Dual-Purpose Hardware Algorithms and Architecture: Part I – Floating-Point Division

Jihee Seo (Synopsys) – Presenter
Dae Hyun Kim (Washington State University)

ARITH 2023(Sep4-6, 2023)
Table of Contents

• Motivation
  – Architectural simulation of online arithmetic

• Review
  – Digit-recurrence offline division algorithm
  – Interval-analysis-based division (proposed)
    • Offline division
    • Online division

• Hardware architecture

• Simulation results

• Discussion

• Conclusion
Motivation

• Data dependency: Two examples in
  – GNU linear programming kit (GLPK)
  – GNU Scientific library (GSL)

\[
X = s \cdot \tan^{-1} \left[ \frac{a \cdot \sqrt{b \cdot c \cdot \cos(d - e) + f \cdot \sqrt{g \cdot h \cdot \cos(i - j)}} \cdot k}{l \cdot (m + n) + p \cdot (q + r)} \right]
\]

\[
Y = \sqrt{\frac{a}{b}} \cdot (c \cdot \cos x + d \cdot e \cdot \sin y)
\]
Motivation

\[ X = s \cdot \tan^{-1} \left[ \frac{a \cdot \sqrt{b \cdot c \cdot \cos(d - e) + f \cdot \sqrt{g \cdot h \cdot \cos(i - j)}}}{l \cdot (m + n) + p \cdot (q + r)} \right] \]

\[
\begin{align*}
    i_1 : & \quad x_1 = b \cdot c \\
    i_2 : & \quad x_2 = \sqrt{x_1} \\
    i_3 : & \quad x_3 = x_2 \cdot a \\
    i_4 : & \quad y = \cos(d - e) \\
    i_5 : & \quad x_4 = x_3 \cdot y
\end{align*}
\]
Motivation

- Resolving dependency chain: $i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow i_5$

\[
i_1 : x_1 = b \cdot c \\
i_2 : x_2 = \sqrt{x_1} \\
i_3 : x_5 = x_2 \cdot a \\
i_4 : y = \cos(d - e) \\
i_5 : x_4 = x_3 \cdot y
\]
Architecture Simulation

- Simulator
  - SESC (cycle-accurate MIPS32 simulator)

```
Benchmarks
(C/C++ code)
```

```
MIPS C compiler
```

```
MIPS executable
```

```
SESC simulator
```

```
# Instructions per cycle (IPC)
```
Architecture Simulation

• Benchmarks
  – B1: Finite element method (FEM)
  – B2: Electromagnetic (EM) simulation
  – B3: Statistical inference
  – B4: Numerical solver of the Bessel function

• # clock cycles

<table>
<thead>
<tr>
<th></th>
<th># Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer ALU</td>
<td>1</td>
</tr>
<tr>
<td>Integer Mult</td>
<td>4</td>
</tr>
<tr>
<td>Integer Div</td>
<td>15</td>
</tr>
<tr>
<td>FP Add/Sub</td>
<td>4</td>
</tr>
<tr>
<td>FP Mult</td>
<td>6</td>
</tr>
<tr>
<td>FP Div</td>
<td>16</td>
</tr>
</tbody>
</table>
## Architecture Simulation

- Profiling results (# instructions)

