Computer Arithmetic - Spring 1998
Dr. Guy Even
Dept. of Electrical Engineering - Systems

Lecture of June 4, 1998

Written by Roman Tsikel,
e-mail: romant@nexus.co.il
Rounding Computation

Introduction:
In this lecture next concepts will be defined and discussed:

- \textit{p-equivalent} numbers,
- \textit{p-representatives},
- \textit{sticky-bit} and \textit{sticky},
- Rounding unit will be presented,
- Addition algorithm will be discussed.

Definition: Two real numbers $x$ and $y$ are \textit{p-equivalent} (ד"כ - p) if:

1. $x - y$ or
2. $x, y \in (q \cdot 2^{-p}, [q + 1] \cdot 2^{-p})$ for any integer $q \in \mathbb{Z}$.

This denoted as $x \equiv y$.

In other words, binary representation of two p-equivalent numbers is the same at least for $p$ positions to the right of the binary point.

For every group of p-equivalent real numbers, let's define \textit{p-representative} (ץפ) as follows:

$$\text{rep}_p(x) = \begin{cases} 
q \cdot 2^{-p} & \text{if } x = q \cdot 2^{-p} \\
(q + \frac{1}{2}) \cdot 2^{-p} & \text{if } x \in (q \cdot 2^{-p}, [q + 1] \cdot 2^{-p}) 
\end{cases}$$

Note: all p-representative numbers are integral multipliers of $2^{-(p+1)}$.

The graphic representation of above definitions is presented below ($q$ is even).

For example: all numbers into the ( ) brackets are p-equivalent, and their p-representative is in the interval middle.

\[ \begin{array}{cccccccc}
q2^{-p+1} & (q+1)2^{-p+1} & (q+2)2^{-p+1} & (q+3)2^{-p+1} & (q+4)2^{-p+1} \\
\end{array} \]
Note:

- if binary representation of $x$ and $y$ are not the same for the first $p$ places after the binary point, then $x$ and $y$ are not $p$-equivalent;
- if binary representation of $x$ and $y$ are the same for the first $p$ places after the binary point, then the $p$-equivalency is not guaranteed.

**Claim 9:** Let $x$ and $y$ will be real numbers such, that $x \neq y$ for an integer $\alpha$, let $\alpha'$ will be integer such, that $\alpha' \leq \alpha$, then:

1. $\text{rep}_{\alpha}(\cdot x) = -\text{rep}_{\alpha}(\cdot x)$, and therefore $\cdot x \neq \cdot y$;
2. if $x \approx y$, then $x \approx y$;
3. for every integer $i$: $i \cdot \cdot 2^{-\alpha} \approx y \cdot y^{-i}$;
4. for every integer $q$: $x \cdot q' \cdot 2^{-\alpha} \neq y \cdot q' \cdot 2^{-\alpha}$.

**Proof:**

1. There are two cases:

   if $x = 2^{-\alpha} \cdot q$, then: $\begin{cases} \text{rep}_{\alpha}(x) = x \\ -x = 2^{-\alpha} \cdot (-q) \Rightarrow \text{rep}_{\alpha}(-x) = -x = -\text{rep}_{\alpha}(x) \end{cases}$

   if $x \in (q \cdot 2^{-\alpha}, [q + 1] \cdot 2^{-\alpha})$, then:

   $\begin{cases} \text{rep}_{\alpha}(x) = (q + \frac{1}{2}) \cdot 2^{-\alpha} \\ -x \in (-[q + 1] \cdot 2^{-\alpha}, -q \cdot 2^{-\alpha}) \Rightarrow \text{rep}_{\alpha}(-x) = (-q - \frac{1}{2}) \cdot 2^{-\alpha} = -\text{rep}_{\alpha}(x) \end{cases}$

   $\text{rep}_{\alpha}(x) = \text{rep}_{\alpha}(y) \Rightarrow \text{rep}_{\alpha}(-x) = -\text{rep}_{\alpha}(x) = -\text{rep}_{\alpha}(y) = \text{rep}_{\alpha}(-y)$, and then

   $-x \approx -y$.

   From this claim follows that every rounding may be made independently of the number sign.

2. If $x \neq y$, then $x, y \in (q \cdot 2^{-\alpha}, [q + 1] \cdot 2^{-\alpha})$.

   If we look at the interval $(q \cdot 2^{-\alpha}, [q + 1] \cdot 2^{-\alpha})$, when $\alpha' \leq \alpha$, is obvious, that the interval is bigger then the initial interval and actually includes it into itself (look the figure below) and therefore $x \approx y$ ($\alpha' \approx \alpha - 1$ in the below figure).
3. If \( x, y \in (q \cdot 2^{-\alpha}, [q + 1] \cdot 2^{-\alpha}) \), then \( x \cdot 2^{-i}, y \cdot 2^{-i} \in (q \cdot 2^{-(\alpha+i)}, [q + 1] \cdot 2^{-(\alpha+i)}) \), and therefore \( x \cdot 2^{-i} \approx y \cdot 2^{-i} \).

