Problem 1: Define the Type I component to be a component that adds nine 3-bit unsigned numbers and produces their full precision summation.

Using only Type I components and one CPA (carry propagate adder) show how you can construct an adder capable of adding nine 30-bit unsigned numbers to produce their full precision summation.

Solution:

The nine 30-bit unsigned numbers are $N_1, N_2, \cdots, N_9$. Then their full precision summation $\sum_{i=1}^{9} N_i$ will require 34 bits for its representation.

(Consider that the worst case summation of nine 30-bit numbers is $(2^{30}-1)\times 9 = (2^{30}-1)\times(2^3+1) = 2^{33}-2^3+2^{30}-1 = 2^{33}+2^{30}-9 > 2^{33}-1$).

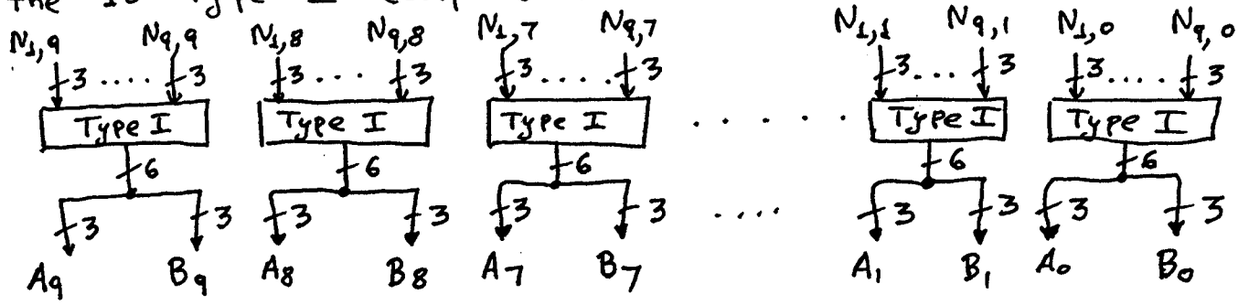
Each N_i can be decomposed into 10 3-bit blocks (or radix-8 digits) as

$$N_i = N_{i,9}\, N_{i,8}\, N_{i,7}\, N_{i,6}\, N_{i,5}\, N_{i,4}\, N_{i,3}\, N_{i,2}\, N_{i,1}\, N_{i,0}.$$

The value of N_i is then

$$N_i = N_{i,9}\times 2^{27} + N_{i,8}\times 2^{24} + N_{i,7}\times 2^{21} + N_{i,6}\times 2^{18} + N_{i,5}\times 2^{15} + N_{i,4}\times 2^{12} + N_{i,3}\times 2^{9} + N_{i,2}\times 2^{6} + N_{i,1}\times 2^{3} + N_{i,0}.$$

The following figure shows the inputs and outputs of the 10 Type I components.



Observe that each Type I component has a 6-bit output since the worst case summation of nine 3-bit numbers is $9\times(2^3-1) = (2^3+1)(2^3-1) = 2^6-1$ which requires 6 bits for its representation.

The summation of the nine 30-bit numbers is

$$\sum_{i=1}^{9} N_i = \left(\sum_{i=1}^{9} N_{i,9}\right)\times 2^{27} + \left(\sum_{i=1}^{9} N_{i,8}\right)\times 2^{24} + \left(\sum_{i=1}^{9} N_{i,7}\right)\times 2^{21} +$$

$$+ \cdots + \left(\sum_{i=1}^{9} N_{i,1}\right)\times 2^{3} + \left(\sum_{i=1}^{9} N_{i,0}\right) =$$

$$= (A_9 B_9)\times 2^{27} + (A_8 B_8)\times 2^{24} + (A_7 B_7)\times 2^{21} + \cdots + (A_1 B_1)\times 2^{3}$$

$$+ (A_0 B_0) =$$

$$= (A_9\times 2^3 + B_9)\times 2^{27} + (A_8\times 2^3 + B_8)\times 2^{24} + (A_7\times 2^3 + B_7)\times 2^{21} + \cdots +$$