<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arithmetic</strong></td>
<td>59.8%</td>
<td>62.8%</td>
<td>74.8%</td>
<td>73.8%</td>
</tr>
<tr>
<td><strong>Non-Arithmetic</strong></td>
<td>40.2%</td>
<td>37.2%</td>
<td>25.2%</td>
<td>26.2%</td>
</tr>
<tr>
<td><strong>Integer ALU</strong></td>
<td>43%</td>
<td>35%</td>
<td>38%</td>
<td>29%</td>
</tr>
<tr>
<td><strong>Integer Mult &amp; Div</strong></td>
<td>0.1%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td><strong>Branch / Jump</strong></td>
<td>8.0%</td>
<td>8.2%</td>
<td>11.2%</td>
<td>8.1%</td>
</tr>
<tr>
<td><strong>Load / Store</strong></td>
<td>32.2%</td>
<td>29.0%</td>
<td>14.0%</td>
<td>28.1%</td>
</tr>
<tr>
<td><strong>FP Add/Sub</strong></td>
<td>11.4%</td>
<td>17.5%</td>
<td>27.7%</td>
<td>22.7%</td>
</tr>
<tr>
<td><strong>FP Mult</strong></td>
<td>4.6%</td>
<td>9.7%</td>
<td>8.9%</td>
<td>11.3%</td>
</tr>
<tr>
<td><strong>FP Div</strong></td>
<td>0.7%</td>
<td>0.6%</td>
<td>0.2%</td>
<td>0.8%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer ALU</td>
<td>1</td>
</tr>
<tr>
<td>Integer Mult</td>
<td>4</td>
</tr>
<tr>
<td>Integer Div</td>
<td>15</td>
</tr>
<tr>
<td>FP ALU</td>
<td>4</td>
</tr>
<tr>
<td>FP Mult</td>
<td>6</td>
</tr>
<tr>
<td>FP Div</td>
<td>16</td>
</tr>
</tbody>
</table>

B1 : Finite element method (FEM)
B2 : EM simulation
B3 : Statistical inference
B4 : Bessel function
• Dependency types
  – Multiplication → Multiplication (MM)
    • Ex: \((a \times b) \times c\) or \(c \times (a \times b)\)

  – Multiplication → Division (MD)
    • Ex: \(\frac{(a\times b)}{c}\) or \(\frac{c}{(a\times b)}\)

  – Division → Multiplication (DM)
    • Ex: \(\left(\frac{a}{b}\right) \times c\) or \(c \times \left(\frac{a}{b}\right)\)

  – Division → Division (DD)
    • Ex: \(\frac{\left(\frac{a}{b}\right)}{c}\) or \(\frac{c}{\left(\frac{a}{b}\right)}\)
Architecture Simulation

- Resolve dependencies

* Not drawn to scale.
* Other instructions are not shown.
• Resolve dependencies (simulation of ideal cases)
  – If there is a dependency between two arithmetic instructions (multiplications/divisions), then we assume that we can start the second instruction two cycles after the first instruction is launched.

* Not drawn to scale.
* Other instructions are not shown.
Architecture Simulation

• Reduction of the total # cycles (%)

\[(a \times b) \times c\]

or

\[c \times (a \times b)\]

\[\frac{(a \times b)}{c}\]

or

\[\frac{c}{(a \times b)}\]

\[\left(\frac{a}{b}\right) \times c\]

or

\[\frac{c}{\left(\frac{a}{b}\right)}\]

<table>
<thead>
<tr>
<th></th>
<th>MM (M→M)</th>
<th>MD (M→D)</th>
<th>DM (D→M)</th>
<th>DD (D→D)</th>
<th>Total (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>0</td>
<td>0.58</td>
<td>3.85</td>
<td>0.36</td>
<td>4.79</td>
</tr>
<tr>
<td>B2</td>
<td>5.71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.71</td>
</tr>
<tr>
<td>B3</td>
<td>0.56</td>
<td>0.25</td>
<td>0.33</td>
<td>0.43</td>
<td>1.57</td>
</tr>
<tr>
<td>B4</td>
<td>23.61</td>
<td>0</td>
<td>1.47</td>
<td>2.96</td>
<td>28.04</td>
</tr>
</tbody>
</table>

B1 : Finite element method (FEM)
B2 : EM simulation
B3 : Statistical inference
B4 : Bessel function
Offline Division: Digit-Recurrence (Review)

- Notation (floating-point)
  - Quotient obtained down to the $j$-th digit
    - $q[j] = q_0 \cdot q_1 q_2 \ldots q_j \text{(xx ...x)} = \sum_{i=0}^{j} q_i \cdot r^{-i}$