Intuitively - if two \( \alpha \)-equivalent binary numbers are shifted right by the \( i \) bits, then they become \((\alpha+i)\)-equivalent.

4. \[ x + q'2^{-\alpha}, y + q'2^{-\alpha} \in (q \cdot 2^{-\alpha} + q'2^{-\alpha}, [q + 1] \cdot 2^{-\alpha} + q'2^{-\alpha}) =
\]
\[ = ([q + q'2^{-\alpha}, [q + 1 + q'2^{-\alpha}] \cdot 2^{-\alpha}).
\]

The both expressions it the [ ] brackets are integers, because of \( q \) and \( q' \) are integers, and because of \( \alpha' \leq \alpha \), \( \alpha-\alpha' \) is positive integer (or zero) and \( q'2^{a-\alpha'} \) is integer too.

Therefore \( x+ q'2^{-\alpha} \approx y+ q'2^{-\alpha} \).

Intuitively, adding \( q'2^{-\alpha} \) to \( x \) and \( y \) still keeps them in the same interval with endpoints \( (q''2^{-\alpha}, [q''+1] \cdot 2^{-\alpha}) \).

**Question:** what value of \( \alpha \) we must choose in order two \( \alpha \)-equivalent numbers will be rounded to the same value?

For the RNE rounding it is enough \( \alpha - p \).

For the Round to zero and Round to \( \pm \infty \) it is enough \( \alpha - p-1 \).

**Claim 5:** If \( f' = \text{rep}_p(f) \), then:
1. \( \text{sign}_\text{md}(f) - \text{sign}_\text{md}(f') \);
2. \( \text{sign}_\text{md}(f) - f \iff \text{sign}_\text{md}(f') - f' \iff f - f' - q \cdot 2^{-(p-1)} \).

**Proof:**
1. Reminder: for the RNE rounding we have:
\[
\text{sig\_rnd}(f) = \begin{cases} 
q \cdot 2^{-(p-1)} & \text{if } f \in (q \cdot 2^{-(p-1)}, [q + \frac{1}{2}] \cdot 2^{-(p-1)}) 
(q + 1) \cdot 2^{-(p-1)} & \text{if } f \in ([q + \frac{1}{2}] \cdot 2^{-(p-1)}, [q + 1] \cdot 2^{-(p-1)}) 
q' \cdot 2^{-(p-1)} & \text{if } f = (q + \frac{1}{2}) \cdot 2^{-(p-1)}
\end{cases}
\]

In fact, operation \(\text{rep}_p(f)\) doesn’t influence on the \(\text{sig\_rnd}\) operation, because if, for example, \(f\) belongs to the first interval, then \(f'\) belongs to the same interval - actually it is now at the middle of the interval, the same happens, when \(f\) belongs to the second interval. In third case, \(f - f'\).

Note: the above Proof doesn't presented in the Class.

2. \(\text{sig\_md}(f) - f \Rightarrow f - q \cdot 2^{-(p-1)} \Rightarrow f - f' \Rightarrow \text{sig\_rnd}(f') - f'.\)

As follows from the Claim, we can use \(\text{rep}_p(f)\) value in \(\text{sig\_rnd}(f)\) computation, rather then the \(f\) value itself.

**Claim 6:** Suppose \(\eta(x) = (s_x, e_x, f_x)\) and \(\eta(y) = (s_y, e_y, f_y)\). If \(x = y\), then:

1. \(s_x = s_y, e_x = e_y, f_x = f_y\);
2. \(r(x) = r(y)\).

**Proof:**

If \(x = y\), the proof is obvious.

From now, let’s discuss the case \(x \neq y\).

If \(x = y\), then \(x, y \in I = (q \cdot 2^{e_x-\rho}, [q + 1] \cdot 2^{e_x-\rho})\). Because the interval is open, and \(q\) and \((q+1)\) have the same sign, zero is not in the interval, therefore or \(x, y > 0\), or \(x, y < 0\), and \(s_x = s_y\). Let’s suppose \(s_x = s_y = 0\).

**Support Claim:** If \(x\) and \(y\) belongs to the interval \(I\), then or \(f_x, f_y \in [0,1)\), or \(f_x, f_y \in [1,2)\).

**Proof:** Suppose, that the above claim is not true, i.e. \(f_x \in [0,1)\) and \(f_y \in [1,2)\), for example. Then \(e_x = e_{\min}\) and \(e_y \geq e_{\min}\) and follows, that \(2^{e_{\min}} \in I\) and in is impossible case.
• If \( f_x, f_y \in [0,1) \), then \( e_x = e_y = e_{\text{min}} \) and follows from Claim 9, part 3 \((x \cdot 2^{-i} \leq y \cdot 2^{-i})\), that \( f_x = f_y \).

