ESE534: Computer Organization

Day 23: April 21, 2014
Control

Previously

• Looked broadly at instruction effects
• Explored structural components of computation
  – Interconnect, compute, retiming
• Explored operator sharing/time-multiplexing
• Explored branching for code compactness

Today

• Instantaneous compute requirement vs. total compute requirement
• Control
  – data-dependent operations
• Different forms
  – local
  – instruction selection
• Control granularity
  – architecture space parameter

Control

• Control: That point where the data affects the instruction stream (operation selection)
  – Typical manifestation
    • data dependent branching
      – if (a!=0) OpA else OpB
      – bne
    • data dependent state transitions
      – new => goto S0
      – else => stay
  • data dependent operation selection

• Viewpoint: can have instruction stream sequence without control
  – i.e. static/data-independent progression through sequence of instructions is control free
    • C0→C1→C2→C0→C1→C2→C0→…
  – Similarly, FSM w/ no data inputs

Programmable Architecture

Day 9
Terminology (reminder)

- **Primitive Instruction** \( (\text{pinst}) \)
  - Collection of bits which tell a bit-processing element what to do
  - Includes:
    - select compute operation
    - input sources in space (interconnect)
    - input sources in time (retiming)

- **Configuration Context**
  - Collection of all bits \( (\text{pinsts}) \) which describe machine’s behavior on one cycle

Back to “Any” Computation

- Design must handle all potential inputs (computing scenarios)
- Requires sufficient generality
- However, computation for any given input may be much smaller than general case.

**Instantaneous** compute \( \ll \) potential compute

Preclass 1

```plaintext
if ((dx*dx+dy*dy)>threshold)
    z=cx*dx+cy*dy
else
    z=dx*dy+c3
```

- How many operations performed?
- Cycles?
- Compute Blocks needed?

Preclass

- Operations?
- Cycles?
- How did it do that? (reduce delay)

If-Conversion

```plaintext
if (P())
    G()
else
    H()
```
If-Conversion

- Trade-off:
  - Latency
  - Work

\[
\text{if } (P()) \\
\quad G() \\
\text{else } H()
\]

If-Conversion \(\approx\) Predicated Operations

\[
\begin{align*}
P1 &= P() \\
G() &= G() \\
H() &= H()
\end{align*}
\]

\[
\begin{align*}
P1 = & P() \\
P2 = & !P1 \\
P1 = & G() \\
P2 = & H()
\end{align*}
\]

Why important?

Pipelining Processor

- What happens if we pipeline between
  - Instruction Memory
  - Datapath

Instruction Control Latency

- For time-multiplexed (data-independent) sequencing
  - Can pipeline instruction distribution
  - Instruction memory read
- With data-dependent branching: decision \(\rightarrow\) PC \(\rightarrow\) distribution \(\rightarrow\) read
  - Latency becomes part of critical path
Clock Cycle Radius

- Radius of logic can reach in one cycle (45 nm)
  - Radius 10
    - Few hundred PEs
  - Chip side 600-700 PE
    - 400-500 thousand PEs
  - 100s of cycles to cross

Two Control Options

1. Local control
   - unify choices
     - build all options into spatial compute structure and select operation → Mux-conversion

2. Instruction selection
   - provide a different instruction (instruction sequence) for each option
   - selection occurs when chose which instruction(s) to issue

Two Control Options

1. Local control
2. Instruction selection

May use both within an application
- Local control in critical path, inner-loops, where latency rather than parallelism limited
- Instruction-selection coarse-grain selection
  - At coarse level
  - Or where have plenty of task parallelism so latency not limit computation

Video Decoder

E.g. Video decoder [frame rate = 33ms]
- if (packet==FRAME)
  - if (type==I-FRAME)
  - I-FRAME computation
  - else if (type==B-FRAME)
  - B-FRAME computation
- Millions of cycles per frame
  - Instruction control between frames
    - Local control within frames

Packet Processing

- If IP-V6 packet
- ...
- If IP-V4 packet
- ...
- If VoIP packet
- ...
- If modem packet
- ...