- Digit-recurrence division
  - Division: $x = d \cdot q + \text{rem}$
  - Condition: $|\text{rem}| < |d| \cdot ulp$
  - At $j$-th iteration
    $$\text{rem}[j] = x - d \cdot q[j]$$
    $$0 \leq \text{rem}[j] < d \cdot r^{-j}$$
    $$\Leftrightarrow 0 \leq x - d \cdot q[j] < d \cdot r^{-j}$$
    $$\Leftrightarrow 0 \leq r^j \cdot (x - d \cdot q[j]) < d$$
    $$w[j]$$
Offline Division: Digit-Recurrence (Review)

- Digit-recurrence division
  - At $j$-th iteration
    \[
    0 \leq r^j \cdot (x - d \cdot q[j]) < d
    \]
    \[
    w[j]
    \]
  - At $(j + 1)$-th iteration
    \[
    0 \leq w[j + 1] < d
    \]
    \[
    q[j + 1] = q_0 \cdot q_1 \ldots q_j q_{j+1}(xx \ldots x)
    \]
    \[
    \Rightarrow 0 \leq r \cdot w[j] - d \cdot q_{j+1} < d
    \]
    \[
    q_{j+1} = k \text{ if }
    \]
    \[
    0 \leq r \cdot w[j] - d \cdot k < d
    \]
    \[
    \Rightarrow k \cdot d \leq r \cdot w[j] < (k + 1) \cdot d
    \]

Radix-$r$ digit-recurrence division
Offline Division: Interval Analysis

- Interval-analysis-based division
  - Quotient obtained down to the $j$-th digit
    - $q[j] = q_0 \cdot q_1 q_2 \ldots q_j (xx \ldots x) = \sum_{i=0}^{j} q_i \cdot r^{-i}$

- Condition for $q_{j+1} = k$

\[
q[j] + k \cdot r^{-(j+1)} \leq \frac{x}{d} < q[j] + (k + 1) \cdot r^{-(j+1)}
\]

\[
\Leftrightarrow k \cdot d \leq r^{j+1} \cdot (x - d \cdot q[j]) < (k + 1) \cdot d
\]
Online division: Review

- **Operands**
  - Given one digit per cycle.
  - Example: \( x = 1.011010_2 \) (the leading 1 is hidden)

- **Principles**
  - Find the result with incomplete operands.
Online Division: Notation

• Input (given until the $j$-th iteration)
  
  – Dividend: $x[j] = x_0 \cdot x_1 \ldots x_{a[j]}(xx \ldots x) = \sum_{i=0}^{a[j]} x_i \cdot r^{-i}$
    
    \[
    \begin{array}{ll}
    \text{given} & \text{to be given} \\
    \end{array}
    \]

  – Divisor: $d[j] = d_0 \cdot d_1 \ldots d_{b[j]}(xx \ldots x) = \sum_{i=0}^{b[j]} d_i \cdot r^{-i}$
    
    \[
    \begin{array}{ll}
    \text{given} & \text{to be given} \\
    \end{array}
    \]

• Output (obtained until the $j$-th iteration)
  
  – Quotient: $q[j] = q_0 \cdot q_1 \ldots q_{c[j]}(xx \ldots x) = \sum_{i=0}^{c[j]} q_i \cdot r^{-i}$
    
    \[
    \begin{array}{ll}
    \text{obtained} & \text{to be obtained} \\
    \end{array}
    \]
Online Division

• Special case (most of the other papers)
  – $a[j] = j$: a digit of $x$ is given every cycle.
    • $x[j] = x_0 \cdot x_1 \ldots x_{a[j]}(xx \ldots x) = x_0 \cdot x_1 \ldots x_j(xx \ldots x)$
  – $b[j] = j$: a digit of $d$ is given every cycle.
    • $d[j] = d_0 \cdot d_1 \ldots d_{b[j]}(xx \ldots x) = d_0 \cdot d_1 \ldots d_j(xx \ldots x)$

<table>
<thead>
<tr>
<th>cycle</th>
<th>$x$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.1xxxxx</td>
<td>1.0xxxxx</td>
</tr>
<tr>
<td>1</td>
<td>1.11xxxxx</td>
<td>1.00xxxxx</td>
</tr>
<tr>
<td>2</td>
<td>1.110xxx</td>
<td>1.001xxx</td>
</tr>
<tr>
<td>3</td>
<td>1.1101xx</td>
<td>1.0011xx</td>
</tr>
<tr>
<td>4</td>
<td>1.11010x</td>
<td>1.00111x</td>
</tr>
<tr>
<td>5</td>
<td>1.110101x</td>
<td>1.001110x</td>
</tr>
<tr>
<td>6</td>
<td>1.1101010</td>
<td>1.0011100</td>
</tr>
</tbody>
</table>

• General case (our work)
  – An arbitrary number of new digits of $x$ and $d$ could be given every cycle.

