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)

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

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

Motivation

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

$$\boxed{a \cdot \sqrt{b \cdot c} \cdot \cos(d-e)} \longrightarrow$$

$$i_1 : x_1 = b \cdot c$$

(mult-sqrt)

$$i_2 : x_2 = \sqrt{x_1}$$

(sqrt-mult)

$$i_3 : x_3 = x_2 \cdot a \qquad i_4 : y = \cos(d-e)$$

(mult-mult)

(cosin-mult)

$$i_5 : x_4 = x_3 \cdot y$$

Motivation

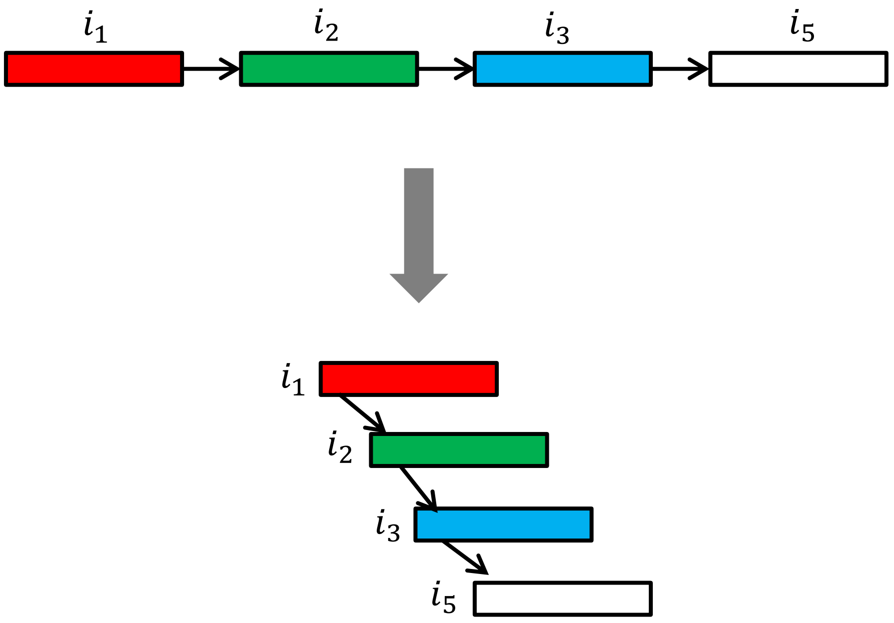
- Resolving dependency chain : $i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow i_5$

$i_1 : x_1 = b \cdot c$

$i_2 : x_2 = \sqrt{x_1}$

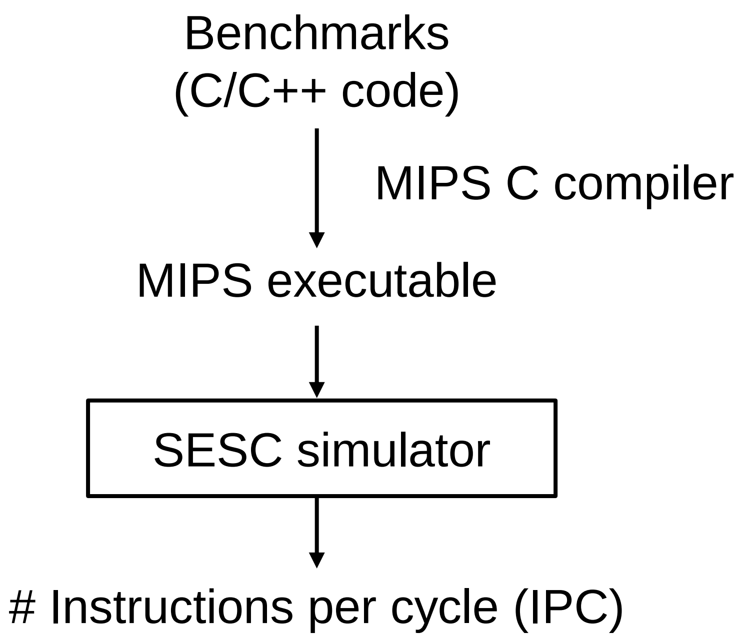
$i_3 : x_3 = x_2 \cdot a$ $\quad$ $i_4 : \mathrm{y} = \cos(d - e)$

$i_5 : x_4 = x_3 \cdot y$



Architecture Simulation

- Simulator
  - SESC (cycle-accurate MIPS32 simulator)



Architecture Simulation

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

- # clock cycles

	# Cycles
Integer ALU	1
Integer Mult	4
Integer Div	15
FP Add/Sub	4
FP Mult	6
FP Div	16

Architecture Simulation

- Profiling results (# instructions)

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

	# Cycles
Integer ALU	1
Integer Mult	4
Integer Div	15
FP ALU	4
FP Mult	6
FP Div	16

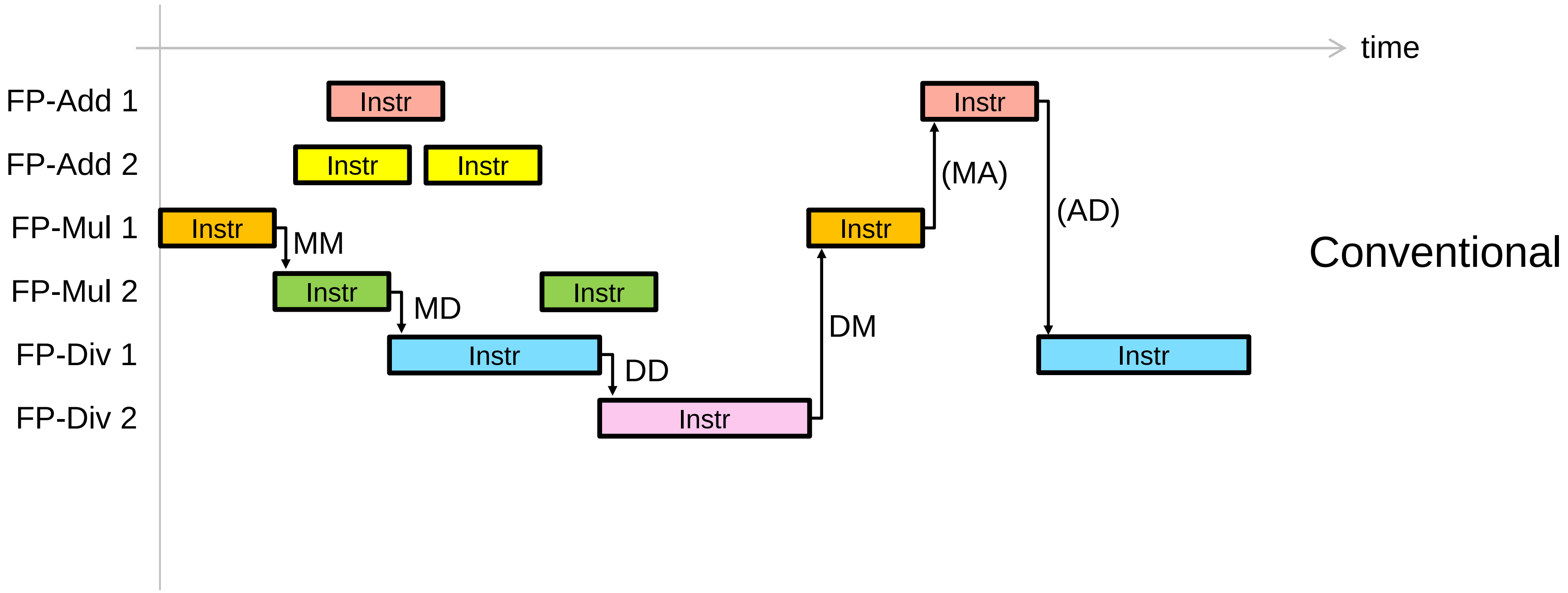
B1 : Finite element method (FEM)
B2 : EM simulation
B3 : Statistical inference
B4 : Bessel function