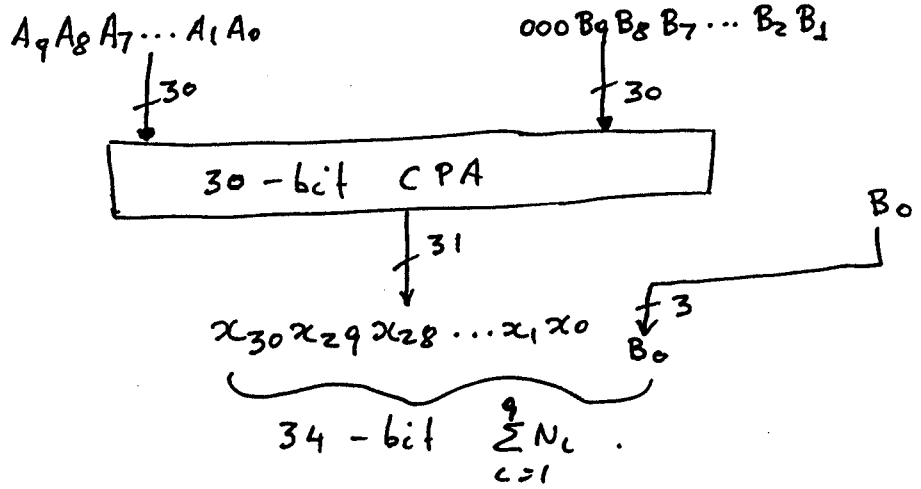
$$+ \cdots + (A_1\times 2^3 + B_1)\times 2^3 + (A_0\times 2^3 + B_0) =$$

$$= (A_9\times 2^{27} + A_8\times 2^{24} + A_7\times 2^{21} + A_6\times 2^{18} + \cdots + A_1\times 2^3 + A_0)\times 2^3$$

$$+ (B_9\times 2^{27} + B_8\times 2^{24} + B_7\times 2^{21} + B_6\times 2^{18} + \cdots + B_1\times 2^3 + B_0) =$$

$$= (A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0)\times 2^3 + (B_9 B_8 B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0).$$

$$= \begin{array}{r|l} A_9\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0 & 000 \\ + \quad B_9\ B_8\ B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1 & B_0 \end{array}$$



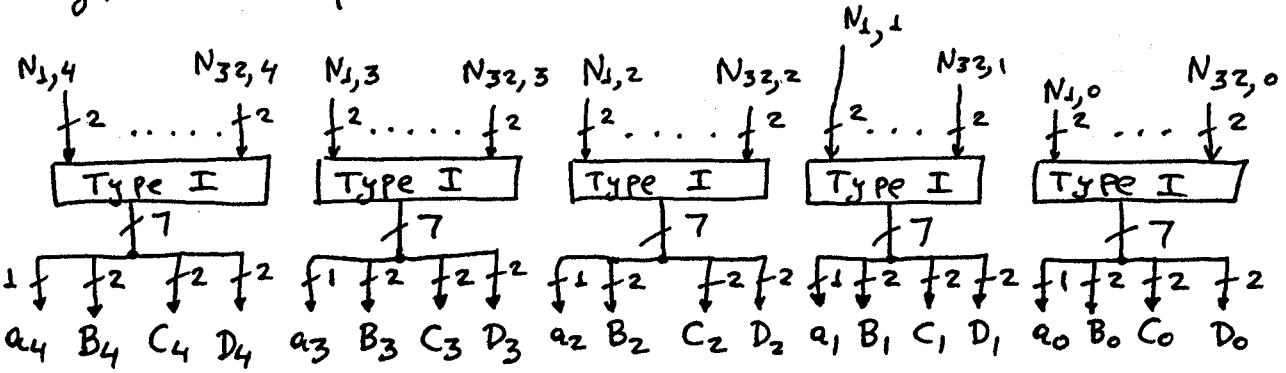
Problem 2: Define the Type I component to be a component that adds 32 2-bit unsigned numbers and produces their full precision summation. Using only Type I components and CPAs (carry propagate adders) show how you can construct an adder capable of adding 32 10-bit unsigned numbers to produce their full precision summation.