• When \( f_x, f_y \in [1,2) \): let's suppose, that \( e_y < e_x \), for example. It is follows that:

\[
y = f_y \cdot 2^{e_y} < 2 \cdot 2^{e_y} \leq 2^{e_x} \leq f_x \cdot 2^{e_x} = x.
\]

From above equation follows, that \( 2^{e_x} \in I \). Because \( 2^{e_x} = i \cdot 2^{e_x-p} \) \((i - \text{integer})\), statement, that \( 2^{e_x} \in I \) is not true, and therefore follows, that \( e_x = e_y \) and \( f_x = f_y \).

Proof of the second part of the Claim - \( r(x) = r(y) \), follows from Claim 5.
Rounding Unit
(without exceptions handling)

Assumptions: the value we want to round is: \( x = \text{val}(s, e, f^e, f^m) \), i.e. the input to the unit is not the exact value of \( x \), but \( \text{val}(s, e, f^e, f^m) \).

Functionality: 1. The normalization shift box computes value of
\[ \eta(\text{val}(s, e, f^e, f^m)) = (s, e^o, f^n) \].
Reminder: Normalization shift maps every real number $x$ to $(s,e,f)$ so that $x - \text{val}(s,e,f)$.

If $\text{abs}(x) \geq 2^{\text{\max}}$, then $1 \leq f < 2$, if $\text{abs}(x) < 2^{\text{\max}}$, then $0 \leq f < 1$.

2. The $f^1 = \text{rep}_p(f^2)$ value is computed, i.e. final Sticky-bit is computed.

3. Significand rounding: $f^2 = \text{sig}_\text{rnd}(f^1)$. It is obvious, that the \textit{sign} and the Rounding mode must be input to this block.

4. Now, if $f^2$ is rounded to 2 ($f[1,0] = 10$), then $e^2$ must be incremented by one, this is done by MUX, and $f$ must be set to one - $f[0] = 1$, by OR, according to the algorithm.

5. Exponent rounding computed:

$$(s, e_{\text{out}}, f_{\text{out}}) = \text{exp}_\text{rnd}(s, e^2, f^2) = \begin{cases} (s, e^2, f^2) & \text{if } f \cdot 2^e \geq x_{\text{max}} \\ \eta(\text{val}(s, e_{\text{out}}, f_{\text{out}})) & \text{otherwise} \end{cases}$$

Of course, this unit uses in computations two additional inputs: sign and Rounding mode.

Correctness:
Sticky-bit and Sticky

As known, if binary representation of the number $f$ is as follows: $f = f_0.f_{-1}f_{-2}...f_{-p}...f_{-l}$, $\sum_{i=0}^{l} 2^i \cdot f_i$. Let’s define Sticky-bit as OR of all bits after the $f_{-p}$ bit:

$$\text{Sticky-bit} (f, p) = f_{-(p+1)}, f_{-(p+2)},... = \bigvee_{i=p+1} f_i,$$

and

$$\text{Sticky} (f, p) = f_0.f_{-1}f_{-2}...f_{-p}f_{-(p+1)}.$$

The Relationship between Sticky-bit and Representatives.

Claim 8: $\text{rep}_p(f) = \text{Sticky} (f, p)$.

Proof: If all the bits of $f$, starting at $p+1$ position after the binary point are zero: $f = f_0.f_{-1}f_{-2}...f_{-p}000...$, then $f$ is integral multiple of $2^{-p}$ and therefore $f - \text{rep}_p(f)$ and, according to the above definitions $\text{Sticky-bit}(f, p) = 0$, and $\text{Sticky} (f, p) = f - \text{rep}_p(f)$.

If $f$ is not integral multiple of $2^{-p}$, then Sticky-bit $(f, p) = 1$, and therefore $\text{rep}_p(f) = \text{Sticky} (f, p)$.

Addition

Input: $(s_1, e_1, f_1)$ and $(s_2, e_2, f_2)$ - normalized factorings.

Let’s choose: $x = \text{val}(s_1, e_1, f_1)$ and $y = \text{val}(s_2, e_2, f_2)$.

Output: $(s, e, f)$, when $x + y = \text{val}(s, e, g)$, $\eta(x+y) = (s, e, f)$.

The algorithm is divided into three steps: Preprocessing, Adding and Rounding.

Preprocessing:

- Swap: First of all both exponents are compared, and operands selected such, that $e_1 \geq e_2$. From now, we assume that $e_1 \geq e_2$.
- Alignment Shift: Replace $f_2$ with $f'_2 = f_2 \cdot 2^{-\delta}$, when $\delta = \min\{e_1 - e_2, p + 2\}$.
- Representative Computing: $f''_2 = \text{rep}_{p+1}(f'_2)$. Now $f'_2 = f_0.f_{-1}f_{-2}...f_{-(p+2)}$. 
Adding:

\[ s' = s_1 \]
\[ e' = e_1 \]
\[ g' = (-1)^{s'} \cdot f_1 + (-1)^{s'} \cdot f_2 \]