Inclass 4(a)

- Local or instruction issue control?
- Optimize average runtime on 2-issue VLIW
  if (odd(a))
    return(factor(a));
else
    return(2);
Inclass 4(b)

- Local or instruction issue control?
- Optimize completion on 4-issue VLIW

```
while (abs(f(xm)-y)>delta)
    if (((f(xh)>y) && (f(xm)<y)) ||
        ((f(xh)<y) && (f(xm)>y)))
        xl=xm; xm=(xh+xm)/2;
    else
        xh=xm; xm=(xl+xm)/2;
return(xm);
```

Control Granularity

Architectural Parameter(s)
For Instruction Selection

Inclass 5

WAIT: if (in.type==header)
    cnt=in.header_payload_size;
    checksum=0;
    goto RECEIVE;
else goto WAIT;
RECEIVE: checksum=checksum xor in;
data[packet][cnt]=in;
cnt--;
if (cnt==0) goto CHECK;
else goto RECEIVE;
CHECK: if (in==checksum)
    packet++;
goto WAIT

Inclass 5

- Preferred Architecture?

Inclass 5

- How support
  - Two ports
  - On two 2-issue VLIWs
  - with separate controllers?

Inclass 5

- How support
  - on one 3-issue VLIW
  - with single controller?
  - Instructions?
  - PC bits?
Instruction Control

• If FSMs (ports) advance orthogonally – (really independent control)
  – context depth => product of states
    • Product of PCs
  – i.e. with single controller (PC)
    • must create product FSM
    • which may lead to state explosion
      – N FSMs, with S states => S^N product states

Architectural Differences

• What differentiates a VLIW from a multicore?

Architectural Questions

• How many pinsts/controller?

Architecture Taxonomy

<table>
<thead>
<tr>
<th>PCs</th>
<th>Pinsts/PC</th>
<th>depth</th>
<th>width</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N</td>
<td>1</td>
<td>1</td>
<td>FPGA</td>
</tr>
<tr>
<td>1</td>
<td>N (48,640)</td>
<td>8</td>
<td>1</td>
<td>Tabula ABAX (A1EC04)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>32</td>
<td>Scalar Processor (RISC)</td>
</tr>
<tr>
<td>1</td>
<td>N</td>
<td>D</td>
<td>W</td>
<td>VLIW (superscalar)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Small</td>
<td>W*N</td>
<td>SIMD, GPU, Vector</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
<td>D</td>
<td>W</td>
<td>MIMD</td>
</tr>
<tr>
<td>16</td>
<td>1 (47)</td>
<td>2048</td>
<td>64</td>
<td>16-core</td>
</tr>
</tbody>
</table>

Architectural Questions

• Effects of:
  – Too many controllers?
  – Too few controllers?
  – Fixed controller assignment?
  – Configurable controller assignment?
Architectural Questions

• Too many:
  – wasted space on extra controllers
  – synchronization?
• Too few:
  – product state space and/or underuse logic
• Fixed:
  – underuse logic if when region too big
• Configurable:
  – cost interconnect, slower distribution

FSM Control Factoring Case Study

FSM Example (local control)

FSM Example

FSM Example (Instruction)

Local Control

• LUTs used ≠ LUT evaluations produced
• Counting LUTs not tell cycle-by-cycle LUT needs
FSM Example

- FSM -- canonical "control" structure
  - captures many of these properties
  - can implement with deep multicontext
    - instruction selection
    - can implement as multilevel logic
      - unify, use local control
  - Serve to build intuition

Partitioning versus Contexts (Area)

- Start with dense mustang state encodings
- Greedily pick state bit that produces
  - least greatest area split
  - least greatest delay split
- Repeat until have desired number of contexts

Partition to Fixed Number of Contexts

- Extend Comparison to Memory
  - Fully local => compute with LUTs
  - Fully partitioned => lookup logic (context) in memory and compute logic
  - How compare to fully memory?
    - Simply lookup result in table?
Big Ideas

[MSB Ideas]

• **Control**: where data effects instructions (operation)
  
  • Two forms:
    - local control
      • all ops resident → fast selection
    - instruction selection
      • may allow us to reduce **instantaneous** work requirements
      • introduce issues
        - depth, granularity, instruction load and select time

[MSB-1 Ideas]

• If-Conversion
  - Latency vs. work tradeoff
  - Intuition ➔ explored canonical FSM case
    - few context can reduce LUT requirements considerably (factor dissimilar logic)
    - similar logic more efficient in local control
    - overall, moderate contexts (e.g. 8)
      • exploits both properties … better than extremes

Memory FSM Compare (large)

