Pipeline Design, Stage Division, Optimum Performance, Wave Pipelining

Dr. A. Sahu
CSE IIT Guwahati

Analyzing performance

<table>
<thead>
<tr>
<th>Components</th>
<th>Delay</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>0</td>
<td>0ns</td>
</tr>
<tr>
<td>Adder</td>
<td>t_a</td>
<td>4ns</td>
</tr>
<tr>
<td>ALU</td>
<td>t_A</td>
<td>5ns</td>
</tr>
<tr>
<td>IMUX</td>
<td>0</td>
<td>0ns</td>
</tr>
<tr>
<td>RF</td>
<td>t_R</td>
<td>3ns</td>
</tr>
<tr>
<td>Instruction Memory</td>
<td>t_i</td>
<td>6ns</td>
</tr>
<tr>
<td>Data Memory</td>
<td>t_m</td>
<td>6ns</td>
</tr>
<tr>
<td>Bit manipulation</td>
<td>0</td>
<td>0ns</td>
</tr>
</tbody>
</table>

Single cycle data path (abstract)

1. Clock period in single cycle design

1. Clock period in multi-cycle design

Unbalanced delays

How to balance =>
* multiple actions in a period and/or
* multiple periods for an action
Pipeline Processor Design

Pipeline Design
- Single Cycle
  - Poor Resource Utilization, \( TC \geq \text{long instr latency} \)
- Multi Cycle
  - \( TC > \text{Loner Stage} \), Better Utilization, Still performance need to improve using pipeline
  - When Decoding \( \text{INS}_1 \), you can Fetch \( \text{INS}_{n+1} \)
- Pipeline

Instruction Pipeline

Performance: 1 instruction per Cycle
All the Stages work in parallel, No resource can be shared by stages

Single cycle data path (abstract)

Don’t share resources in Stages
- In Multi Cycle Design
  - ALU used for PC++ and Offset Adding
  - Shared: ALU, used for 1st Adder and 2nd Adder
  - Register FILE is used in 2nd and 4th Cycle
- In Pipeline
  - Use Separate resource 1st Adder, 2nd Adder & ALU
  - Register FILE is accesses 2nd Half of 2nd Cycle and 1st Half of 4th Cycle (To resolve RAW dependency)
Abstract: Adding control

Put back multiplexers

Correction for WB stage

Control signals with delays

Correction for RF write signal

Execution Time and Clock Period

Instruction execution time = $T_{inst} = CPI \times \Delta t$

IF D RF EX/AG M WB

Program exec time = $T_{prog} = N \times T_{inst} = N \times CPI \times \Delta t$

N : Number of instructions
CPI : Cycles per instruction(Av)
$\Delta t$ : Clock cycle time
What influences clock period?

\[ T_{prog} = N \times CPI \times \Delta t \]
- Technology - \( \Delta t \)
- Software - \( N \)
- Architecture - \( N \times CPI \times \Delta t \)
  - Instruction set architecture (ISA)
  - trade-off \( N \) vs CPI * \( \Delta t \)
  - Micro architecture (\( \mu \)A)
  - trade-off CPI vs \( \Delta t \)

Determining Clock Period

\[ T_{prog} = N \times CPI \times \Delta t \]
\[ N \downarrow \]

Technology - \( \Delta t \)

Software - \( N \)

Architecture - N * CPI * \( \Delta t \)

Instruction set architecture (ISA)

trade-off N vs CPI * \( \Delta t \)

Micro architecture (\( \mu \)A)

trade-off CPI vs \( \Delta t \)

Ideal Pipelining

\[ T_{inst} \]
\[ S \text{ stages} \]

\[ \Delta t = \frac{T_{inst}}{S} \]

CPI = 1

Effective time per inst \( T_{eff} = 1 \times \frac{T_{inst}}{S} \)

Pipelining with hazards

\[ T_{inst} \]
\[ S \text{ stages} \]

\[ \Delta t = \frac{T_{inst}}{S} \]

Probability of interruptions - \( b \)

CPI = 1 \( (1-b) \) + \( S \times b = 1-b+S*b = 1+(S-1)*b \)

\[ T_{eff} = (1+(S-1)*b) \times \frac{T_{inst}}{S} \]

A more realistic view

\[ T_{max} \]
\[ C \]

\[ \Delta t = P_{max} + C \]

\( P_{max} = \text{max propagation delay} \)

\( C = \text{clocking overhead} \)

Clocking Overhead

- Fixed overhead \( c \)
  - Setup time
  - Output delay
- Variable overhead
  - (stretching factor) \( k \)
  - Clock skew

\[ \Delta t = \frac{T_{inst}}{S + k \times T_{inst}} + c \]

\[ = (1+k) \times \frac{T_{inst}}{S + c} \]
**Pipelining with Clocking Overhead**

\[ T_{\text{eff}} = \left[ 1 + (S - 1) \times \frac{b}{b \times c} \right] \times \left[ (1 + k) \times \frac{T_{\text{inst}}}{S + c} \right] \]

\[ S_{\text{opt}} = \sqrt{\left[ (1 - b) \times (1 + k) \times \frac{T_{\text{inst}}}{(b \times c)} \right]} \]