Solution: Let the 32 10-bit unsigned numbers be $N_1, N_2, N_3, \ldots, N_{32}$. Then their full precision summation $\sum_{i=1}^{32} N_i$ will require 15 bits for its representation (observe that the worst case summation of 32 10-bit numbers is $32 \times (2^{10}-1) = 2^5 \times (2^{10}-1) = (111111111100000)_2$).

Each N_i can be decomposed into five 2-bit blocks (or radix-4 digits) as $N_i = N_{i,4}\, N_{i,3}\, N_{i,2}\, N_{i,1}\, N_{i,0}$ where each $N_{i,j}$ is a 2-bit number (or a radix-4 digit). The value of N_i is then

$$N_i = N_{i,4} \times 2^8 + N_{i,3} \times 2^6 + N_{i,2} \times 2^4 + N_{i,1} \times 2^2 + N_{i,0}.$$

The figure below shows the inputs and outputs of the five Type I components.



(4) d

Each Type I component has a 7-bit output since the worst case summation of 32 2-bit unsigned numbers is $32 \times (2^2-1) = 2^5 \times (2^2-1) = (1100000)_2$.

The summation of the 32 10-bit numbers is

$$\sum_{i=1}^{32} N_i = \left(\sum_{i=1}^{32} N_{i,4}\right) \times 2^8 + \left(\sum_{i=1}^{32} N_{i,3}\right) \times 2^6 + \left(\sum_{i=1}^{32} N_{i,2}\right) \times 2^4 + \left(\sum_{i=1}^{32} N_{i,1}\right) \times 2^2 + \sum_{i=1}^{32} N_{i,0} =$$

$$= (a_4 B_4 C_4 D_4) \times 2^8 + (a_3 B_3 C_3 D_3) \times 2^6 + (a_2 B_2 C_2 D_2) \times 2^4 + (a_1 B_1 C_1 D_1) \times 2^2 + (a_0 B_0 C_0 D_0) =$$

$$= (a_4 \times 2^6 + B_4 \times 2^4 + C_4 \times 2^2 + D_4) \times 2^8 + (a_3 \cdot 2^6 + B_3 \cdot 2^4 + C_3 \cdot 2^2 + D_3) \cdot 2^6 + (a_2 \cdot 2^6 + B_2 \cdot 2^4 + C_2 \cdot 2^2 + D_2) \cdot 2^4 + (a_1 \cdot 2^6 + B_1 \cdot 2^4 + C_1 \cdot 2^2 + D_1) \cdot 2^2 + (a_0 \cdot 2^6 + B_0 \cdot 2^4 + C_0 \cdot 2^2 + D_0) =$$

$$= (a_4 \cdot 2^8 + a_3 \cdot 2^6 + a_2 \cdot 2^4 + a_1 \cdot 2^2 + a_0) \cdot 2^6 + (B_4 \cdot 2^8 + B_3 \cdot 2^6 + B_2 \cdot 2^4 + B_1 \cdot 2^2 + B_0) \cdot 2^4 + (C_4 \cdot 2^8 + C_3 \cdot 2^6 + C_2 \cdot 2^4 + C_1 \cdot 2^2 + C_0) \cdot 2^2 + (D_4 \cdot 2^8 + D_3 \cdot 2^6 + D_2 \cdot 2^4 + D_1 \cdot 2^2 + D_0) =$$

$$= (a_4 0 a_3 0 a_2 0 a_1 0 a_0) \cdot 2^6 + (B_4 B_3 B_2 B_1 B_0) \cdot 2^4 + (C_4 C_3 C_2 C_1 C_0) \cdot 2^2 + (D_4 D_3 D_2 D_1 D_0).$$