<table>
<thead>
<tr>
<th>cycle</th>
<th>$x$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.110xxx</td>
<td>1.0xxxxx</td>
</tr>
<tr>
<td>1</td>
<td>1.110xxx</td>
<td>1.00111x</td>
</tr>
<tr>
<td>2</td>
<td>1.110xxx</td>
<td>1.00111x</td>
</tr>
<tr>
<td>3</td>
<td>1.110101</td>
<td>1.00111x</td>
</tr>
<tr>
<td>4</td>
<td>1.110101</td>
<td>1.00111x</td>
</tr>
<tr>
<td>5</td>
<td>1.110101</td>
<td>1.001110</td>
</tr>
<tr>
<td>6</td>
<td>1.110101</td>
<td>1.001110</td>
</tr>
</tbody>
</table>

(fully given)
Online Division: Interval Analysis

• Given
  – \( x[j] = x_0 \cdot x_1 \ldots x_{a[j]}(xx \ldots x) \)
    - \( x_{\text{MIN}} = x_0 \cdot x_1 \ldots x_{a[j]}0 \ldots 0 \) (padded 0’s)
    - \( x_{\text{MAX}} = x_0 \cdot x_1 \ldots x_{a[j]}1 \ldots 1 \) (padded 1’s)
  – \( d[j] = d_0 \cdot d_1 \ldots d_{b[j]}(xx \ldots x) \)
    - \( d_{\text{MIN}} = d_0 \cdot d_1 \ldots d_{b[j]}0 \ldots 0 \)
    - \( d_{\text{MAX}} = d_0 \cdot d_1 \ldots d_{b[j]}1 \ldots 1 \)

• Range of \( \frac{x}{d} = \left[ \left( \frac{x}{d} \right)_{\text{MIN}}, \left( \frac{x}{d} \right)_{\text{MAX}} \right] \)
  – \( \left( \frac{x}{d} \right)_{\text{MIN}} = \frac{x_{\text{MIN}}}{d_{\text{MAX}}} = \frac{x_0.x_1\ldots x_{a[j]}0\ldots0}{d_0.d_1\ldots d_{b[j]}1\ldots1} \)
  – \( \left( \frac{x}{d} \right)_{\text{MAX}} = \frac{x_{\text{MAX}}}{d_{\text{MIN}}} = \frac{x_0.x_1\ldots x_{a[j]}1\ldots1}{d_0.d_1\ldots d_{b[j]}0\ldots0} \)
Online Division: Interval Analysis

- Obtained
  - \( q[j] = q_0 \cdot q_1 ... q_{c[j]}(xx ... x) \)

- Find \( q_{c[j]+1} \)
  - \( q_{c[j]+1} = k \) if
    \[
    \frac{x}{d} \leq q_0 \cdot q_1 ... q_{c[j]} [k0 ... 0] < q_0 \cdot q_1 ... q_{c[j]} [(k + 1)0 ... 0]
    \]
    which means \( q[j] = q_0 \cdot q_1 ... q_{c[j]}^k \)...

