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 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

**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.* HW4...non-branching multiplier

Programmable Architecture

Day 8 and 9
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 (*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

\[
\begin{align*}
\text{if } (P()) & \text{ then } G() \\
\text{else } & H()
\end{align*}
\]

If-Conversion ~ Predicated Operations

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

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

Why important?

Instruction Control Latency

- For time-multiplexed (data-independent) sequencing
  - Can pipeline instruction distribution
  - Instruction memory read

- Now decision → PC → distribution → read
  - latency becomes part of critical path

Clock Cycle Radius

- Radius of logic can reach in one cycle (45 nm)
  - Radius 10 (preclass 19: \( L_{seg}=5 \rightarrow 50\)ps)
  - 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
- ....

Control Granularity

Architectural Parameter(s)
For Instruction Selection

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

Preclass

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

- How support
  - two ports
  - on one 3-issue VLIW
  - with single controller?

Instruction Control

- If FSMs (ports) advance orthogonally
  - (really independent control)
  - context depth \(\Rightarrow\) product of states
  - Product of PCs
  - *i.e.* w/ single controller (PC)
    - must create product FSM
    - which may lead to state explosion
      - \(N\) FSMs, with \(S\) states \(\Rightarrow\) \(S^N\) product states

Architectural Differences

- What differentiates a VLIW from a multicore?

Architectural Questions

- How many pins/ts/controller?

Architecture Taxonomy

<table>
<thead>
<tr>
<th>PCs</th>
<th>Pints/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>WN</td>
<td>SIMD, GPU, Vector</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
<td>D</td>
<td>W</td>
<td>MIMO</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>2048</td>
<td>64</td>
<td>16-core</td>
</tr>
</tbody>
</table>

Architectural Questions

- How many pins/ts/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?

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

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

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

Local Control

- LUTs used ≠ LUT evaluations produced
- Counting LUTs not tell cycle-by-cycle LUT needs
FSM Example (Instruction)

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

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

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

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

<table>
<thead>
<tr>
<th>FSM</th>
<th>States</th>
<th>Best Single Context</th>
<th>Area Ratio by Number of Context</th>
<th>Delay Target</th>
<th>Average Ratio</th>
<th>Average Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Area Target</td>
<td>average ratio</td>
<td>1.00</td>
<td>1.51</td>
<td>0.86</td>
<td>0.86</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>average delta</td>
<td>0.00</td>
<td>-0.27</td>
<td>0.33</td>
<td>1.27</td>
<td>2.18</td>
</tr>
</tbody>
</table>

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

- 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)

Big 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

Admin

- Reading for Monday on Blackboard
- FM2 Monday