Advanced Topic in Pipeline:
Pipeline scheduling

Contents
- Linear Pipelines
- Nonlinear pipelines
- Instruction Pipelines
- Arithmetic Operations
- Design of Multifunction Pipeline

Linear Pipeline
- Processing Stages are linearly connected
- Perform fixed function
- Synchronous Pipeline
  - Clocked latches between Stage i and Stage i+1
  - Equal delays in all stages
- Asynchronous Pipeline (Handshaking)
  - No latches between stage i and i+1
  - Special signals are required for acknowledgement

Quick Review: Floating Point
- We need a way to represent
  - numbers with fractions, e.g., 3.1416
  - very small numbers (in absolute value), e.g., .00000000023
  - very large numbers (in absolute value), e.g., -3.15576 * 10^{46}
- Representation:
  - scientific: sign, exponent, significand form:
    - \((-1)^{\text{sign}} \times \text{significand} \times 2^{\text{exponent}}\)
    - E.g., -101.001101 * 2^{111001}
  - more bits for significand gives more accuracy
  - more bits for exponent increases range
  - if 1 ≤ significand < 10_{\text{max}}(=2_{\text{max}}) then number is normalized, except for number 0 which is normalized to significand 0
    - E.g., -101.001101 * 2^{111001} = -1.01001101 * 2^{111001} (normalized)
IEEE 754 Floating-point Standard

- **IEEE 754 floating point standard:**
  - single precision: one word
  - double precision: two words

<table>
<thead>
<tr>
<th>31</th>
<th>bits 30 to 23</th>
<th>bits 22 to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sign</td>
<td>8-bit exponent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>bits 30 to 20</th>
<th>bits 19 to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sign</td>
<td>11-bit exponent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Floating Point Addition

- **Algorithm:**
  1. Compare the exponents of the two numbers. Shift the smaller number to the right until its exponent would match the larger exponent.
  2. Add the significands.
  3. Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent.
  4. Round the significand to the appropriate number of bits.

Floating Point Addition

- **Hardware:**
  - Start
  - Add the significands.
  - Normalize the sum.
  - Round the significand.
  - Check for exceptions.
  - End.
Latches

Slowest stage determines delay
Equal delays $\Rightarrow$ clock period

Reservation Table

- Each row corresponds to a stage in the pipeline.
- Each column corresponds to a pipeline cycle.
- An "X" at the intersection of the $i^{th}$ row and $j^{th}$ column indicates that stage $i$ is busy at cycle $j$.

5 tasks on 4 stages

Non Linear Pipelines

- Variable functions
- Feed-Forward
- Feedback
**Linear Instruction Pipelines**

- Assume the following instruction execution phases:
  - Fetch (F)
  - Decode (D)
  - Operand Fetch (O)
  - Execute (E)
  - Write results (W)
Dependencies

- Data Dependency
  (Operand is not ready yet)

- Instruction Dependency
  (Branching)

Will that Cause a Problem?

Solutions

- STALL
- Forwarding
- Write and Read in one cycle
- ….

Data Dependency

\[ I_1 \rightarrow \text{Add} R1, R2, R3 \]
\[ I_2 \rightarrow \text{Sub} R4, R1, R5 \]

Instruction Dependency
Solutions

- STALL
- Predict Branch taken
- Predict Branch not taken
- ...

Floating Point Multiplication

- Inputs \((\text{Mantissa}_1, \text{Exponent}_1), (\text{Mantissa}_2, \text{Exponent}_2)\)
- Add the two exponents → Exponent-out
- Multiply the 2 mantissas
- Normalize mantissa and adjust exponent
- Round the product mantissa to a single length mantissa. You may adjust the exponent

Linear Pipeline for floating-point multiplication

1. Add Exponents
2. Multiply Mantissa
3. Normalize
4. Round
5. Partial Products
6. Add Exponents
7. Accumulator
8. Normalize
9. Round
10. Re normalize

Linear Pipeline for floating-point Addition

1. Subtract Exponents
2. Partial Shift
3. Add Mantissa
4. Find Leading 1
5. Partial Shift
6. Round
7. Re normalize
Combined Adder and Multiplier

Reservation Table for Multiply

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reservation Table for Addition

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nonlinear Pipeline Design

- **Latency**: The number of clock cycles between two initiations of a pipeline
- **Collision**: Resource Conflict
- **Forbidden Latencies**: Latencies that cause collisions
Nonlinear Pipeline Design

- Latency Sequence
  A sequence of permissible latencies between successive task initiations
- Latency Cycle
  A sequence that repeats the same subsequence
- Collision vector
  \[ C = (C_m, C_{m-1}, \ldots, C_2, C_1), m \leq n-1 \]
  \( n \) = number of column in reservation table
  \( C_i = 1 \) if latency \( i \) causes collision, 0 otherwise

### Mul – Mul Collision (lunch after 1 cycle)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Mul – Mul Collision (lunch after 2 cycles)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Mul – Mul Collision (lunch after 3 cycles)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Collision Vector for Multiply after Multiply

Forbidden Latencies: 1, 2

Collision vector

(0 0 0 0 1 1) → 11

Latency: 6 5 4 3 2 1

Maximum forbidden latency = 2 → m = 2

Example: Reservation for X

Example: Reservation Tables for Y

Forbidden Latencies

• X after X
• X after Y
• Y after X
• Y after Y
**Collision Vector**

- Forbidden Latencies: \{2, 4, 5, 7\}
- Collision Vector = (1 0 1 1 0 1 0)

**Y after Y**

\[
\begin{array}{cccc}
S1 & Y & Y & Y \\
S2 & Y & Y & Y \\
S3 & Y & Y & Y \\
\end{array}
\]
Collision Vector

- Forbidden Latencies: \{2, 4\}
- Collision Vector = (1 0 1 0)

Pipeline Control

Place collision vector (CV) in shift register (SR)
If LSB of CV in SR is ‘1’
- do not initiate an operation at that cycle
- shift the CV right once
- insert ‘0’ at the MSB
Else
- initiate an operation at that cycle
- shift the CV right once
- insert ‘0’ at the MSB
- perform $CV_{org} + CV_{shift}$

HW– Find the collision vector

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>Stage 2</th>
<th>State 3</th>
<th>State 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1</td>
<td>0 0 1 1</td>
<td>0 0 0 1</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

Reduced state diagram

State 4
State Diagram for X

Cycles

- Simple cycles $\rightarrow$ each state appears only once
  (3), (6), (8), (1, 8), (3, 8), and (6, 8)

- Greedy Cycles $\rightarrow$ simple cycles whose edges are all made with minimum latencies from their respective starting states (1,8), (3) $\rightarrow$ one of them is MAL (Minimal Average Latency)
  Average latency for (1,8) = \(\frac{1+8}{2} = 4.5\)
  Average latency for (3) = \(\frac{3}{1} = 3\)