Obviously, the following three two-operand additions need to take place

$$\begin{array}{rl} & a_4\,0\,a_3\,0\,a_2\,0\,a_1\,0\,a_0 \;|\; 000000 \\ +) & \quad B_4\;B_3\;B_2\;B_1 \;|\; B_0\,0\,0\,0 \\ \hline & x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0 \;|\; B_0\,0\,0\,0 \end{array}$$

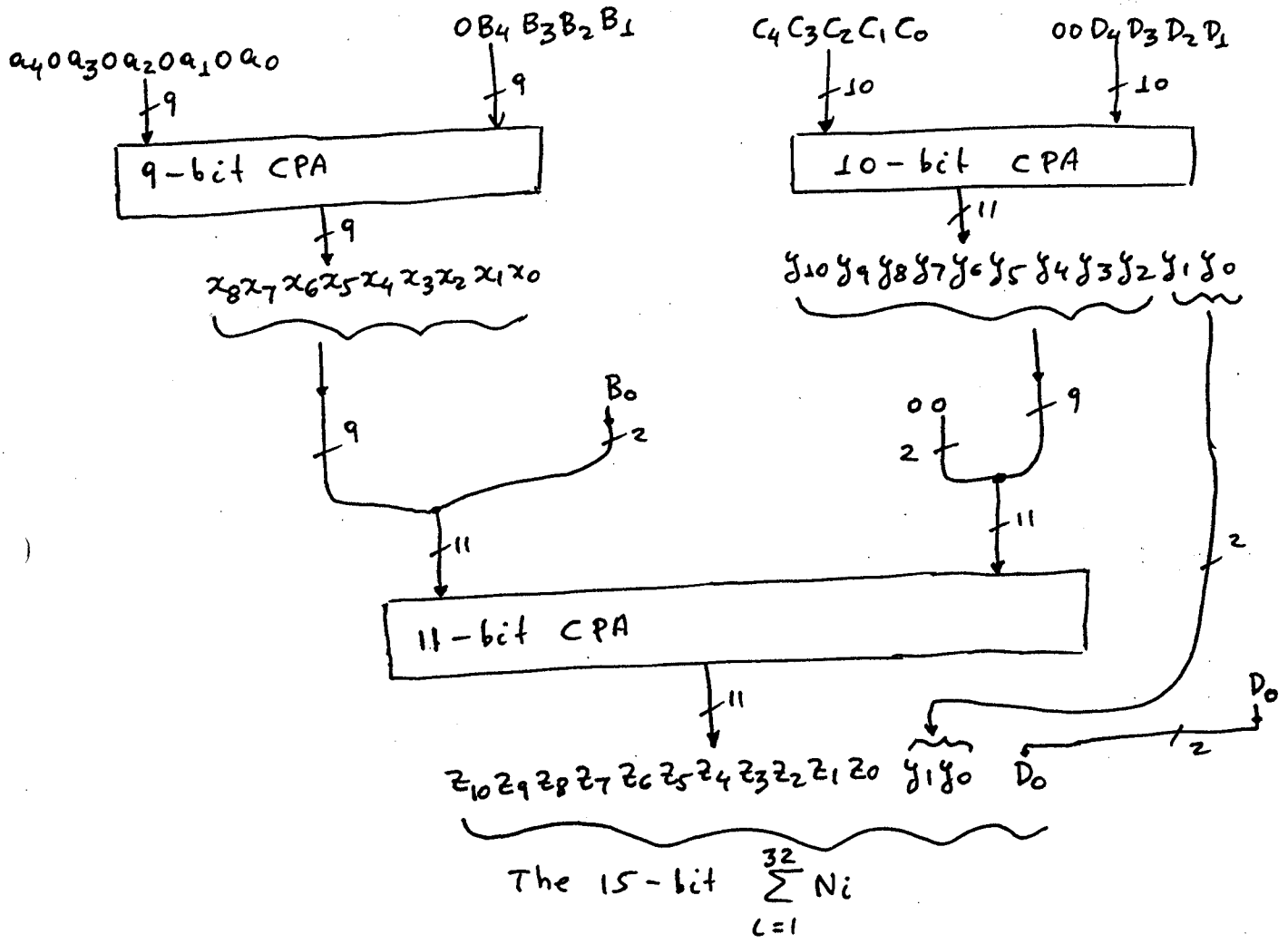
9 bits; 6 bits; 15 bits

$$\begin{array}{rl} & C_4\;C_3\;C_2\;C_1\;C_0 \;|\; 0\,0 \\ +) & \quad D_4\;D_3\;D_2\;D_1 \;|\; D_0 \\ \hline & y_{10} y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0 \;|\; D_0 \end{array}$$

11 bits; 2 bits; 13 bits

⑤ d

	$x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0\ B_0$	0000
+)	$y_{10} y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2$	$y_1 y_0\ D_0$
	$z_{10} z_9 z_8 z_7 z_6 z_5 z_4 z_3 z_2 z_1 z_0$	$y_1 y_0\ D_0$



The 15-bit $\sum_{i=1}^{32} N_i$

Signed-Digit (SD) Addition

For this handout $\overline{a}$ means $-a$. For example $\overline{1}$ means -1; $\overline{5}$ means -5 etc.