Architecture Simulation

- 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

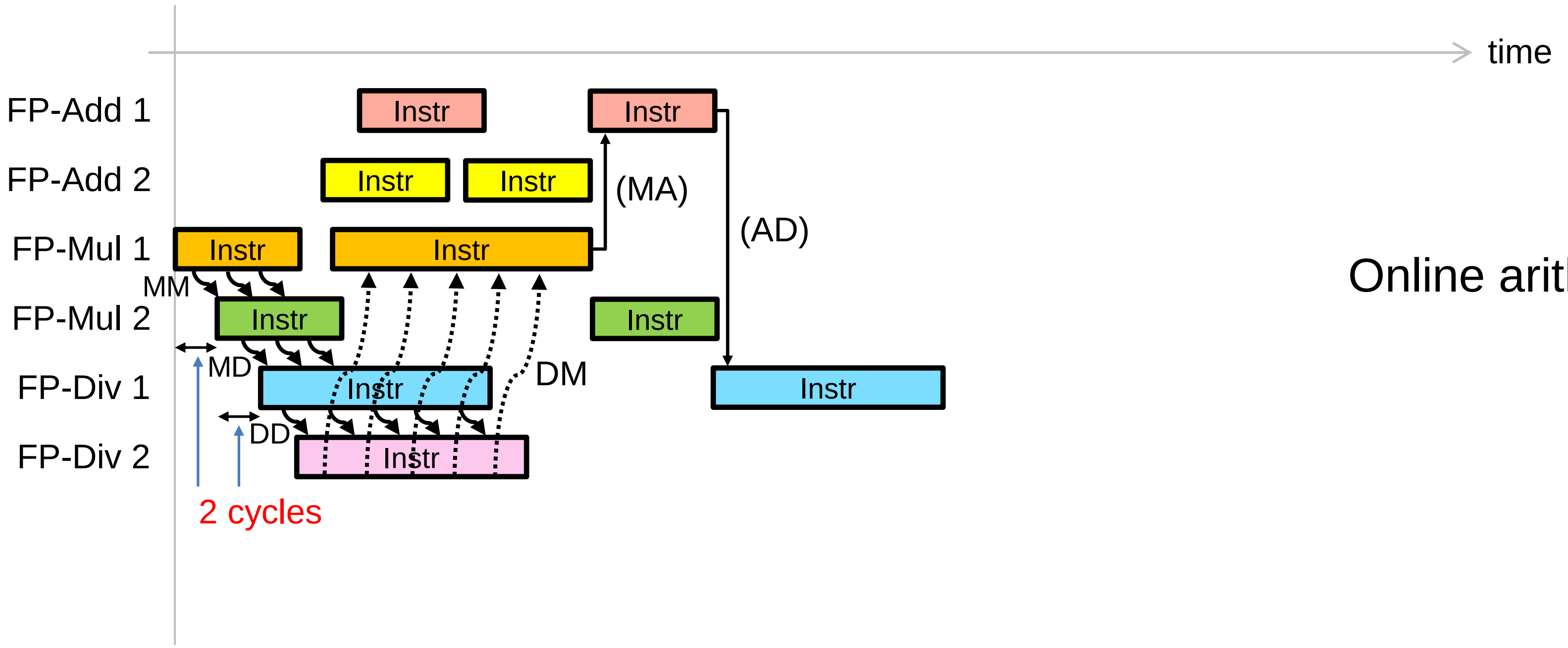
- Resolve dependencies



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

Architecture Simulation

- 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.



Online arithmetic

* 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 $c \times \left(\frac{a}{b}\right)$	$\frac{\left(\frac{a}{b}\right)}{c}$ or $\frac{c}{\left(\frac{a}{b}\right)}$	
	MM (M→M)	**MD (M→D)**	**DM (D→M)**	**DD (D→D)**	**Total (%)**
B1	0	0.58	3.85	0.36	4.79
B2	5.71	0	0	0	5.71
B3	0.56	0.25	0.33	0.43	1.57
B4	23.61	0	1.47	2.96	28.04

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
    - $q[j] = \underbrace{q_0.q_1q_2\ldots q_j}_{\text{obtained}}\underbrace{(\mathrm{xx}\ldots\mathrm{x})}_{\text{to be obtained}} = \sum_{i=0}^{j} q_i \cdot r^{-i}$

- Digit-recurrence division
  - Division: $x = d \cdot q + rem$
  - Condition: $|rem| < |d| \cdot ulp$
  - At j-th iteration

$$rem[j] = x - d \cdot q[j]$$

$$\begin{aligned} & 0 \le rem[j] < d \cdot r^{-j} \\ \Leftrightarrow\ & 0 \le x - d \cdot q[j] < d \cdot r^{-j} \\ \Leftrightarrow\ & 0 \le \underbrace{r^j \cdot (x - d \cdot q[j])}_{w[j]} < d \end{aligned}$$

Offline Division: Digit-Recurrence (Review)

- Digit-recurrence division
  - At j-th iteration

$$0 \leq \underbrace{r^j \cdot (x - d \cdot q[j])}_{w[j]} < d$$

  - At $(j+1)$-th iteration

$$q[j+1] = \underbrace{q_0 . q_1 \dots q_j}_{\text{obtained}} \underbrace{q_{j+1}}_{\text{Find}} (\mathrm{xx} \dots \mathrm{x})$$