- For example, for \( r = 4 \)
  - \( 1.3212 ... \leq \frac{x}{d} < 1.3213 ... \Rightarrow \frac{x}{d} = 03212 ... \)
    \( q-(c[j]+1) = 2 \)
Online Division: Interval Analysis

- **Case 1:** \( q_{-c[j]+1} = k \)

- **Case 2:** \( q_{c[j]+1} =? \)

- In the example above, \( q_{c[j]+1} \) could be \( k - 1 \) or \( k \).
- We need more digits of \( x \) and/or \( d \) to find \( q_{c[j]+1} \).
Online Division: Interval Analysis

- **Formula:** Two inequalities

\[
q[j] + k \cdot r^{c[j]} - 1 \leq \frac{x}{d}_{\text{MIN}}
\]

\[
\rightarrow q[j] + k \cdot r^{c[j]} - 1 \leq \frac{x_{n-1} \ldots x_{a[j]} 1 \ldots 0}{d_{n-1} d_{n-2} \ldots d_{b[j]} 1 \ldots 1} = \frac{x[j]}{d[j] + r^{b[j]} - 1}
\]

\[
\left(\frac{x}{d}\right)_{\text{MAX}} < q[j] + (k + 1) \cdot r^{c[j]} - 1
\]

\[
\rightarrow \frac{x_{n-1} \ldots x_{a[j]} 1 \ldots 1}{d_{n-1} d_{n-2} \ldots d_{b[j]} 0 \ldots 0} = \frac{x[j] + r^{a[j]} - 1}{d[j]} < q[j] + (k + 1) \cdot r^{c[j]} - 1
\]
Online Division: Interval Analysis

1. \[ q[j] + k \cdot r^{c[j]-1} \leq \frac{x[j]}{d[j] + r^{b[j]-1}} \] (Solve it for \( k = 0, 1, ..., r - 1 \))

\[ \iff (q[j] + k \cdot r^{c[j]-1}) \cdot (d[j] + r^{b[j]} - 1) \leq x[j] \] Inequality 1

2. \[ \frac{x[j] + r^{a[j]-1}}{d[j]} < q[j] + (k + 1) \cdot r^{c[j]-1} \] (Solve it for \( k = 0, 1, ..., r - 1 \))

\[ \iff x[j] + r^{a[j]} - 1 < d[j] \cdot (q[j] + (k + 1) \cdot r^{c[j]-1}) \] Inequality 2

• What if \( x \) and \( d \) are offline (fully given at time 0)?
  – Then, \( a[j] = b[j] = 0 \) for all \( j \) and \( c[j] = n - j \).

  1. \( (q[j] + k \cdot r^{n-j-1}) \cdot d[j] \leq x[j] \)
  2. \( x[j] < d[j] \cdot (q[j] + (k + 1) \cdot r^{n-j-1}) \)

\[ \Rightarrow k \cdot d \leq r^{j+1-n} \cdot (x - d \cdot q[j]) < (k + 1) \cdot d \]

Radix-\( r \) digit-recurrence division

In this case, the online division algorithm is equivalent to the offline division algorithm. (dual-purpose)
Online Division: Interval Analysis

• Example (decimal numbers)
  – $x$ (dividend): 1.8036
  – $d$ (divisor): 1.2631
  – $q$ (quotient): $x.xxxx$

• Assumption
  – One digits of $x$ and $d$ are given every cycle.
  – We try to obtain one digit of the quotient.

• Division
  – $x$: 1.8xxx, $d$: 1.2xxx  $\frac{x}{d}_{\text{MIN}} = \frac{1.8000}{1.2999} \leq \frac{x}{d} = \frac{1.8999}{1.2000} \leq 1.58 \ldots \rightarrow q = 1.xxxxx$
  – $x$: 1.80xx, $d$: 1.26xx  $\frac{x}{d}_{\text{MIN}} = \frac{1.8000}{1.2699} \leq \frac{x}{d} = \frac{1.8099}{1.2600} \leq 1.43 \ldots \rightarrow q = 1.4xxxx$
  – $x$: 1.803x, $d$: 1263x  $\frac{x}{d}_{\text{MIN}} = \frac{1.8030}{1.2639} \leq \frac{x}{d} = \frac{1.8039}{1.2630} \leq 1.428 \ldots \rightarrow q = 1.42xx$
  – $x$: 1.8036, $d$: 1.2631  $\frac{x}{d}_{\text{MIN}} = \frac{1.8036}{1.2631} \leq \frac{x}{d} = \frac{1.8036}{1.2631} \leq 1.4279 \ldots \rightarrow q = 1.427x$
  – Find the next quotient digit. $\frac{x}{d}_{\text{MIN}} = \frac{1.8036}{1.2631} \leq \frac{x}{d} = \frac{1.8036}{1.2631} \leq 1.4279 \ldots \rightarrow q = 1.4279$
Online Division: Interval Analysis