Let $X=(x_{n-1}x_{n-2}\cdots x_1x_0)_r$; $Y=(y_{n-1}\cdots y_0)_r$ be two radix-r signed-digit (SD) numbers; (this means numbers with signed digits).

In the presentation below w_i indicates the intermediate summation digit of stage i, t_i indicates the carry-in to stage i, t_{i+1} indicates the carry-out of stage i while s_i is the final summation digit.

The procedure for signed-digit addition follows:

- Range of w_i; $w_i \in [w_{min} \quad w_{max}]$

(1) $$t_{i+1} = \begin{cases} 0 \text{ if } w_{min} \le x_i+y_i \le w_{max} \\ 1 \text{ if } x_i+y_i > w_{max} \\ \overline{1} \text{ if } x_i+y_i < w_{min} \end{cases}$$

$w_i = x_i + y_i - r \times t_{i+1}$; (this operation brings x_i+y_i in the correct range; r = radix).

** Note that both t_{i+1} and w_i are functions of x_i, y_i.

② $s_i = w_i + t_i$

Ranges of w_i, x_i, y_i, s_i

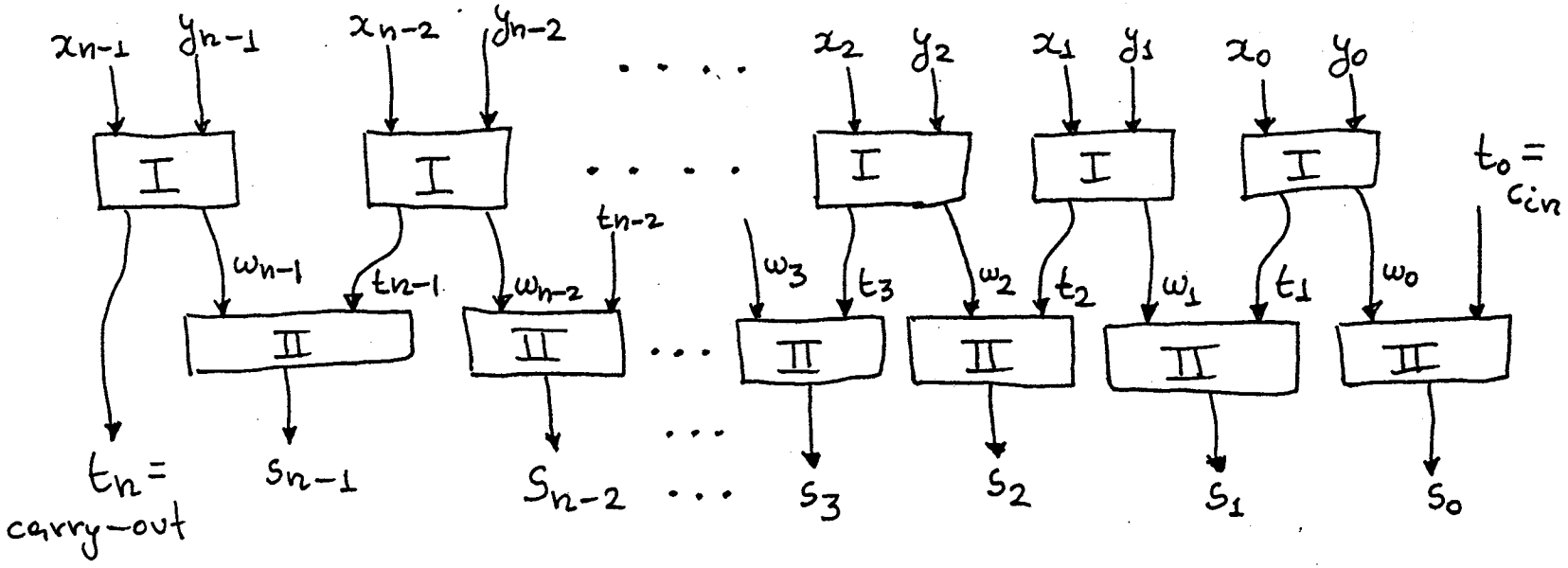
$w_i \in \left[-\frac{r}{2} \quad \frac{r}{2}\right]$ if r = even

