ESE534: Computer Organization

Day 22: April 14, 2010
Time Multiplexing

Tabula
• March 1, 2010
  – Announced new architecture
• We would say
  – w=1, c=8 arch.

Previously
• Saw how to pipeline architectures
  – specifically interconnect
  – talked about general case
• Saw how to reuse resources at maximum rate to do the same thing

Today
• Multicontext
  – Review why
  – Cost
  – Packing into contexts
  – Retiming requirements
  – Some components
• [concepts we saw in overview week 2-3, we can now dig deeper into details]

How often is reuse of the same operation applicable?
• In what cases can we exploit high-frequency, heavily pipelined operation?

…and when can we not?

How often is reuse of the same operation applicable?
• Can we exploit higher frequency offered?
  – High throughput, feed-forward (acyclic)
  – Cycles in flowgraph
    • abundant data level parallelism [C-slow]
    • no data level parallelism
  – Low throughput tasks
    • structured (e.g. datapaths) [serialize datapath]
    • unstructured
  – Data dependent operations
    • similar ops [local control -- next time]
    • dis-similar ops
Structured Datapaths

- Datapaths: same pinst for all bits
- Can serialize and reuse the same data elements in succeeding cycles
- Example: adder

Throughput Yield

FPGA Model -- if throughput requirement is reduced for wide word operations, serialization allows us to reuse active area for same computation

Remaining Cases

- Benefit from multicontext as well as high clock rate
  - i.e.
    - cycles, no parallelism
    - data dependent, dissimilar operations
    - low throughput, irregular (can’t afford swap?)

Single Context

- When have:
  - cycles and no data parallelism
  - low throughput, unstructured tasks
  - dis-similar data dependent tasks
- Active resources sit idle most of the time
  - Waste of resources
- Cannot reuse resources to perform different function, only same

Resource Reuse

- To use resources in these cases
  - must direct to do different things.
- Must be able tell resources how to behave
  - separate instructions (pinsts) for each behavior
Preclass 1

• How schedule onto 3 contexts?

Preclass 1

• How schedule onto 4 contexts?

Preclass 1

• How schedule onto 6 contexts?

Example: Dis-similar Operations

Multicontext Organization/Area

• $A_{\text{ctx}} = 80K \lambda^2$
  - dense encoding
• $A_{\text{base}} = 800K \lambda^2$

Preclass 2

• Area:
  - Single context?
  - 3 contexts?
  - 4 contexts?
  - 6 contexts?
### Multicontext Tradeoff Curves

- Assume Ideal packing: \( N_{\text{active}} = \frac{N_{\text{total}}}{L} \)

![Graph showing Multicontext Tradeoff Curves]

**Reminder:** Robust point: \( c'A_{\text{tot}} = A_{\text{base}} \)

### In Practice

**Limitations from:**
- Scheduling
- Retiming

### Scheduling Limitations

- \( N_A (\text{active}) \) – size of largest stage
- **Precedence:**
  - can evaluate a LUT only after predecessors have been evaluated
  - cannot always, completely equalize stage requirements

### Scheduling

- **Precedence limits packing freedom**
- Freedom do have
  - shows up as slack in network

- **Computing Slack:**
  - **ASAP (As Soon As Possible) Schedule**
    - propagate depth forward from primary inputs
    - depth = 1 + max input depth
  - **ALAP (As Late As Possible) Schedule**
    - propagate distance from outputs back from outputs
    - level = 1 + max output consumption level
  - **Slack**
    - slack = \( L + 1 - (\text{depth} + \text{level}) \) [PI depth=0, PO level=0]
Preclass 3

- With precedence constraints, 4 context evaluation needs
  - Number of physical compute blocks?

Preclass 4

- Slack on nodes?
- Where can schedule
  - B
  - C
  - F

- Where can schedule
  - B if F in 2?
  - Where can schedule
    - B if F in 3?

- Where want to put
  - B
  - C
  - F
- Physical blocks?
Reminder (Preclass 1)

Sequentialization

- Adding time slots
  - more sequential (more latency)
  - add slack
    - allows better balance

L=4 → N_L=2 (4 contexts)

Retiming

Multicontext Data Retiming

- How do we accommodate intermediate data?

Signal Retiming

- Single context, non-pipelined
  - hold value on LUT Output (wire)
    - from production through consumption
  - Wastes wire and switches by occupying
    - for entire critical path delay L
    - not just for 1/L'th of cycle takes to cross wire segment
  - How show up in multicontext?
ASCII→Hex Example

- Single Context: 21 LUTs @ 880K\(\lambda^2\)=18.5M\(\lambda^2\)

ASCII→Hex Example

- Three Contexts: 12 LUTs @ 1040K\(\lambda^2\)=12.5M\(\lambda^2\)

ASCII→Hex Example

- All retiming on wires (active outputs)
  - saturation based on inputs to largest stage

 ASCII→Hex Example

- Ideal=Perfect scheduling spread + no retime overhead

Alternate Retiming

- Recall from last time (Day 21)
  - Net buffer
    - smaller than LUT
  - Output retiming
    - may have to route multiple times
  - Input buffer chain
    - only need LUT every depth cycles