- Implementation example (binary, radix-4)
  - given
    - $x$ (dividend): 1.110101101101xxxx
    - $d$ (divisor): 1.011011110001xxxx
  - we have obtained
    - $q$ (quotient): 1.0100xxxxxxxxxx

- Solve

<table>
<thead>
<tr>
<th>$k$</th>
<th>Inequality 1</th>
<th>Inequality 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$(q + 0 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (True)</td>
<td>$x + 4^{-6} - 4^{-8} &lt; d \cdot (q + 1 \cdot 4^{-(2+1)})$ (False)</td>
</tr>
<tr>
<td>1</td>
<td>$(q + 1 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (True)</td>
<td>$x + 4^{-6} - 4^{-8} &lt; d \cdot (q + 2 \cdot 4^{-(2+1)})$ (False)</td>
</tr>
<tr>
<td>2</td>
<td>$(q + 2 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (True)</td>
<td>$x + 4^{-6} - 4^{-8} &lt; d \cdot (q + 3 \cdot 4^{-(2+1)})$ (True)</td>
</tr>
<tr>
<td>3</td>
<td>$(q + 3 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (False)</td>
<td>$x + 4^{-6} - 4^{-8} &lt; d \cdot (q + 4 \cdot 4^{-(2+1)})$ (True)</td>
</tr>
</tbody>
</table>

- Thus, $q_5q_6 = k = 2$ (10) $\rightarrow q=1.010010xxxxxxx$
Online Division: Interval Analysis

1. Inequality 1

\[(q[j] + k \cdot r^{-(c[j]+1)}) \cdot (d[j] + r^{-b[j]} - ulp) \leq x[j]\]

\[\Leftrightarrow q[j] \cdot d[j] + q[j] \cdot r^{-b[j]} - q[j] \cdot ulp + k \cdot d[j] \cdot r^{-(c[j]+1)} + k \cdot d[j] \cdot r^{-(b[j]+c[j]+1)} - k \cdot r^{-(c[j]+1)} \cdot ulp \leq x[j]\]

\[\text{shift } q[j] \quad \text{shift } d[j] \quad \text{shift } r^{-b[j]} \quad \text{shift } ulp \quad \text{shift } k \cdot d[j] \quad \text{shift } r^{-(c[j]+1)} \quad \text{shift } k \cdot r^{-b[j]+c[j]+1} \quad \text{shift } k \cdot r^{-(c[j]+1)} \cdot ulp \]

2. Inequality 2

\[x[j] + r^{-a[j]} - ulp < d[j] \cdot (q[j] + (k + 1) \cdot r^{-(c[j]+1)})\]

\[\Leftrightarrow x[j] + r^{-a[j]} - ulp < q[j] \cdot d[j] + (k + 1) \cdot d[j] \cdot r^{-(c[j]+1)}\]

\[\text{shift } q[j] \quad \text{shift } d[j] \quad \text{shift } r^{-a[j]} \quad \text{shift } ulp \quad \text{shift } (k + 1) \cdot d[j] \quad \text{shift } r^{-(c[j]+1)} \]

\[\bullet m[g, h] = (d_0 \cdot d_1 \ldots d_g 0 \ldots 0) \cdot (q_0 \cdot q_1 \ldots q_h 0 \ldots 0)\]

- We can incrementally update \(m[g, h]\).
Online Division: Interval Analysis

• Incremental update of $m[g, h] = (d_{n-1} \ldots d_{n-g}0 \ldots 0) \cdot (q_{n-1} \ldots q_{n-h}0 \ldots 0)$.
  - Suppose we have obtained $m[g, h]$.

    $\begin{array}{c|c|c}
    n-1 & n-g & \hline \\
    d & d_{old} & \hline \\
    q & q_{old} & n-h \\
    \end{array}$

  - A few more digits are given ($d$: w more digits. $q$: p more digits).