$w_i \in \left[\frac{-(r-1)}{2} \quad \frac{r-1}{2}\right]$ if r = odd

$x_i, y_i, s_i \in \left[-\left(\frac{r}{2}+1\right) \quad \frac{r}{2}+1\right]$ if r = even

$x_i, y_i, s_i \in \left[\frac{-(r+1)}{2} \quad \frac{r+1}{2}\right]$ if r = odd

(3) e



A parallel SD Adder

Example 1: Given $X = 12\bar{5}\bar{6}$, $Y = 65\bar{2}\bar{3}$, radix $r = 10$, $x_i, y_i, s_i \in \{\bar{6}, \bar{5}, \bar{4}, \ldots, 4, 5, 6\}$, perform the signed-digit addition $S = X + Y$.

Solution: Here $w_i \in \{\bar{5}, \bar{4}, \ldots, 4, 5\}$.

					t_i
	1	1	$\bar{1}$	$\bar{1}$	
$X =$	1	2	$\bar{5}$	$\bar{6}$	x_i
$Y =$	6	5	$\bar{2}$	$\bar{3}$	y_i
	$\bar{3}$	$\bar{3}$	3	1	$w_i = x_i + y_i - r \times t_{i+1}$
$S =$	1	$\bar{2}$	$\bar{4}$	2	1 $\quad s_i = w_i + t_i$

Double check to see that

$$X = (12\bar{5}\bar{6})_{10} = 1000 + 200 - 50 - 6 = 1144$$

$$Y = (65\bar{2}\bar{3})_{10} = 6000 + 500 - 20 - 3 = 6477$$

and $S = 1\bar{2}\bar{4}21 = 10{,}000 - 2{,}000 - 400 + 20 + 1$
$= 7621$ which is $X + Y$.

Example 2: Given $X=(25\bar{3}\bar{4})_8$, $Y=(\bar{3}\bar{2}23)_8$, radix $r=8$, $x_i, y_i, s_i \in \{\bar{5}, \bar{4}, \cdots, 4, 5\}$, perform the SD subtraction $S=X-Y$.