$$0 \leq w[j+1] < d$$

$$\Leftrightarrow 0 \leq r \cdot w[j] - d \cdot q_{j+1} < d$$

  - $q_{j+1} = k$ if

$$0 \leq r \cdot w[j] - d \cdot k < d$$

$$\Leftrightarrow \boxed{k \cdot d \leq r \cdot w[j] < (k+1) \cdot d}$$

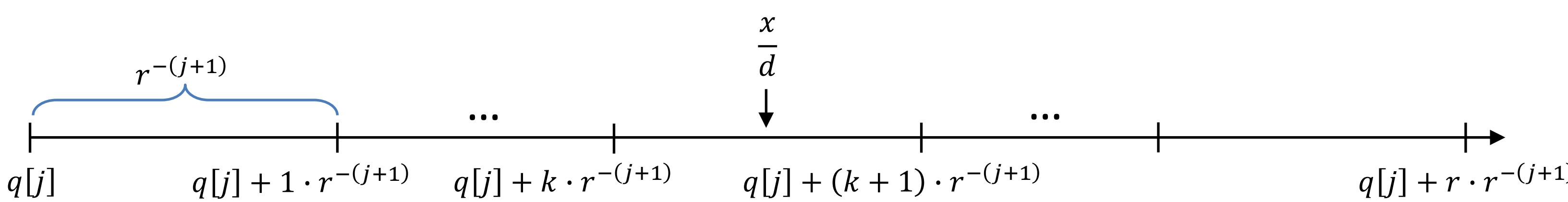
Radix-r digit-recurrence division

Offline Division: Interval Analysis

- Interval-analysis-based division
  - Quotient obtained down to the j-th digit
    - $q[j] = \underbrace{q_0.q_1q_2 \dots q_j}_{\text{obtained}}\underbrace{(\mathrm{xx}\dots\mathrm{x})}_{\text{to be obtained}} = \sum_{i=0}^{j} q_i \cdot r^{-i}$

$$q[j+1] = \underbrace{q_0.q_1 \dots q_j}_{\text{obtained}}\underset{\text{Find}}{\underline{q_{j+1}}}(\mathrm{xx}\dots\mathrm{x})$$

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

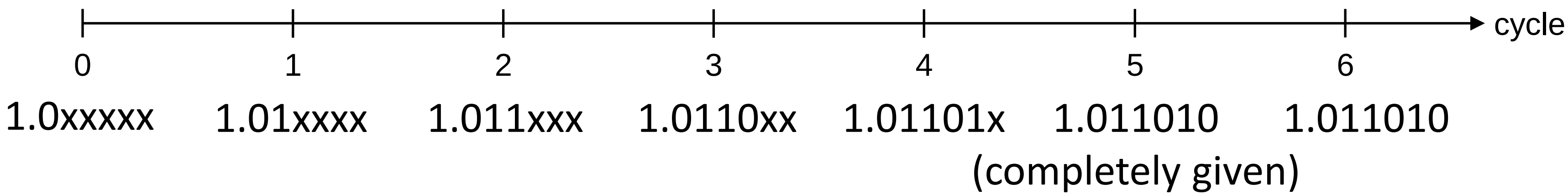


$$q[j] + k \cdot r^{-(j+1)} \le \frac{x}{d} < q[j] + (k+1) \cdot r^{-(j+1)}$$

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

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: $x[j] = \underbrace{x_0 . x_1 \dots x_{a[j]}}_{\text{given}} \underbrace{(\mathrm{xx} \dots \mathrm{x})}_{\text{to be given}} = \sum_{i=0}^{a[j]} x_i \cdot r^{-i}$
  - Divisor: $d[j] = \underbrace{d_0 . d_1 \dots d_{b[j]}}_{\text{given}} \underbrace{(\mathrm{xx} \dots \mathrm{x})}_{\text{to be given}} = \sum_{i=0}^{b[j]} d_i \cdot r^{-i}$
- Output (obtained until the j-th iteration)
  - Quotient: $q[j] = \underbrace{q_0 . q_1 \dots q_{c[j]}}_{\text{obtained}} \underbrace{(\mathrm{xx} \dots \mathrm{x})}_{\text{to be obtained}} = \sum_{i=0}^{c[j]} q_i \cdot r^{-i}$

Online Division

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

cycle	0	1	2	3	4	5	6
x	1.1xxxxx	1.11xxxx	1.110xxx	1.1101xx	1.11010x	1.110101 (fully given)	1.110101
d	1.0xxxxx	1.00xxxx	1.001xxx	1.0011xx	1.00111x	1.001110 (fully given)	1.001110

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

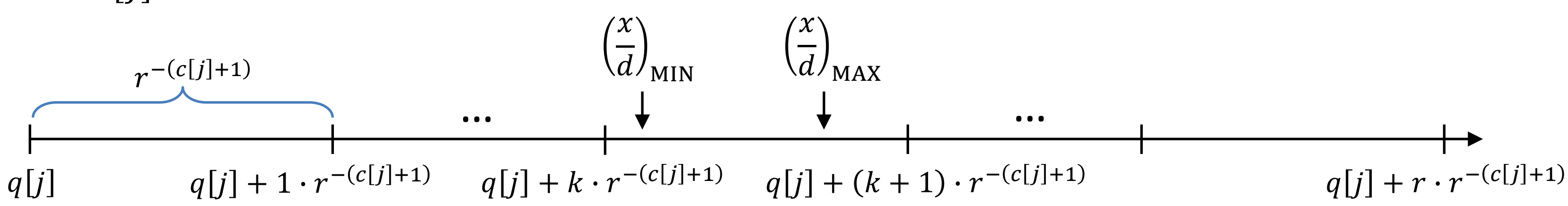
cycle	0	1	2	3	4	5	6
x	1.110xxx	1.110xxx	1.110xxx	1.110101 (fully given)	1.110101	1.110101	1.110101
d	1.0xxxxx	1.00111x	1.00111x	1.00111x	1.00111x	1.001110 (fully given)	1.001110

Online Division: Interval Analysis

- Given
  - $x[j] = x_0.x_1 \dots x_{a[j]}(\mathrm{xx} \dots \mathrm{x})$
    - $x_{\mathrm{MIN}} = x_0.x_1 \dots x_{a[j]}0 \dots 0$ (padded 0’s)
    - $x_{\mathrm{MAX}} = x_0.x_1 \dots x_{a[j]}1 \dots 1$ (padded 1’s)
  - $d[j] = d_0.d_1 \dots d_{b[j]}(\mathrm{xx} \dots \mathrm{x})$
    - $d_{\mathrm{MIN}} = d_0.d_1 \dots d_{b[j]}0 \dots 0$
    - $d_{\mathrm{MAX}} = d_0.d_1 \dots d_{b[j]}1 \dots 1$
- Range of $\frac{x}{d} = \left[\left(\frac{x}{d}\right)_{\mathrm{MIN}}, \left(\frac{x}{d}\right)_{\mathrm{MAX}}\right]$
  - $\left(\frac{x}{d}\right)_{\mathrm{MIN}} = \frac{x_{\mathrm{MIN}}}{d_{\mathrm{MAX}}} = \frac{x_0.x_1 \dots x_{a[j]}0 \dots 0}{d_0.d_1 \dots d_{b[j]}1 \dots 1}$
  - $\left(\frac{x}{d}\right)_{\mathrm{MAX}} = \frac{x_{\mathrm{MAX}}}{d_{\mathrm{MIN}}} = \frac{x_0.x_1 \dots x_{a[j]}1 \dots 1}{d_0.d_1 \dots d_{b[j]}0 \dots 0}$

Online Division: Interval Analysis

- Obtained
  - $q[j] = q_0.q_1 \dots q_{c[j]}(\underline{\mathrm{x}}\mathrm{x} \dots \mathrm{x})$
- Find $q_{c[j]+1}$
  - $q_{c[j]+1} = k$ if

