EECS 151 Disc 9

Rahul Kumar (session 1)
Yukio Miyasaka (session 2)
Contents

- Memory composition
- FIFOs
- Direct-mapped caches
- Loop unrolling
- C slowing
Memory composition

Common memory configurations:

- 1 read/write port
- 1 read/write port, 1 read-only port
- 1 write-only port, 1 read-only port

Sometimes want different configurations.

Regfile: 2 read ports, 1 write port.
Memory composition: increasing width
Memory composition: increasing depth
Memory composition: adding read port
Memory composition: adding write port
Ready-valid interfaces

Source produces valid signal; sink produces ready signal.

Transaction occurs when ready and valid are high at a clock edge.
(If valid or ready is always 1, it can be omitted, where transaction may happen asynchronously.)
FIFOs

Read end:

- Read enable
- Empty
- Data out

Sometimes have an “almost empty” signal.
FIFOs

Write end:

- Write enable
- Full
- Data in

Sometimes have an “almost full” signal.
FIFOs
Asynchronous FIFO
Direct-mapped caches

Main idea: a given line can only be stored in one position.
Direct-mapped caches

- Suppose there are $2^M$ cache lines, each holding $2^B$ bytes. Suppose address is $A$ bits.
- Offset is $B$ bits.
- Index is $M$ bits.
- Cache tag is $A - M - B$ bits.
- Cache stores $2^{(M+B)}$ bytes of data.
Direct-mapped caches
Direct-mapped caches

Advantages:

- Only need to check one location, so hardware implementation is relatively simple.
- Fast, low power, low area.

Main disadvantage: lower hit rate compared to higher associativity caches.
Writeback policies

Writethrough: on every write, write to cache AND memory.

Writeback: write to memory only upon eviction of a cache line. Store a dirty bit to indicate if the line has been modified.
Loop unrolling

Example: serial adder. Performs one bit addition each cycle.

To add N-bit integers, run the adder for N cycles.
Loop unrolling

We can unroll with an interval 2. We generate 2 sum bits each cycle.

To add N-bit integers, run the adder for N/2 cycles.
Loop unrolling

- Fully unrolling the loop results in a ripple carry adder.
- The full N-bit addition occurs in 1 cycle.
- However, the logic depth is N. To make it shallow, we could parallelize or precompute carry calculation.
C-slowing

Example: want to compute $Y[i] = C(B(A(Y[i-1])))$, with pipelining.
# C-slowing

Many wasted cycles:

<table>
<thead>
<tr>
<th></th>
<th>A0</th>
<th>A1</th>
<th>A2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0=Y1</td>
<td></td>
<td>C1=Y2</td>
<td></td>
<td>C2=Y3</td>
</tr>
</tbody>
</table>
C-slowing

Solution: if there are 3 independent data streams (we’ll call them X, Y, and Z), we can fill the pipeline.

<table>
<thead>
<tr>
<th>AX0</th>
<th>AY0</th>
<th>AZ0</th>
<th>AX1</th>
<th>AY1</th>
<th>AZ1</th>
<th>AX2</th>
<th>AY2</th>
<th>AZ2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BZ(-1)</td>
<td>BX0</td>
<td>BY0</td>
<td>BZ0</td>
<td>BX1</td>
<td>BY1</td>
<td>BZ1</td>
<td>BX2</td>
<td>BY2</td>
</tr>
<tr>
<td>CY(-1)=</td>
<td>CX0=X1</td>
<td>CY0=Y1</td>
<td>CZ0=Z1</td>
<td>CX1=X2</td>
<td>CY1=Y2</td>
<td>CZ1=Z2</td>
<td>CX2=X3</td>
<td>Y0</td>
</tr>
</tbody>
</table>

Diagram:

```
Y(0) -----> A -----> B -----> C
```

Berkeley

University of California