ESE534: Computer Organization

Day 23: April 19, 2010
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
- Control
  - data-dependent operations
- Different forms
  - local
  - instruction selection
- Control granularity as another parameter in our architecture space

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

Control
- **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
    - *E.g.* HW3…non-branching multiplier

Programmable Architecture

Day 6
Terminology (reminder)

- **Primitive Instruction** \( pinst \)
  - 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

if \((dx*dx+dy*dy)>\text{threshold})\)
    \(z=\text{cx}*dx+\text{cy}*dy\)
else
    \(z=\log(dx*dy)+c3\)

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

Preclass 1

- Delay?
- Operations?
- How do?

If-Conversion

if \((P())\)
    \(G()\)
else
    \(H()\)

- Trade-off:
  - Latency
  - Work
**Preclass**

- Operations
- Cycles

**Day 3**

**Idea**

- Compute both possible values and select correct result when we know the answer

**If-Conversion ~ Predicated Operations**

- \( P() \)
- \( G() \)
- \( H() \)

\[
P_1 = P() \\
P_2 = P() \\
P_1 \ G() \\
P_2 \ H() \\
\]

**Instruction Control Latency**

- For time-multiplexed (data-independent) sequencing
  - Can pipeline instruction distribution
  - Instruction memory read
  - Now decision \( \rightarrow \) PC \( \rightarrow \) distribution \( \rightarrow \) read latency becomes part of critical path

**Screwdriver Analogy**

- Need capability to handle
  - Slothead
  - Phillips
  - Torq
  - Hex...

- But only need one at a time...

**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
Packet Processing

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

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

LSM Example (local control)

Local Control

- LUTs used ≠ LUT evaluations produced
- → Counting LUTs not tell cycle-by-cycle LUT needs

FSM Example

Context 0 \((S1=0)\)

- Dout = 0
- NS0 = \(s0\cdot\text{Acyc}\cdot\text{myAddr}\cdot\text{Read}\)
- NS1 = S0

Context 1 \((S1=1)\)

- Dout = \(s0\)
- NS0 = \(s0\)
- NS1 = \(s0\)
FSM Example (Instruction)

Context 0 (S1=0)
- Dout = 0
- NS0 = /S0*Acyc*myAddr*Read
- NS1 = S0

Context 1 (S1=1)
- Dout = /S0
- NS0 = /S0
- NS1 = /S0

Local vs. Instruction

- If can decide early enough
  - and afford schedule/reload
  - instruction select → less computation
- If load too expensive
  - local instruction
    • faster
    • maybe even less capacity (AT)

Slow Context Switch

- When context selection is slow,
  Instruction selection profitable only at coarse grain
  - Xilinx ms reconfiguration times
- E.g. Video decoder [frame rate = 33ms]
  - if (packet==FRAME)
    • if (type==I-FRAME)
      - IF-context
    • else if (type==B-FRAME)
      - BF-context

Local vs. Instruction

- For multicontext device
  - i.e. fast (single) cycle switch
  - factor according to available contexts
- For conventional FPGAs
  - factor only for gross differences
  - and early binding time

FSM Control Factoring Experiment

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
Full Partitioning Experiment

- Give each state its own context
- Optimize logic in state separately
- Tools
  - mustang, espresso, sis, Chortle
- Use:
  - one-hot encodings for single context
  - smallest/fastest
  - dense for multicontext
  - assume context select needs dense

Full Partitioning (Area Target)

<table>
<thead>
<tr>
<th>FN</th>
<th>States</th>
<th>Single Context</th>
<th>Context per State</th>
<th>tool</th>
<th>Slots</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1246</td>
<td>989</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Full Partitioning (Delay Target)

<table>
<thead>
<tr>
<th>FN</th>
<th>States</th>
<th>Single Context</th>
<th>Context per State</th>
<th>tool</th>
<th>Slots</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1246</td>
<td>989</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comparison

- Assume stay in context for a number of LUT delays to evaluate logic/next state
- Pick delay from worst-case
- Assume single LUT-delay for context selection?
- savings of 1 LUT-delay => comparable time
- Count LUTs in worst-case state

Partitioning versus Contexts (Area)

<table>
<thead>
<tr>
<th>Contexts</th>
<th>Area (in LUTs)</th>
<th>Area in % of 2-context</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1246</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>989</td>
<td>79%</td>
</tr>
<tr>
<td>3</td>
<td>794</td>
<td>63%</td>
</tr>
<tr>
<td>4</td>
<td>604</td>
<td>48%</td>
</tr>
</tbody>
</table>

CSE benchmark

<table>
<thead>
<tr>
<th>Number of Contexts</th>
<th>Area in LUTs</th>
<th>LUT Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1246</td>
<td>100%</td>
</tr>
<tr>
<td>1</td>
<td>989</td>
<td>79%</td>
</tr>
<tr>
<td>2</td>
<td>794</td>
<td>63%</td>
</tr>
<tr>
<td>3</td>
<td>604</td>
<td>48%</td>
</tr>
</tbody>
</table>

Note: full partition may not be optimal area case
- e.g. intro example,
  - no reduction in area or time beyond 2-context implementation
  - 4-context (full partition) just more area (additional contexts)
Partitioning versus Contexts

(Delay)

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

Partitioning versus Contexts

(Heuristic)

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

Partitions to Fixed Number of Contexts

<table>
<thead>
<tr>
<th>FSM</th>
<th>States</th>
<th>Best Single Context</th>
<th>Area Ratio by Number of Context</th>
<th>Dense Encodings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Area Target</td>
<td>average ratio</td>
<td>1.00</td>
<td>1.51</td>
<td>0.86</td>
</tr>
<tr>
<td>Delay Target</td>
<td>average ratio</td>
<td>0.00</td>
<td>0.27</td>
<td>0.33</td>
</tr>
</tbody>
</table>

N.B. - more realistic, device has 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?

Memory FSM Compare

(small)

<table>
<thead>
<tr>
<th>FSM</th>
<th>status</th>
<th>ins</th>
<th>outs</th>
<th>Mix area (M(^2))</th>
<th>FPGA area (M(^2))</th>
<th>Flip-flop area (M(^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>coop</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>state</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>coop</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>state</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>coop</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
</tbody>
</table>

Memory FSM Compare

(large)

<table>
<thead>
<tr>
<th>FSM</th>
<th>status</th>
<th>ins</th>
<th>outs</th>
<th>Mix area (M(^2))</th>
<th>FPGA area (M(^2))</th>
<th>Flip-flop area (M(^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>coop</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>state</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>coop</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>state</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>coop</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>state</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
<tr>
<td>coop</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>2.11</td>
<td>2.11</td>
<td>2.11</td>
</tr>
</tbody>
</table>

N.B. - more realistic, device has fixed number of contexts.
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)

Control Granularity

Architectural Parameter(s)
Instruction Selection

• How support two ports on VLIW with single controller?

Preclass

```
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
```

Instruction Control

• If FSMs advance orthogonally
  - (really independent control)
  - context depth => product of states
    • for full partition
  - i.e. w/ single controller (PC)
    • must create product FSM
    • which may lead to state explosion
      - N FSMs, with S states => $S^N$ product states
Preclass

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

• How support on two VLIWs with separate controllers?

Architectural Questions

• How many pins/controller?

Architectural Questions

• How many pins/controller?
  • Fixed or Configurable assignment of controllers to pins?
    • …what level of granularity?

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
Admin

- Final Exercise
  - update with sparing-based design
  - discussion period end 4/26
- Remember online course feedback
- Reading for Wednesday online

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