Solution: Here $w_i \in \{\bar{4}, \bar{3}, \cdots, 3, 4\}$;

Also $-Y = 32\bar{2}\bar{3}$.

					t_i
1	1	$\bar{1}$	$\bar{1}$		
	2	5	$\bar{3}$	$\bar{4}$	x_i
	3	2	$\bar{2}$	$\bar{3}$	y_i
	$\bar{3}$	$\bar{1}$	3	1	$w_i = x_i + y_i - r \times t_{i+1}$
$S=$ 1	$\bar{2}$	$\bar{2}$	2	1	$s_i = w_i + t_i$

Double check to see that

$X = (25\bar{3}\bar{4})_8 = 2\times8^3+5\times8^2-3\times8-4 = 1316$

$-Y = (32\bar{2}\bar{3})_8 = 3\times8^3+2\times8^2-2\times8-3 = 1645$

and $S = 1\bar{2}\bar{2}21 = 8^4-2\times8^3-2\times8^2+2\times8+1 = 2961$

which is $X-Y$.

Converting from conventional radix-r to signed-digit (SD) radix-r systems

Problem: Let $X = (x_{n-1}\, x_{n-2} \cdots x_1\, x_0)_r$ be a conventional radix-r (unsigned-digit) number; (here $x_i \in [0 \quad r-1]$). Convert X into a signed-digit number with signed digits belonging to the SD set $[\bar{a} \quad a]$.

Solution:

① $$t_{i+1} = \begin{cases} 0 & \text{if } x_i < a \\ 1 & \text{if } x_i \geq a \end{cases}$$

$$y_i = x_i - r \times t_{i+1}$$

② $$z_i = y_i + t_i$$

Then the converted X is the number $Z = z_{n-1}\, z_{n-2} \cdots z_1\, z_0$ where the digits z_i are given by ②.

Example: Let X be a radix-10 (decimal) number where $X=(0678)_{10}$. Convert X into a signed-digit number with signed digits belonging to the set $\{\overline{6}, \overline{5}, \cdots, 5, 6\}$.

Solution:

i	0	1	1	1	t_i
$X=$	0	6	7	8	x_i
	0	$\overline{4}$	$\overline{3}$	$\overline{2}$	$y_i = x_i - r \times t_{i+1}$
$Z=$	1	$\overline{3}$	$\overline{2}$	$\overline{2}$	$z_i = y_i + t_i$

Check to see that $Z=(1\overline{3}\,\overline{2}\,\overline{2})_{10} =$

$1000 - 300 - 20 - 2 = 678$

Multipliers based on string property (Booth multipliers)

String property: 0111....1110
1000....00$\bar{1}$0 where $\bar{1} = \blacksquare - 1$

- Examining two bits from multiplier field; performing one-bit right shifts

Multiplier bits

00 string property → (0)0 → value = 0 ⇒ 0 × multiplicand / 1-bit right shift

01 string → (1)$\bar{1}$ → value = 1 ⇒ 1 × multiplicand / 1-bit right shift

10 string → ($\bar{1}$)0 → value = −1 ⇒ −1 × multiplicand / right shift

11 string → (0)$\bar{1}$ → value = 0 ⇒ 0 × multiplicand / right shift

* Details on the sequential Booth algorithm for signed (2's complement) multiplication (examining two bits at a time) are offered in the EE 3755 handout entitled "Fixed Point Multiplication".

- Examining three bits from multiplier field; performing 2-bit right shifts

Multiplier bits

000 string → (00)0
→ value = 0 ⇒ 0 × multiplicand / double right shift

001 string → (01)$\overline{1}$
→ = 1 ⇒ 1 × multiplicand / double right shift

010 string → (1$\overline{1}$)0
→ = 2 − 1 = 1 ⇒ 1 × multiplicand / double right shift

011 string → (10)$\overline{1}$
→ = 2 ⇒ 2 × multiplicand / double right shift

100 string → ($\overline{1}$0)0
→ = −2 ⇒ −2 × multiplicand / double right shift.

101 string → ($\overline{1}$1)$\overline{1}$
→ = −2 + 1 = −1 ⇒ −1 × multiplicand / double shift

110 string → (0$\overline{1}$)0
→ = −1 ⇒ −1 × multiplicand / double right shift

111 string → (00)$\overline{1}$
→ = 0 ⇒ 0 × multiplicand / double right shift.

