ESE534
Computer Organization

Day 9: February 13, 2012
Interconnect Introduction

Previously
• Universal building blocks
• Programmable Universal Temporal Architecture

Today
• Universal Spatially Programmable
• Crossbar
• Programmable compute blocks
• Hybrid Spatial/Temporal
• Bus
• Ring
• Mesh

Spatial Programmable

Day 8
Temporal Programmable

Spatially Programmable
• Program up “any” function
• Not sequentialize in time
• E.g. Want to build any FSM
Needs?
• Need a collection of gates.
• What else will we need?

Needs
• Need some registers
• Need way to programmably wire gates together

Multiplexer Interconnect
• Use a multiplexer for programmable interconnect
• Can select any source to be an input for a gate
• How big is an N-input multiplexer?

Sources?
• What are potential sources?
  – Inputs to circuit -- I
  – Outputs of gates -- G
  – Outputs of registers -- R
  – \( N = I + G + R \)

Sinks
• Which things need programmable inputs? (and how many?)
  – Circuit outputs -- O
  – Gate – needs one per input -- kG
    • Assuming k-input gates
  – Registers -- R
  – \( M = O + R + kG \)

N-input, M-output Multiplexing
• Area?
• Instruction Bits?
• Data input switching
  – Capacitance Switched?
  – Delay?
• Control input switching
  – Capacitance switched?
  – Delay?
Mux Programmable Interconnect

\[
\text{Area } M \times N = (I+G+R) \times (O+kG+R) = kG^2 + \ldots
\]

- Scales faster than gates!

Interconnect Costs

- We can do better than this
  - Touch on a little later in lecture
  - Dig into details later in term
- Even when we do better
- Interconnect can be dominate
  - Area, delay, energy
  - Particularly for Spatial Architectures
  - (saw in HW4, memory can dominate for Temporal Architectures)

Dominant Time

<table>
<thead>
<tr>
<th>Design</th>
<th>Path</th>
<th>Total Delay</th>
<th>LUT Delay</th>
<th>Inter. %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altera 10K130V-2</td>
<td>LUT-local-LUT</td>
<td>2.5 ns</td>
<td>2.1 ns</td>
<td>16%</td>
</tr>
<tr>
<td></td>
<td>LUT-row-local-LUT</td>
<td>6.6 ns</td>
<td>2.1 ns</td>
<td>68%</td>
</tr>
<tr>
<td></td>
<td>LUT-columns-local-LUT</td>
<td>11.1 ns</td>
<td>2.1 ns</td>
<td>81%</td>
</tr>
<tr>
<td></td>
<td>LUT-row-column-local-LUT</td>
<td>15.6 ns</td>
<td>2.1 ns</td>
<td>87%</td>
</tr>
<tr>
<td></td>
<td>LUT-row-fanout-local-LUT</td>
<td>28 ns</td>
<td>2.1 ns</td>
<td>90%</td>
</tr>
</tbody>
</table>

Dominant Power [Energy]

- XC4003A data from Eric Kusse (UCB MS 1997)
- [Virtex II, Shang et al., FPGA 2002]

Crossbar

- Allows us to connect any of a set of inputs to any of the outputs.
- This is functionality provided with our muxes
Crossbar Structure

- Can be more efficient

Crossbar Costs

- Area still goes as $M \times N$
- Delay proportional to $M + N$
  - More realistic even for mux implementation
- Energy still goes as $M \times N$

Crossbar Notation

Gates with Crossbar Interconnect

Programmable Functions

Universal Computation with Fixed Compute Operator

- Being minimalists, show do not need compute to be programmable
  - Just use fixed nor2 or nand2
Mux can be a programmable gate

• bool mux4(bool a, b, c, d, s0, s1) {
  return(mux2( mux2(a,b, s0),
              mux2(c,d, s0),
              s1));
}

Mux as Logic

• bool and2(bool x, y)
  {return (mux4(false,false,false,true,
                x,y));}
• bool or2(bool x, y)
  {return (mux4(false,true,true,true,
                x,y));}
• Just by routing "data" into this mux4,
  – Can select any two input function

Programmable Compute

• Can use programmable gate in place of nor gate

Is an Adder Universal?

• Assuming interconnect:
  – (big assumption as we have just seen)
  – Consider:
    A: 001
    B: 000
    S: 00cd
• What’s c?

Practically

• To reduce (some) interconnect,
  and to reduce number of operations,
  do tend to build a bit more general
  “universal” computing function

Arithmetic Logic Unit (ALU)

• Observe:
  – with small tweaks can get many functions
    with basic adder components
ALU Size

- Adder took 6 2-input gates.
- How many 2-input gates did your ALU bitslice require? (HW4.1d?)

Alu Functions

- A+B w/ Carry
- B-A
- A xor B (squash carry)
- A*B (squash carry)
- /A

Instructions

- Identify the bits which control the function of our programmable device as:
  - Instructions

Slightly more conventional Programmable Architecture

Multibit Word Ops

- What are we doing when we make the ALU (and register file) width > 1
  - E.g. w=16 on HW4
- Benefit?
- Limitation?
Switching w-bit words

• Consider grouping outputs (inputs) into w-bit words
  – E.g. maybe operators are 16-bit ALUs
• How does this change switching requirements?
  – Don’t need to switch bit 3 to bit 7
  – Reduces switching needed

Switching w-bit words

• N/w w-bit inputs, M/w w-bit outputs
• Instruction Bits
  – Factor of w fewer outputs to switch
  – Factor of w fewer inputs \( \rightarrow M/w \log_2(N/w) \)
• Area:
  – Factor of w fewer switches, \( w^2 \) memories
• Delay:
  – Factor of w fewer sources
• Energy:
  – Factor of w fewer switches

Locality

• Maybe we don’t need to connect everything to everything?
• Cluster groups of C things at leaves
  – CG gates, CR registers
  – Limit cluster I/O – CI, CO
  – Crossbar within cluster
  – Crossbar among clusters
Comparing

- Full Crossbar needs: $kG^2$ switches
- How many switches needed for:
  - CG gates per cluster
  - CI inputs to cluster
  - CO outputs from cluster
  - (ignore registers and circuit input/output)

Costs

- Cluster: $G^2 \times (CI \times CO / CG^2) + k \times G \times (CG + CI + CO)$
- Full Crossbar: $kG^2$

Compare at: $CG=8$, $CI=CO=2$, $G=256$, $k=2$
  - Cluster case?
  - Full crossbar case?
**Extremes**

- Fully Spatial
- Fully Temporal

**General Case Between**

- How many concurrent operators?
- How much serialization?

**Separate Data Memory and Compute**

- Memory banks and compute

**Crossbar Generalized**

- What’s different about this crossbar?
  - Compared to one we used in purely sequential case in first part of lecture?

**Dynamic Crossbar**

- Need to switch crossbar configuration on each cycle

**Dynamic Crossbar**

- Switching time matters
- Must also supply crossbar controls
  - More wires into array
  - How many?
- Area
  - Bit-level switching case? $N(1 + \log(N))M$
  - W-bit word case? $-N(1 + \log(N)/W)M$
Note on Remainder

• Rest of lecture to introduce issues
  – Be illustrative
• Not intended to be comprehensive
• Will return to interconnect and address systematically starting on Day 15

Local Memory Case

• Put memory local to compute

Reduce Interconnect?

• How can we reduce interconnect?
  – Maybe don’t need to deliver a non-local value to every bank on every cycle?
  – Maybe don’t need to communicate everywhere?

Single Global Bus

• Pros and Cons?
Multiple Global Busses

- Can communication be per cycle
- Could be dominant area/energy
  - Don't want too large
- Could be bottleneck in computation?
  - Don't want too small
- Example of an architectural parameter

Nearest Neighbor Interconnect

- Compare to crossbar?
  - Number can transmit per cycle?
  - Area?

Ring Interconnect

- Compare to bus?
  - Area?
  - Cycle time?
  - Data transfers/cycle?
Ring Interconnect

- Compare to xbar?
  - Area?
  - Cycle time?
  - Data transfers/cycle?

Mesh Interconnect

- Nearest neighbors in 2D

Mesh Interconnect

- Compare to ring?
  - Area?
  - Latency?
  - Throughput?

Interconnect Design Space

- Large interconnect design space
- We will be exploring systematically
  - Day15—18+24

Admin

- Drop Date Friday
- HW5 out – 1 problem due Monday
  - Next due following Monday
- No class next Wednesday (2/23)
  - Class this Wednesday and Monday
  - Office hours this Tuesday (not next)
- Reading for Wednesday, Monday on Blackboard

Big Ideas

- Interconnect can be programmable
- Interconnect area/delay/energy can dominate compute area
- Exploiting structure can reduce area
  - Word structure
  - Locality