Input Buffer Retiming

- Can only take K unique inputs per cycle
- Configuration depth differ from context-to-context

DES Latency Example

- Single Output case
**ASCII→Hex Example**

- All retiming on wires (active outputs)
  - saturation based on inputs to largest stage

**Ideal=Perfect scheduling spread + no retiming overhead**

**General throughput mapping:**
- If only want to achieve limited throughput
  - Target produce new result every $t$ cycles
  1. Spatially pipeline every $t$ stages cycle = $t$
  2. Retime to minimize register requirements
  3. Multicontext evaluation w/in a spatial stage try to minimize resource usage
  4. Map for depth ($i$) and contexts ($c$)

**Benchmark Set**
- 23 MCNC circuits
  - area mapped with SIS and Chortle

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Mapped UFs</th>
<th>Path Length</th>
<th>Circuit</th>
<th>Mapped UFs</th>
<th>Path Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcm</td>
<td>167</td>
<td>8</td>
<td>crisc</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>mc7050</td>
<td>156</td>
<td>10</td>
<td>hitachi</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>top</td>
<td>167</td>
<td>8</td>
<td>me20</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>vg1</td>
<td>17</td>
<td>4</td>
<td>omron2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>clip</td>
<td>17</td>
<td>4</td>
<td>omron1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>connc</td>
<td>17</td>
<td>4</td>
<td>star</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>count</td>
<td>17</td>
<td>4</td>
<td>star2</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

**Multicontext vs. Throughput**

**Multicontext vs. Throughput**
General Theme
- Ideal Benefit
  - e.g. Active=N/C
- Logical Constraints
  - Precedence
- Resource Limits
  - Sometimes bottleneck
- Net Benefit
- Resource Balance

Beyond Area

Only an Area win?
- If area were free, would I always want a fully spatial design?

Communication Latency
- Communication latency across chip can limit designs
- Serial design is smaller → less latency

Optimal Delay for Graph App.

Optimal Delay Phenomena
What Minimizes Energy

- HW5

\[ B = \sqrt{N} \]

Components

DPGA (1995)

<table>
<thead>
<tr>
<th>Process</th>
<th>1.0 ( \mu ) CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip</td>
<td>7.1mm x 6.8mm</td>
</tr>
<tr>
<td>AE</td>
<td>144</td>
</tr>
<tr>
<td>Contexts</td>
<td></td>
</tr>
<tr>
<td>AE Area</td>
<td>( 640K \lambda^2 )</td>
</tr>
<tr>
<td>( A_{\text{base}} )</td>
<td>( 544K \lambda^2 )</td>
</tr>
<tr>
<td>( A_{\text{ctx}} )</td>
<td>( 24K \lambda^2 )</td>
</tr>
<tr>
<td>( A_{\text{base}} : A_{\text{ctx}} )</td>
<td>20:1</td>
</tr>
<tr>
<td>(nominal delay)</td>
<td>9ns</td>
</tr>
</tbody>
</table>

Xilinx Time-Multiplexed FPGA

- Mid 1990s Xilinx considered Multicontext FPGA
  - Based on XC4K (pre-Virtex) devices
  - Prototype layout in F=500nm
  - Required more physical interconnect than XC4K
  - Concerned about power (10W at 40MHz)

Xilinx Time-Multiplexed FPGA

- Two unnecessary expenses:
  - Used output registers with separate outs
  - Based on XC4K design
    - Did not densely encode interconnect configuration
    - Compare 8 bits to configure input C-Box connection
      - Versus \( \log_2(8) = 3 \) bits to control mux select
      - Approx. 200b pins vs. 64b pins

Tabula

- 8 context, 1.6GHz, 40nm
  - 64b pins

- Our model w/ input retime
  - \( 1M \lambda^2 \) base
    - \( 80K \lambda^2 \) / 64b pinst Instruction mem/context
    - \( 40K \lambda^2 \) / input-retime depth
  - \( 1M \lambda^2 + 8 \times 0.12M \lambda^2 = 2M \lambda^2 \to 4 \times \text{LUTs (ideal)} \)
  - Recall ASCIItoHex 3.4, similar for thput map

- They claim 2.8 \( \times \) LUTs
Admin

- Course Evaluations
- Final exercise updated
  - Baseline design and technology defined
  - Still planning to get more details on sparing-based scheme by Monday
- Office Hours today 2pm
  - Next week
- Reading for next week on web

Big Ideas

[MSB Ideas]

- Several cases cannot profitably reuse same logic at device cycle rate
  - cycles, no data parallelism
  - low throughput, unstructured
  - dis-similar data dependent computations
- These cases benefit from more than one instructions/operations per active element
  - $A_{ubx} << A_{active}$ makes interesting
  - save area by sharing active among instructions

[MSB-1 Ideas]

- Economical retiming becomes important here to achieve active LUT reduction
  - one output reg/LUT leads to early saturation
- $c=4\ldots8$, $I=4\ldots6$ automatically mapped designs roughly 1/3 single context size
- Most FPGAs typically run in realm where multicontext is smaller
  - How many for intrinsic reasons?
  - How many for lack of HSRA-like register/CAD support?