    $\begin{array}{c|c|c|c}
    n-1 & n-g & w \text{ digits} & \hline \\
    d & d_{old} & d_{new} & \hline \\
    q & q_{old} & q_{new} & n-h \\
    \end{array}$

    $\begin{array}{c|c|c|c}
    n-1 & n-g & w \text{ digits} & \hline \\
    d & d_{old} & d_{new} & \hline \\
    q & q_{old} & q_{new} & n-h \\
    \end{array}$

  - $m[g + w, h + p] = m[g, h] \cdots \begin{array}{c}
  1 \end{array} d_{old} \cdot q_{old}$
    
    If $w$ and $p$ are small, this update can be done in a cycle.

    ➔ Use $w$, $p$: 2 and 4bit
Online Division: Interval Analysis

- When do we perform the incremental update of \( m = d \cdot q \)?
  1. When new digits of \( d \) are given.
  2. When we find new digits of \( q \).

- Flowchart

We have \( m[b[j], c[j]] \).

Is \( d_{b[j]+1:b[j]+w} \) valid?

- Yes
  - \( m = m + 2 \)
  - Find a quotient digit by the two inequalities.
  - Found?
    - Yes
      - \( m = m + 3 + 4 \)
    - No
      - \( m \)

- No
  - \( m = m + 2 \)

Four cases

<table>
<thead>
<tr>
<th>New valid digits of ( d )</th>
<th>Found new Digit of ( q )</th>
<th>Updated ( m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>( m[b[j], c[j]] )</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>( m[b[j], c[j] + 1] )</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>( m[b[j] + w, c[j]] )</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>( m[b[j] + w, c[j] + 1] )</td>
</tr>
</tbody>
</table>
Hardware Architecture

- **Input**
  - Dividend $x$ and its valid bits
    - $x = x_0 \cdot x_1 \ldots x_n$ ($x_0$ is the hidden 1)
    - $v_x = v_{x_0}v_{x_1} \ldots v_{x_n}$ ($11\ldots100\ldots0$)
      - $v_{x_i} = 1$ (or 0): $x_i$ is valid (invalid)
      - The generator of $x$ has to generate $v_x$ too.
  - Divisor $d$ and its valid bits
    - $d = d_0 \cdot d_1 \ldots d_n$ ($d_0$ is the hidden 1)
    - $v_d = v_{d_0}v_{d_1} \ldots v_{d_n}$

- **Output**
  - Quotient $q$ and its valid bits
    - $q = q_{n-1}q_{n-2} \ldots q_0$
    - $v_q = v_{q_{n-1}}v_{q_{n-2}} \ldots v_{q_0}$ (we generate)

- **Design parameters**
  - $r$: Radix-$r$ division. Try to obtain $\log_2 r$ bits per cycle.
  - $w$: # unused bits of $d$ (divisor) to update $m = d[j] \cdot q[j]$ ($w = 2$ or 4 bits)
Hardware Architecture

For example $k = 6 \ (0110)$, we add $k \cdot d[j] \cdot r^{-(c[j]+1)}$ by adding (by CSAs)

$$d[j] \cdot r^{-(c[j]+1)} \ll 1$$
$$d[j] \cdot r^{-(c[j]+1)} \ll 2$$

Pre-computed $m$ assuming $q_{c[j]+1} = k$

Inequality 1

$\begin{align*}
q[j] \cdot d[j] + q[j] \cdot r^{-b[j]} - q[j] \cdot ulp \\
+ k \cdot d[j] \cdot r^{-(c[j]+1)}
\end{align*}$

$\begin{align*}
+ k \cdot d[j] \cdot r^{-(b[j]+c[j]+1)} - k \cdot r^{-(c[j]+1)} \cdot ulp \\
\leq x[j]
\end{align*}$

Inequality 2

$\begin{align*}
x[j] + r^{-a[j]} - ulp \\
< q[j] \cdot d[j] + (k + 1) \cdot d[j] \cdot r^{-(c[j]+1)}
\end{align*}$
Simulation Setup