<table>
<thead>
<tr>
<th>FSM</th>
<th>status</th>
<th>ins</th>
<th>outs</th>
<th>Min area (M^2)</th>
<th>Integral Area</th>
<th>Data &amp; Control Organization</th>
<th>Memory area (M^2)</th>
<th>FPGA area (M^2)</th>
<th>8-ch DPGA area (M^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>saa</td>
<td>16</td>
<td>7</td>
<td>7</td>
<td>27.0</td>
<td>2^{20}</td>
<td>2^{11}</td>
<td>27.0</td>
<td>36.9</td>
<td>21.4</td>
</tr>
<tr>
<td>cim6</td>
<td>13</td>
<td>7</td>
<td>7</td>
<td>23.0</td>
<td>2^{10}</td>
<td>2^{11}</td>
<td>27.0</td>
<td>36.9</td>
<td>21.4</td>
</tr>
<tr>
<td>keyb</td>
<td>19</td>
<td>7</td>
<td>2</td>
<td>20.4</td>
<td>2^{20}</td>
<td>2^{7}, 2</td>
<td>34.4</td>
<td>48.3</td>
<td>31.3</td>
</tr>
<tr>
<td>planet</td>
<td>18</td>
<td>7</td>
<td>19</td>
<td>16.3</td>
<td>2^{20}</td>
<td>2^{2}, 25</td>
<td>245.8</td>
<td>331.7</td>
<td>54.1</td>
</tr>
<tr>
<td>pno</td>
<td>24</td>
<td>8</td>
<td>8</td>
<td>95.8</td>
<td>2^{20}</td>
<td>2^{13}</td>
<td>127.6</td>
<td>72.0</td>
<td>34.2</td>
</tr>
<tr>
<td>t1</td>
<td>20</td>
<td>8</td>
<td>6</td>
<td>67.5</td>
<td>2^{20}</td>
<td>2^{11}</td>
<td>168.1</td>
<td>123.3</td>
<td>56.7</td>
</tr>
<tr>
<td>t2</td>
<td>20</td>
<td>8</td>
<td>6</td>
<td>67.5</td>
<td>2^{20}</td>
<td>2^{11}</td>
<td>168.1</td>
<td>123.3</td>
<td>56.7</td>
</tr>
<tr>
<td>s11</td>
<td>30</td>
<td>9</td>
<td>10</td>
<td>244.9</td>
<td>2^{20}</td>
<td>2^{11}</td>
<td>471.9</td>
<td>165.4</td>
<td>55.5</td>
</tr>
<tr>
<td>s168</td>
<td>32</td>
<td>8</td>
<td>10</td>
<td>308.6</td>
<td>2^{20}</td>
<td>2^{2}, 25</td>
<td>491.5</td>
<td>133.5</td>
<td>74.6</td>
</tr>
<tr>
<td>stay</td>
<td>30</td>
<td>9</td>
<td>10</td>
<td>276.5</td>
<td>2^{20}</td>
<td>2^{11}</td>
<td>294.9</td>
<td>163.3</td>
<td>57.0</td>
</tr>
<tr>
<td>s201</td>
<td>16</td>
<td>11</td>
<td>2</td>
<td>397.7</td>
<td>2^{20}</td>
<td>2^{2}, 7</td>
<td>590.6</td>
<td>33.4</td>
<td>12.8</td>
</tr>
<tr>
<td>sand</td>
<td>32</td>
<td>11</td>
<td>9</td>
<td>1101.0</td>
<td>2^{20}</td>
<td>2^{14}</td>
<td>1101.0</td>
<td>155.3</td>
<td>62.7</td>
</tr>
<tr>
<td>std3</td>
<td>25</td>
<td>18</td>
<td>19</td>
<td>1887.8</td>
<td>2^{20}</td>
<td>2^{2}, 25</td>
<td>3019.1</td>
<td>46.8</td>
<td>64.1</td>
</tr>
<tr>
<td>std4</td>
<td>18</td>
<td>19</td>
<td>2</td>
<td>1797.3</td>
<td>2^{20}</td>
<td>2^{2}, 7</td>
<td>2829.8</td>
<td>35.1</td>
<td>14.2</td>
</tr>
<tr>
<td>std10</td>
<td>47</td>
<td>19</td>
<td>7</td>
<td>584599.9</td>
<td>2^{20}</td>
<td>2^{13}</td>
<td>525446.1</td>
<td>47.4</td>
<td>25.6</td>
</tr>
</tbody>
</table>

Memory FSM Compare (notes)

• Memory selected was “optimally” sized to problem
  - in practice, not get to pick memory allocation/organization for each FSM
  - no interconnect charged

• Memory operate in single cycle
  - but cycle slowing with inputs

• Smaller for <11 state+input bits

• Memory size not affected by CAD quality (FPGA/DPGA is)

Admin

• Grading: HW7 done
• FM1 due Wednesday
• Office Hours on Tuesday 3:30-4:30pm
  - Shifting up, won’t be there past 4:30pm
• Reading for Wednesday on canvas