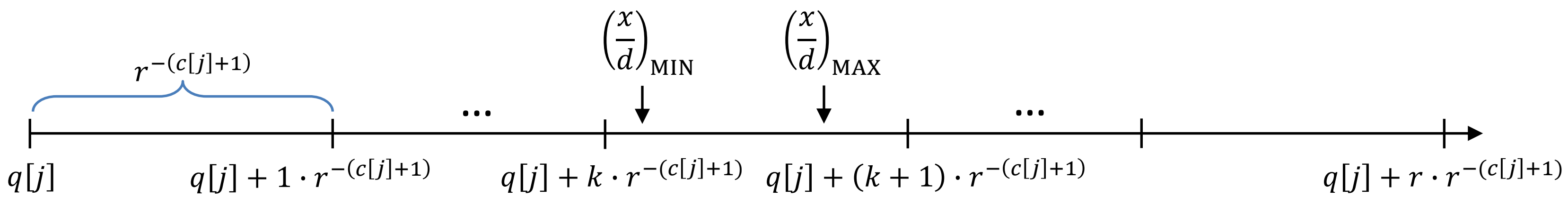


  - This just means
    - $q_0.q_1 \dots q_{c[j]}k0 \dots 0 \le \frac{x}{d} < q_0.q_1 \dots q_{c[j]}(k+1)0 \dots 0$
    - which means $q[j] = q_0.q_1 \dots q_{c[j]}\boldsymbol{k} \dots$
  - For example, for $r = 4$
    - $\underbrace{1.321}_{\text{obtained}}\mathbf{2} \dots \le \frac{x}{d} < \underbrace{1.321}_{\text{obtained}}\mathbf{3} \dots \Rightarrow \frac{x}{d} = 0321\mathbf{2} \dots$

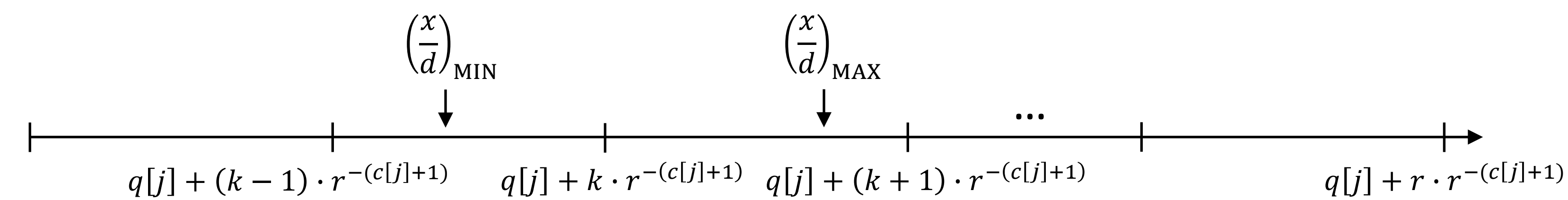
$q_{-(c[j]+1)} = 2$

Online Division: Interval Analysis

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



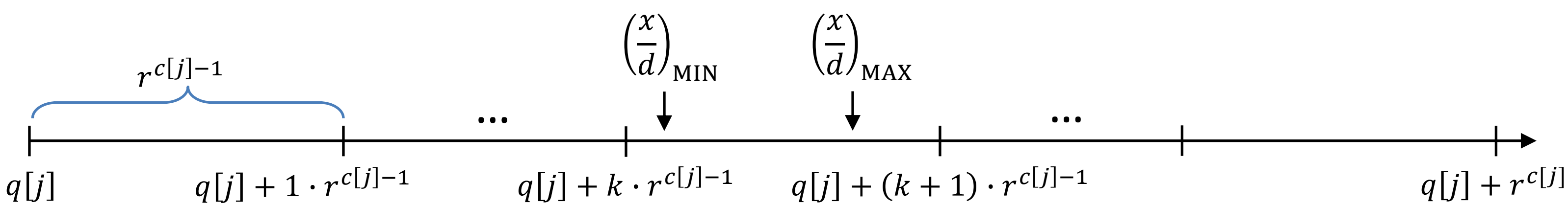
- Case 2: $q_{c[j]+1} =?$



  - 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



Ineq#1) $\mathrm{q[j]} + \mathrm{k} \cdot \mathrm{r}^{\mathrm{c[j]}-1} \leq \left(\frac{\mathrm{x}}{\mathrm{d}}\right)_{\mathrm{MIN}}$

$\rightarrow$ $\mathrm{q[j]} + \mathrm{k} \cdot \mathrm{r}^{\mathrm{c[j]}-1} \leq \frac{\mathrm{x}_{\mathrm{n}-1}\mathrm{x}_{\mathrm{n}-2}\ldots\mathrm{x}_{\mathrm{a[j]}}0\ldots0}{\mathrm{d}_{\mathrm{n}-1}\mathrm{d}_{\mathrm{n}-2}\ldots\mathrm{d}_{\mathrm{b[j]}}1\ldots1} = \frac{\mathrm{x[j]}}{\mathrm{d[j]}+\mathrm{r}^{\mathrm{b[j]}}-1}$

Ineq#2) $\left(\frac{\mathrm{x}}{\mathrm{d}}\right)_{\mathrm{MAX}} < \mathrm{q[j]} + (\mathrm{k}+1) \cdot \mathrm{r}^{\mathrm{c[j]}-1}$

$\rightarrow$ $\frac{\mathrm{x}_{\mathrm{n}-1}\mathrm{x}_{\mathrm{n}-2}\ldots\mathrm{x}_{\mathrm{a[j]}}1\ldots1}{\mathrm{d}_{\mathrm{n}-1}\mathrm{d}_{\mathrm{n}-2}\ldots\mathrm{d}_{\mathrm{b[j]}}0\ldots0} = \frac{\mathrm{x[j]}+\mathrm{r}^{\mathrm{a[j]}}-1}{\mathrm{d[j]}} < \mathrm{q[j]} + (\mathrm{k}+1) \cdot \mathrm{r}^{\mathrm{c[j]}-1}$

Online Division: Interval Analysis

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

$\Leftrightarrow \left(q[j] + k \cdot r^{c[j]-1}\right) \cdot \left(d[j] + r^{b[j]} - 1\right) \le x[j]$ Inequality 1

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

$\Leftrightarrow x[j] + r^{a[j]} - 1 < d[j] \cdot \left(q[j] + (k+1) \cdot r^{c[j]-1}\right)$ 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. $\left(q[j] + k \cdot r^{n-j-1}\right) \cdot d[j] \le x[j]$
2. $x[j] < d[j] \cdot \left(q[j] + (k+1) \cdot r^{n-j-1}\right)$

