

DSP in VLSI Design

Homework (IX)

Processing Elements Design

Reference Answer

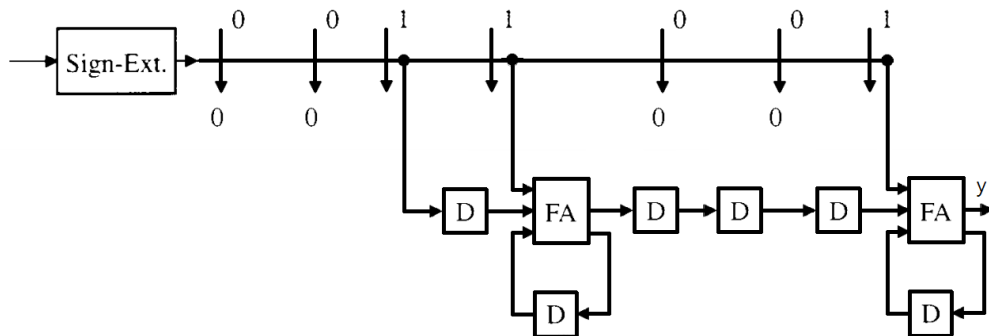
1. Find the simplest implementation of a serial/parallel multiplier with fixed coefficient when the coefficient is

- (a) $(0.011001)_{2C}$
- (b) $(0.111011)_{2C}$
- (c) $(1.011001)_{2C}$

The simplest implementation need transform 2C to CSDC.

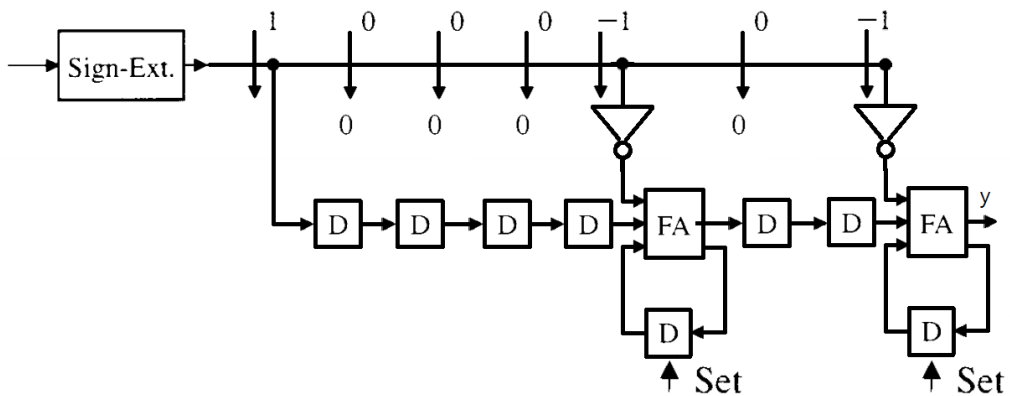
(a) $(0.011001)_{2C} = (0.10-1001)_{CSDC}$

In this case, the number of non-zero is the same, so use 2C is fine.



(b) To ensure the output is the same, so we do a sign extension.

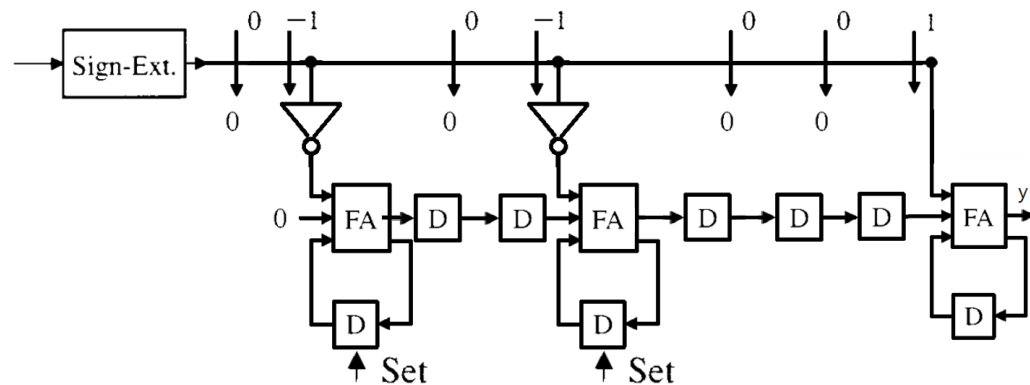
$(00.111011)_{2C} = (01.00-110-1) \rightarrow (01.000-10-1)_{CSDC}$



(c) To ensure the output is the same, so we do a sign extension.

$$(11.011001)_{2C} = (11.10-1001) \rightarrow (100.-10-1001) \rightarrow (00.-10-1001)_{CSDC}$$

Original ext. data is 8bit, so one bit will be overflow.



2. Conversion between RGB and YCbCr digital color video image formats can be performed by the following transformations:

$$R = