Last Time: Leon, an open-source SPARC

0.35μ, 65 MHz
40 mm²

Removal of FPU would reduce area. (power? cycle time?).
Today: Focus on Leon’s multiplier

- Configurable: Leon offers 5 multiplier design options.

- Mapping: on FPGAs, uses built-in multiplier and fast adder resources.

- Final Project: All groups must add a multiplier, one design option is a “fast” multiplier.
SPARC Unsigned Multiply: UMUL

General-purpose 32-bit registers

UMUL reg_rs1, reg_rs2 or immed, reg_rd

Q. Why use GP register for LSB destination?
Recall: Unsigned Multiply Algorithm

multiplicand 1101 (13)
multiplier * 1011 (11)

Partial products

1101
1101
0000
1101
10001111 (143)

Facts to remember

m bits x n bits = m+n bit product

Binary makes it easy:

• 0 => place 0 (0 x multiplicand)
• 1 => place a copy (1 x multiplicand)
Design #1: Spatially compute $A \times B = P$

1-bit signals: $x, y, z, s, Cin, Cout$

If $z = 1$, $\{Cout, s\} \leq x + y + Cin$
If $z = 0$, $\{Cout, s\} \leq y + Cin$
Array to compute $A \times B = P$

Q. Number of clock cycles?

Q. Clock cycle time?

Critical path?

Q. Can we pipeline?

What do we give up?

Q. What is the downside to a spatial design?
**Administrivia: Lab 4 and Homework 4**

- **11/5 (this Friday):** Lab 4 “milestone demo” in section.

- **HW 4:** due Weds 11/10, 5PM, 283 Soda. Problem list is now complete (no new problems were added).

- **11/12 (next Friday):** Lab 4 final demo in section.

- **11/15 (following Monday):** Lab 4 final report due, 11:59 PM.
Administrivia: Mid-term and Field Trip

Mid-Term II Review Session:
Sunday, 11/21, 7-9 PM, 306 Soda.

Mid-Term II: Tuesday, 11/23,
5:30 to 8:30 PM, 101 Morgan.

Xilinx field trip date: 11/30. Details on bus transport from Soda Hall soon.

Thanksgiving Holiday!
Design #2: Sequentially compute $A \times B = P$

Recall: Mini-Lab 2.

Q. Why is area so small?
Q. Clock cycle time?
Q. Does pipelining this design make sense?
Q. What is the downside of this design?

FPGAs support continuum: spatial to sequential.
Recall: Leon Configuration GUI ...

Five options for multiplier latency ...

1-cycle option is fully spatial.

35-cycle is mini-Lab 2.

2, 4, 5 cycles?
Trick: State machine + small spatial array

Q. With this general architecture, what needs to be “configured” in Leon VHDL to trade off space and latency?
### From Leon manual: Space vs Time

<table>
<thead>
<tr>
<th>Configuration</th>
<th>latency (clocks)</th>
<th>approx. area (Kgates)</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterative</td>
<td>35</td>
<td>1000</td>
</tr>
<tr>
<td>m16x16 + pipeline reg</td>
<td>5</td>
<td>6,500</td>
</tr>
<tr>
<td>m16x16</td>
<td>4</td>
<td>6,000</td>
</tr>
<tr>
<td>m32x8</td>
<td>4</td>
<td>5,000</td>
</tr>
<tr>
<td>m32x16</td>
<td>2</td>
<td>9,000</td>
</tr>
<tr>
<td>mx32x32</td>
<td>1</td>
<td>15,000</td>
</tr>
</tbody>
</table>

**Names indicate the size of multiplier array.**

**Leon will use FPGA multiplier arrays (not on Virtex E).**
Use technique shown in performance lecture to choose best latency, given the application program Leon will run.
Multiplier state machine design ... 

Think spatially, then map to time ... 

Figure 5: 22x16 Multiplier Implementation
Multiplier state machine design ...
In conclusion: Multiplier Design

- Design a multiplier state machine whose ALU is a smaller array.

- Trade off space and time to pick the best array size for your application.

- To design state machine, think in space then map to time.