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

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 $1.38\ldots \le \left(\frac{x}{d}\right)_{MIN} = \frac{1.8000}{1.2999} \le \left(\frac{x}{d}\right)_{MAX} = \frac{1.8999}{1.2000} \le 1.58\ldots \rightarrow q$ = 1.xxxx
  - x: 1.80xx, d: 1.26xx $1.43\ldots \le \left(\frac{x}{d}\right)_{MIN} = \frac{1.8000}{1.2699} \le \left(\frac{x}{d}\right)_{MAX} = \frac{1.8099}{1.2600} \le 1.43\ldots \rightarrow q$ = 1.4xxx
  - x: 1.803x, d: 1263x $1.426\ldots \le \left(\frac{x}{d}\right)_{MIN} = \frac{1.8030}{1.2639} \le \left(\frac{x}{d}\right)_{MAX} = \frac{1.8039}{1.2630} \le 1.428\ldots \rightarrow q$ = 1.42xx
  - x: 1.8036, d: 1.2631 $1.4279 \le \left(\frac{x}{d}\right)_{MIN} = \frac{1.8036}{1.2631} \le \left(\frac{x}{d}\right)_{MAX} = \frac{1.8036}{1.2631} \le 1.4279\ldots \rightarrow q$ = 1.427x
  - Find the next quotient digit. $1.4279 \le \left(\frac{x}{d}\right)_{MIN} = \frac{1.8036}{1.2631} \le \left(\frac{x}{d}\right)_{MAX} = \frac{1.8036}{1.2631} \le 1.4279\ldots \rightarrow q$ = 1.4279

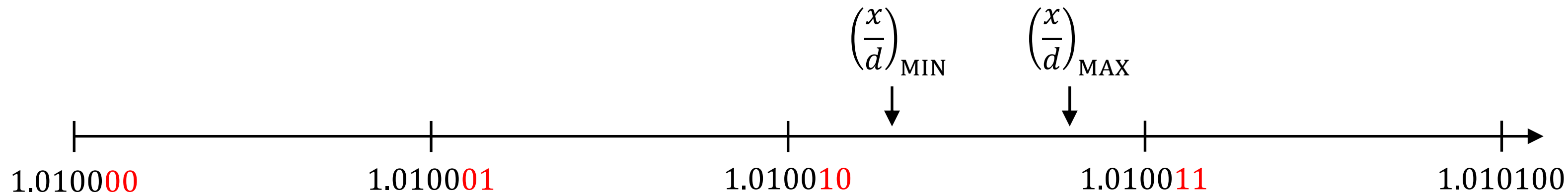
Online Division: Interval Analysis

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

      Find
- Solve

k	Inequality 1	Inequality 2
0	$(q + 0 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (True)	$x + 4^{-6} - 4^{-8} < d \cdot (q + 1 \cdot 4^{-(2+1)})$ (False)
1	$(q + 1 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (True)	$x + 4^{-6} - 4^{-8} < d \cdot (q + 2 \cdot 4^{-(2+1)})$ (False)
2	$(q + 2 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (True)	$x + 4^{-6} - 4^{-8} < d \cdot (q + 3 \cdot 4^{-(2+1)})$ (True)
3	$(q + 3 \cdot 4^{-(2+1)}) \cdot (d + 4^{-6} - 4^{-8}) \leq x$ (False)	$x + 4^{-6} - 4^{-8} < d \cdot (q + 4 \cdot 4^{-(2+1)})$ (True)

  - Thus, $q_5 q_6 = k = 2\ (10) \rightarrow$ q=1.010010xxxxxxxxxx



Online Division: Interval Analysis

1. Inequality 1

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

$$\Leftrightarrow \underbrace{q[j] \cdot d[j]}_{m} + \underbrace{q[j] \cdot r^{-b[j]}}_{\text{shift } q[j]} - \underbrace{q[j] \cdot ulp}_{\text{shift } q[j]} + \underbrace{k \cdot d[j] \cdot r^{-(c[j]+1)}}_{\text{shift } k \cdot d[j]} + \underbrace{k \cdot d[j] \cdot r^{-(b[j]+c[j]+1)}}_{\text{shift } k \cdot d[j]} - \underbrace{k \cdot r^{-(c[j]+1)} \cdot ulp}_{\text{shift } k} \le x[j]$$

2. Inequality 2

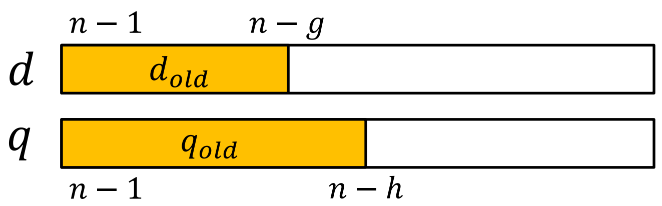
$$x[j] + r^{-a[j]} - ulp < d[j] \cdot \left(q[j] + (k+1) \cdot r^{-(c[j]+1)}\right)$$

$$\Leftrightarrow x[j] + r^{-a[j]} - ulp < \underbrace{q[j] \cdot d[j]}_{m} + \underbrace{(k+1) \cdot d[j] \cdot r^{-(c[j]+1)}}_{\text{shift } (k+1) \cdot d[j]}$$

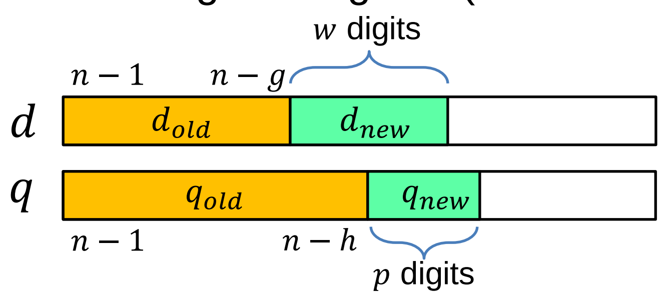
- $m[g,h] = \left(d_0.d_1 \ldots d_g 0 \ldots 0\right) \cdot \left(q_0.q_1 \ldots q_h 0 \ldots 0\right)$
  - We can incrementally update $m[g,h]$.

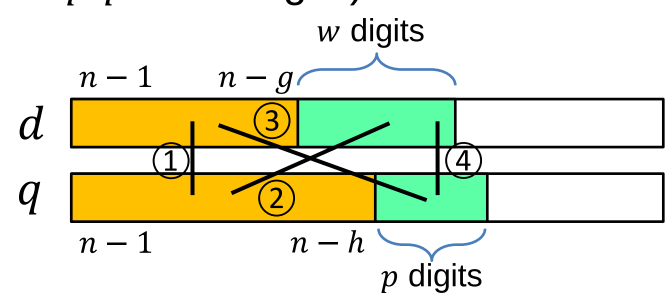
Online Division: Interval Analysis

- 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 $m[g,h]$.



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





  - $m[g+w, h+p] = m[g,h] \cdots ①\ d_{old} * q_{old}$

$$+\left(d_{n-g-1:n-g-w} \cdot r^{n-g-w} \cdot q_{n-1:n-h} \cdot r^{n-h}\right) \cdots ②\ d_{new} * q_{old}$$

$$+\left(d_{n-1:n-g} \cdot r^{n-g} \cdot q_{n-h-1:n-h-p} \cdot r^{n-h-p}\right) \cdots ③\ d_{old} * q_{new}$$

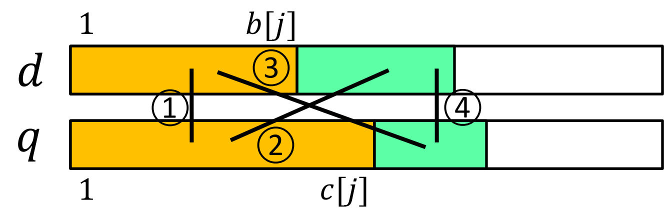
$$+\left(d_{n-g-1:n-g-w} \cdot r^{n-g-w} \cdot q_{n-h-1:n-h-p} \cdot r^{n-h-p}\right) \cdots ④\ d_{new} * q_{new}$$

If w and p are small, this update can be done in a cycle.

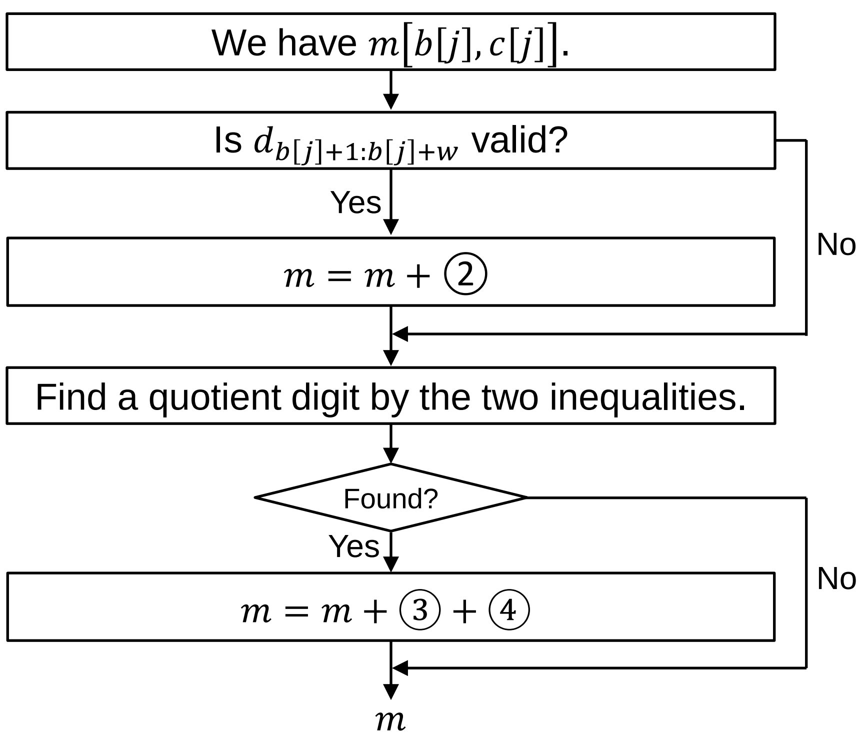
➔ Use w, p: 2 and 4bit

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



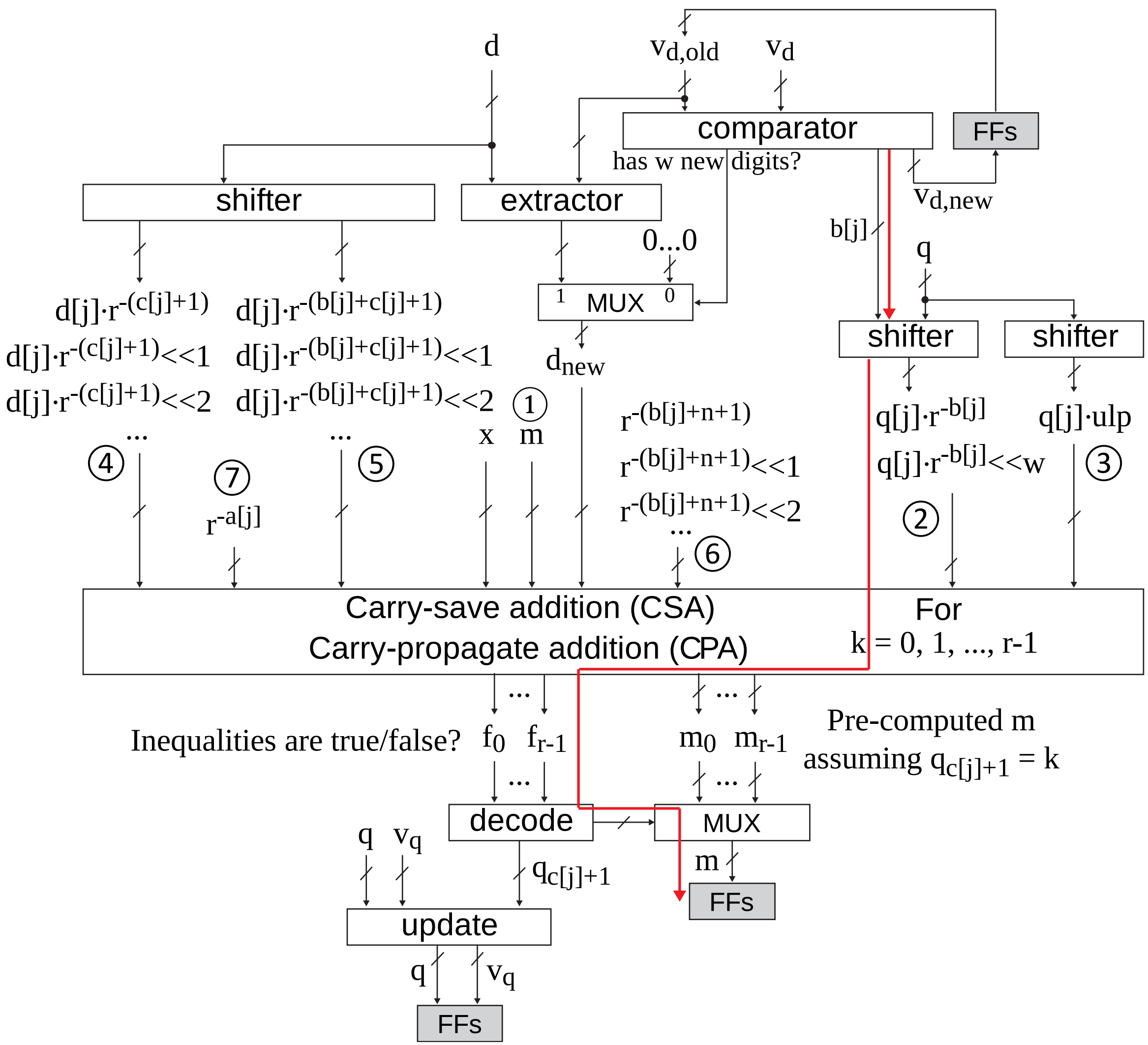
Four cases

New valid digits of d	Found new Digit of q	Updated m
No	No	$m[b[j], c[j]]$
No	Yes	$m[b[j], c[j] + 1]$
Yes	No	$m[b[j] + w, c[j]]$
Yes	Yes	$m[b[j] + \mathrm{w}, c[j] + 1]$

Hardware Architecture

- Input
  - Dividend x and its valid bits
    - $x = x_0.x_1 \dots x_n$ (x_0 is the hidden 1)
    - $v_x = v_{x_0} v_{x_1} \dots v_{x_n}$ (11…100…0)
      - $v_{x_i} = 1$ (or 0): x_i is valid (invalid)
      - The generator of x has to generate v_x too.
  - Divisor d and its valid bits
    - $d = d_0.d_1 \dots d_n$ (d_0 is the hidden 1)
    - $v_d = v_{d_0} v_{d_1} \dots v_{d_n}$
- Output
  - Quotient q and its valid bits
    - $q = q_{n-1} q_{n-2} \dots q_0$
    - $v_q = v_{q_{n-1}} v_{q_{n-2}} \dots v_{q_0}$ (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



$$\overset{(1)}{q[j]\cdot d[j]} + \overset{(2)}{q[j]\cdot r^{-b[j]}} - \overset{(3)}{q[j]\cdot ulp} + \overset{(4)}{k\cdot d[j]\cdot r^{-(c[j]+1)}} + \underset{(5)}{k\cdot d[j]\cdot r^{-(b[j]+c[j]+1)}} - \underset{(6)}{k\cdot r^{-(c[j]+1)}\cdot ulp} \le x[j]$$

Inequality 1

$$x[j] + \overset{(7)}{r^{-a[j]}} - ulp < \underset{(1)}{q[j]\cdot d[j]} + \underset{(4)}{(k+1)\cdot d[j]\cdot r^{-(c[j]+1)}}$$

Inequality 2

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

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

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

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

- AN16 : A. Nannarelli, “Performance/Power Space Exploration for Binary64 Division Units,” in IEEE Trans. on Computers, vol. 65, no. 5, May 2016, pp. 1671–1677
- SA17 : S. Amanollahi and G. Jaberipur, “Energy-Efficient VLSI Realization of Binary64 Division with Redundant Number Systems,” in IEEE Trans. on VLSI Systems, vol. 25, no. 3, Mar. 2017, pp. 954–961
- JB20 : J. D. Bruguera, “Low Latency Floating-Point Division and Square Root Unit,” in IEEE Trans. on Computers, vol. 69, no. 2, Feb. 2020, pp. 274–287
- AT03 : A. F. Tenca, A. Shantilal, and M. Sinky, “A Radix-4 On-line Division Design and Its Application to Networks of On-line Modules,” in Proceedings of SPIE, vol. 5205, 2003, pp. 529–540

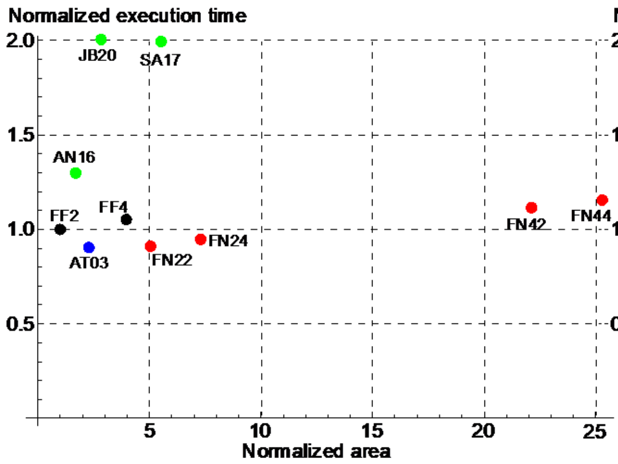
Our Designs

- 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)

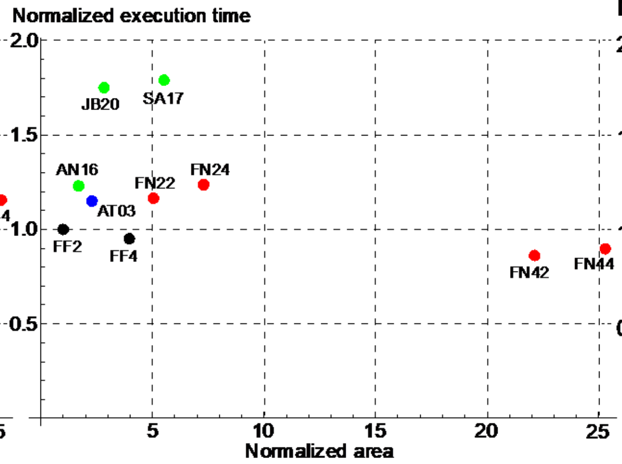
- Online
  - FNqd
    - q: # quotient bits to try to obtain per cycle
    - d: # divisor bits that are used to update m in a cycle
  - FN22, FN24: Radix-4
  - FN42, FN44: Radix-16

Simulation Results

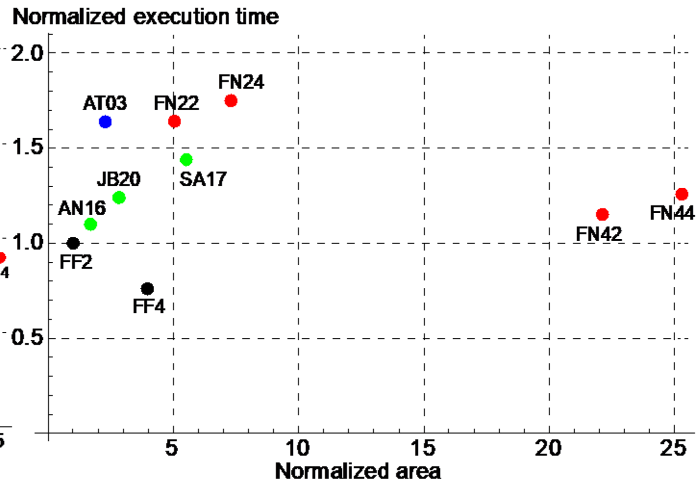
- T: # bits of x and d given per cycle.
- Results: normalized to the FF2 offline divider.



(a) Online: $T = 2$



(b) Online: $T = 4$

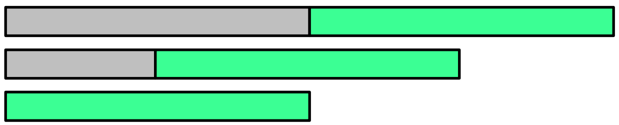


(c) Offline: $T = 52$

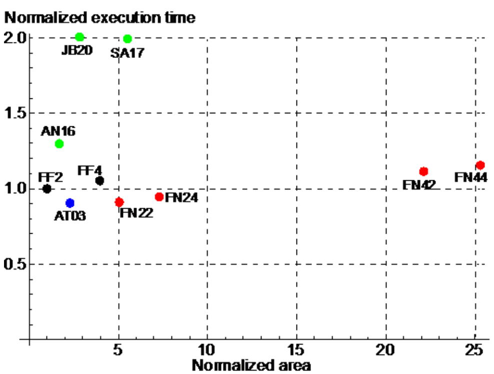
- Offline dividers: The execution time goes down (gets better) as T goes up.
- AT03, FN22, FN24: The execution time goes down as T goes up to $T = 2$ and then saturates.
- FN42, FN44: The execution time goes down as T goes up to $T = 4$, then saturates.

Simulation Results: Analysis

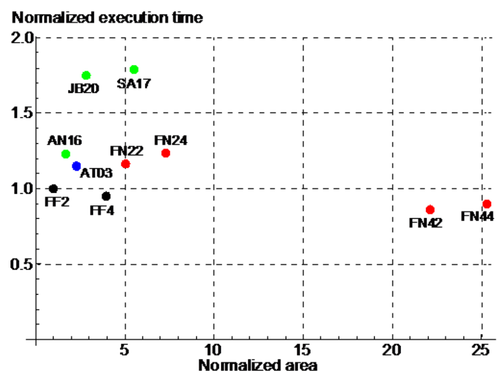
- T: # bits of x and d given per cycle.
- Offline dividers: The execution time goes down as T goes up. Why?
  - Execution time: Wait time ( ) + computation time ( ).
  - As T goes up, the wait time goes down.
    - $T = 2$ (online):
    - $T = 4$ (online):
    - $T = 52$ (offline):



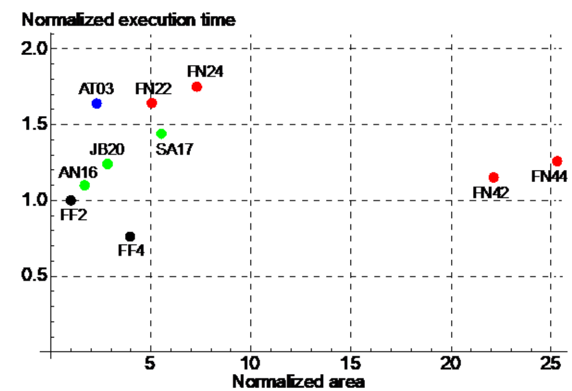
- Radix-4 online dividers (AT03, FN22, FN24): saturates when $T = 2$.
  - If $T < 2$: Need more bits to find a quotient digit per cycle.
  - If $T > 2$: Do not process more than two bits.
- Radix-16 online dividers (FN42, FN44): saturates when $T = 4$.
  - If $T < 4$: Need more bits to find a quotient digit per cycle.
  - If $T > 4$: Do not process more than four bits.



(a) Online: $T = 2$



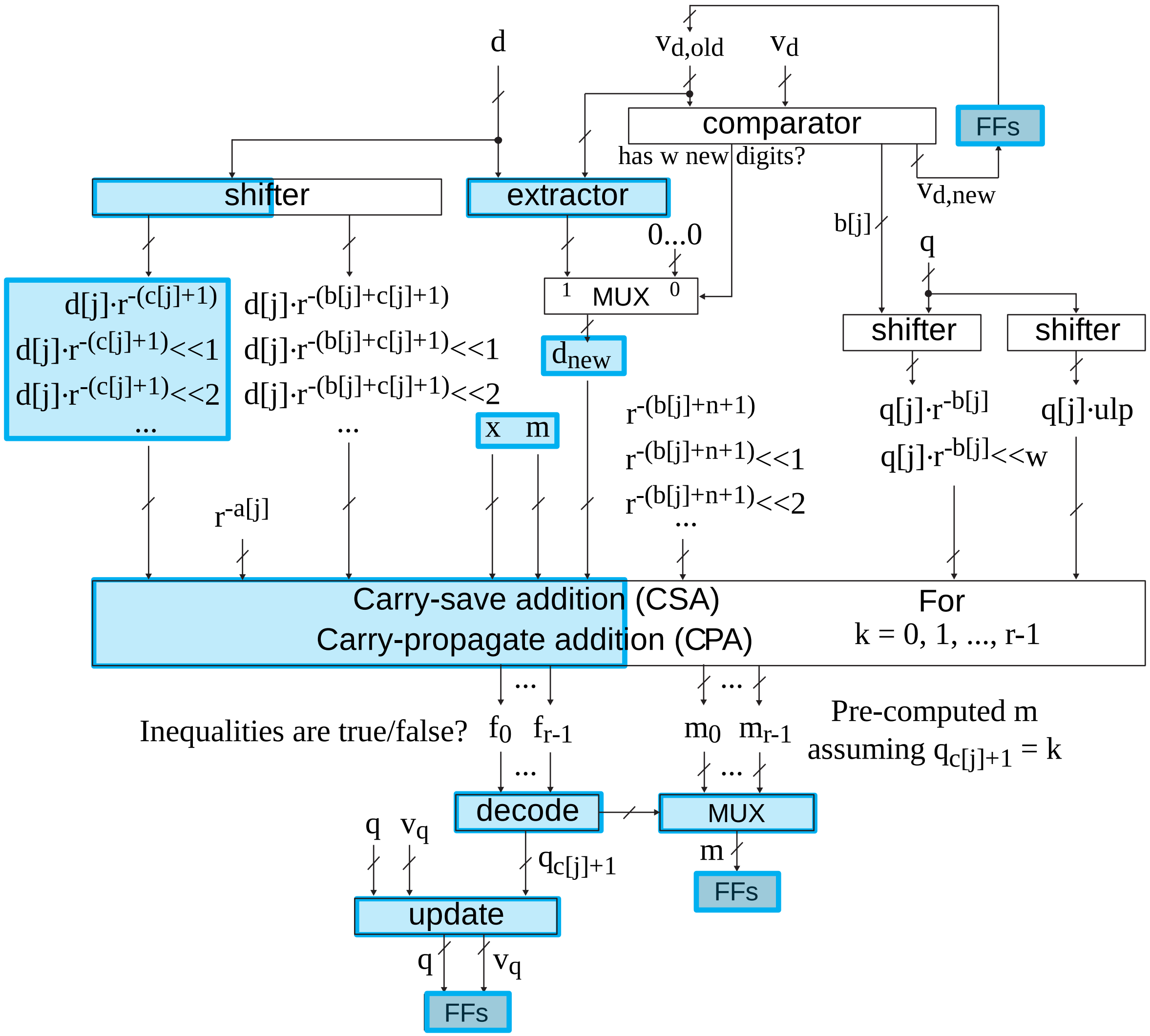
(b) Online: $T = 4$



(c) Offline: $T = 52$

Discussion

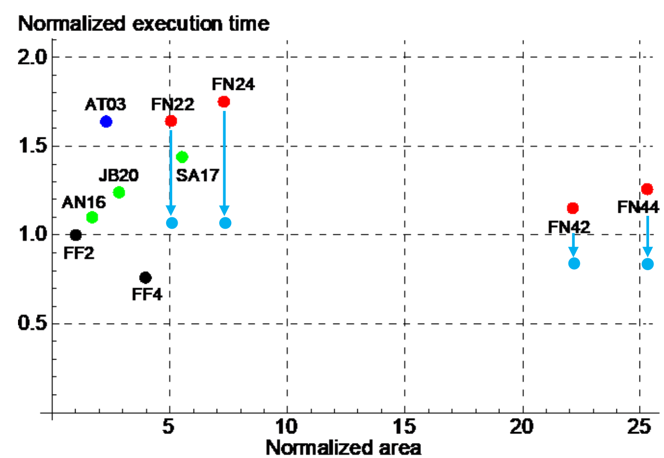
- Integration of both offline and online dividers in a single architecture



Used for offline division

If the operands are offline, then we can generate the output much more quickly with a small additional logic (bypass).

In this case, the execution time of our online dividers for offline division will be close to the execution time of the offline dividers (FF2, FF4).



(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.