• **Design**
  – 64bit divider (64bit dividend & divisor)

• **Implementation**
  – Verilog

• **Synthesis**
  – Synopsys Design Compiler
  – 22nm standard cell library

• **Performance metrics**
  – Clock period, Area, Energy
  – Execution time
## Design Characteristics

<table>
<thead>
<tr>
<th>Design</th>
<th>Type</th>
<th>Radix</th>
<th>#divisor bits processed / cycle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN16</td>
<td>Offline</td>
<td>8</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SA17</td>
<td>Offline</td>
<td>16</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>JB20</td>
<td>Offline</td>
<td>64</td>
<td>-</td>
<td>ARM</td>
</tr>
<tr>
<td>AT03</td>
<td>Online</td>
<td>4</td>
<td>2</td>
<td>The latest on-line divider</td>
</tr>
<tr>
<td>FF2</td>
<td>Offline</td>
<td>4</td>
<td>-</td>
<td>Floating-point offline (Interval-analysis-based, not partial-remainder-based)</td>
</tr>
<tr>
<td>FF4</td>
<td>Offline</td>
<td>16</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>FN22</td>
<td>Online</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>FN24</td>
<td>Online</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FN42</td>
<td>Online</td>
<td>16</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>FN44</td>
<td>Online</td>
<td>16</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Our Designs

• Offline
  – FF2: Radix-4 ("2" in “FF2” means that we obtain 2 quotient bits per cycle)
  – FF4: Radix-16 (we obtain 4 quotient bits per cycle)

• Online
  – FNqd
    • q: # quotient bits to try to obtain per cycle
    • d: # divisor bits that are used to update $m$ in a cycle
  – FN22, FN24: Radix-4
  – FN42, FN44: Radix-16
Simulation Results

- $T$: # bits of $x$ and $d$ given per cycle.
- Results: normalized to the FF2 offline divider.

(a) Online: $T = 2$
(b) Online: $T = 4$
(c) Offline: $T = 52$

- Offline dividers: The execution time goes down (gets better) as $T$ goes up.
- AT03, FN22, FN24: The execution time goes down as $T$ goes up to $T = 2$ and then saturates.
- FN42, FN44: The execution time goes down as $T$ goes up to $T = 4$, then saturates.
Simulation Results: Analysis

- \( T \): # bits of \( x \) and \( d \) given per cycle.
- Offline dividers: The execution time goes down as \( T \) goes up. Why?
  - Execution time: Wait time (■) + computation time (■).
  - As \( T \) goes up, the wait time goes down.
    - \( T = 2 \) (online):
    - \( T = 4 \) (online):
    - \( T = 52 \) (offline):
- Radix-4 online dividers (AT03, FN22, FN24): saturates when \( T = 2 \).
  - If \( T < 2 \): Need more bits to find a quotient digit per cycle.
  - If \( T > 2 \): Do not process more than two bits.
- Radix-16 online dividers (FN42, FN44): saturates when \( T = 4 \).
  - If \( T < 4 \): Need more bits to find a quotient digit per cycle.
  - If \( T > 4 \): Do not process more than four bits.
Discussion

• Integration of both offline and online dividers in a single architecture

- Carry-save addition (CSA)
- Carry-propagate addition (CPA)

For $k = 0, 1, \ldots, r-1$

Inequalities are true/false?

$\{ f_0, f_{r-1}, m_0, m_{r-1}, \ldots \}

Pre-computed $m$

assuming $q_{c[j]+1} = k$

(OF course, this can be applied to the integer dividers too!)
Conclusion

• We proposed the interval-analysis-based division algorithm, which can be used for both offline and online division.
  – Radix-4
  – Radix-16
  – Uses the normal binary number system.

• Performance (offline division)
  – Online dividers are slower than offline dividers.
    • A small modification of our online dividers could improve the performance of the FN designs significantly (close to the FF2 and FF4 designs).

• Performance (online division)
  – Online dividers outperform offline dividers.
  – Lower bit rate: AT03 outperforms our dividers.
  – Higher bit rate: Our dividers outperforms AT03.