
    - Access to gate, wire, and pin resources

---

“ISA gap”

- Gap between software-usable processing elements and actual underlying physical resources is steadily increasing
- 8-way superscalar 21464 is 27x as large as the 2-way 21064
- Management dwarfs area of ALUs
- Intel has 300+ page document on how to avoid stalls in the Pentium 4
- Power consumption, design and verification cost is sky-rocketing
Minimize the “ISA Gap”

- Minimize “ISA gap” by exposing physical resources as architectural entities

<table>
<thead>
<tr>
<th>Physical Entity</th>
<th>Raw ISA analogue</th>
<th>Conventional ISA Analogue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gates</td>
<td>Tiles</td>
<td>Dynamic mapping of sequential program to small # of ALUs</td>
</tr>
<tr>
<td>Wire delay</td>
<td>Network Hops</td>
<td>Dynamic stalls for non-fastpath and mispredicted code</td>
</tr>
<tr>
<td>Pins</td>
<td>I/O ports</td>
<td>Speculative cache-miss handling (prefetching and large line sizes)</td>
</tr>
</tbody>
</table>

**RAW Processor**

- Tightly integrated synchronous network interface
  - Register access latencies
  - Static scheduling of operands
  - Eliminates explicit synchronization
  - Single-cycle message injection and reception
- Multigranular (bit-, byte- and word-level) operations
- Configurable logic for application-specific operations
- Pipelined, point-to-point network between registers of different tiles increases ILP
- SRAM distributed across tiles decreases memory latency
- High-bandwidth paths to external devices (DRAM, I/O)
- Separate control for processing elements and static routing

**RAW Processor**

**Tile Processor Pipeline**
Interconnect Architecture

- RAW — switched point-to-point interconnect
- Superscalar — single register file and global bus
- Multiprocessor — coarser grained through memory

Programmable Interconnect

- Two compile time static networks
  - 5-stage pipeline static routing processor
  - 15 operations/cycle per tile
    - small command and 14 routes
  - Single-cycle/hop latency
- Two runtime dynamic networks
  - Dimension-ordered, wormhole-routed
  - “Memory” network with deadlock avoidance
    - Operating system, data cache, interrupts, DMA, I/O
  - “General” network relies on deadlock recovery
    - Untrusted clients

Architecture Comparison

- Systolic Arrays
  - Point-to-point networks for static scheduling
  - Startup costs are high and static scheduling for long periods is difficult
- FPGAs
  - Fine-grained parallelism and fast static communication
  - Does not support instruction sequencing
- VLIW Processors
  - Large number of operands and compiler-dependent
  - Does not support multiple instruction streams

Architecture Comparison

- Multiscalar processors
  - Deceptively similar except for exposure
  - Register renaming and dependency checking in hardware not software
- Single-chip multiprocessor
  - Expensive message startup and synchronization
- IRAM architectures
  - Long memory lines or long crossbar wires increase memory latency
Compilation

- Parallelizes C code onto static network
- View N tiles as functional units for ILP
- Partition parallel code to tiles
- Placement of threads to physical tiles
- Routing and global scheduling
- Logic synthesis for custom operations

Systolic Structures

for (i = 0; i < n; i++)
a[b[i]] = a[c[i]]

- Value dependency between a load and store of a
- Two-cycles/iteration
- RAW compiler handles memory dependencies

Results on RawLogic Prototype

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Benchmark access</th>
<th>Door count (K)</th>
<th>No. of elements</th>
<th>No. of gate (thousand)</th>
<th>No. of FPGAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>butterflop</td>
<td>butterflop</td>
<td>92</td>
<td>75</td>
<td>67</td>
<td>10</td>
</tr>
<tr>
<td>Ming23</td>
<td>Ming23</td>
<td>30</td>
<td>75</td>
<td>70</td>
<td>10</td>
</tr>
<tr>
<td>Bubble sort</td>
<td>Bubble sort</td>
<td>60</td>
<td>75</td>
<td>87</td>
<td>10</td>
</tr>
<tr>
<td>Gift</td>
<td>Gift</td>
<td>20</td>
<td>75</td>
<td>1.20</td>
<td>10</td>
</tr>
<tr>
<td>Cholesky</td>
<td>Cholesky</td>
<td>64</td>
<td>86</td>
<td>1.30</td>
<td>20</td>
</tr>
<tr>
<td>Fibonacci</td>
<td>Fibonacci</td>
<td>16</td>
<td>86</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Cholesky</td>
<td>Cholesky</td>
<td>17</td>
<td>86</td>
<td>347</td>
<td>20</td>
</tr>
<tr>
<td>Simplex</td>
<td>Simplex</td>
<td>12</td>
<td>256</td>
<td>1.20</td>
<td>20</td>
</tr>
<tr>
<td>Convolution</td>
<td>Convolution</td>
<td>21</td>
<td>724</td>
<td>372</td>
<td>20</td>
</tr>
<tr>
<td>Integer multiply</td>
<td>Integer multiply</td>
<td>20</td>
<td>724</td>
<td>372</td>
<td>20</td>
</tr>
<tr>
<td>Merge sort</td>
<td>Merge sort</td>
<td>32</td>
<td>54</td>
<td>24</td>
<td>1.20</td>
</tr>
<tr>
<td>G-Sort</td>
<td>G-Sort</td>
<td>12</td>
<td>54</td>
<td>24</td>
<td>1.20</td>
</tr>
<tr>
<td>Reverse</td>
<td>Reverse</td>
<td>12</td>
<td>54</td>
<td>24</td>
<td>1.20</td>
</tr>
<tr>
<td>Negate</td>
<td>Negate</td>
<td>12</td>
<td>54</td>
<td>24</td>
<td>1.20</td>
</tr>
<tr>
<td>Multiplication</td>
<td>Multiplication</td>
<td>12</td>
<td>54</td>
<td>24</td>
<td>1.20</td>
</tr>
<tr>
<td>Brahma</td>
<td>Brahma</td>
<td>12</td>
<td>54</td>
<td>24</td>
<td>1.20</td>
</tr>
</tbody>
</table>

Reference Processor: Sparc 20/71
RawLogic: Ikos VirtucaLogic Emulator coupled with Sparc 10/51, 5 boards, 64 Xilinx 4013 FPGAs/board (25 MHz)
Conclusion

- Does RAW solve all computation problems?
  - Stream-based multi-media applications?
  - Scientific computation?