* Details on the sequential version of this algorithm (examining three bits at a time) are offered in the EE 3755 handout entitled "Fixed Point Multiplication".

- Examining four bits from multiplier field; performing 3-bit right shifts.

Multiplier bits

0000 string → (000)0
↳ value = 0 ⇒ 0 × multiplicand / 3-bit right shift.

0001 string → (001)$\bar{1}$
↳ = 1 ⇒ 1 × multiplicand / 3-bit right shift.

0010 string → (01$\bar{1}$)0
↳ 2 − 1 = 1 ⇒ 1 × multiplicand / 3-bit right shift.

0011 string → (010)$\bar{1}$
↳ = 2 ⇒ 2 × multiplicand / 3-bit right shift.

0100 string → (1$\bar{1}$0)0
↳ = 4 − 2 = 2 ⇒ 2 × multiplicand / 3-bit r. shift

0101 string → (1 1 1)$\bar{1}$
↳ = 4 − 2 + 1 = 3 ⇒ 3 × multiplicand / 3-bit right shift.

⋮
etc
⋮

* Details on the sequential version of this algorithm (examining four bits at a time) are offered in the EE 3755 handout entitled "Fixed Point Multiplication".

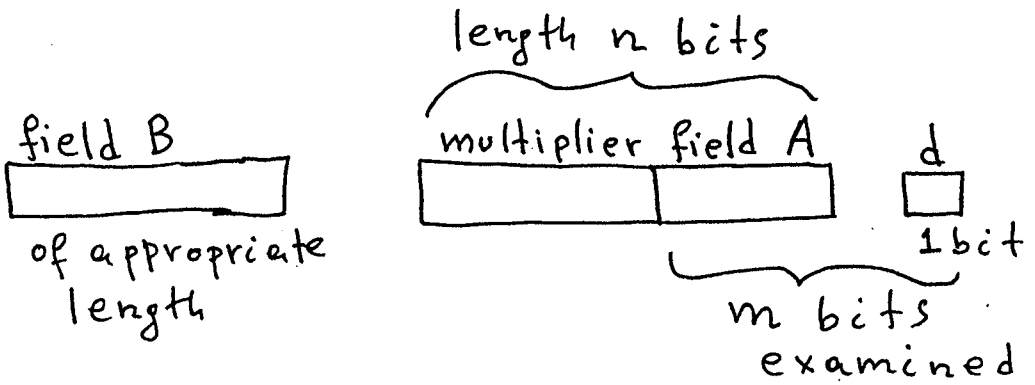
• In general, if m bits are examined from the multiplier field and (m−1)-bit right shifts are performed, then the multiple of the multiplicand that has to be added to the partial product can be found by reencoding (recoding) (using the string property) the m-bit multiplier pattern and find the value of the (m−1)-bit leftmost part of the re-coded version.

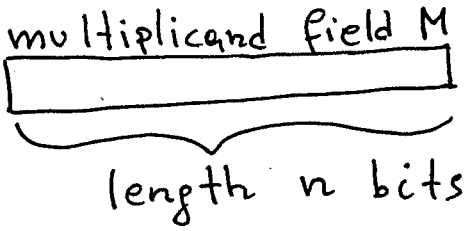
Example: Consider a multiplier based on the string property (Booth multiplier). Here 8 bits are examined and 7-bit right shifts are performed. If the observed multiplier pattern is 11011101, what is the multiple of the multiplicand that has to be added to the partial product?

Answer: 11011101 $\xrightarrow{\text{string}}$ $0\bar{1}100\bar{1}1\bar{1}$

→ value $= -2^5 + 2^4 - 2 + 1 = -17$

(5) f





- Initialization: A ← multiplier;
  M ← multiplicand;
  B ← 0, 0 ..., 0, 0.
  d ← 0.

- The algorithm:

  Perform $\frac{n}{m-1}$ cycles as follows:

(i) B ← B + k × M; (ignore carry out of addition; k = function of the m bits examined)

(ii) Right-shift by (m−1) bits B, A, d with sign extension at the left.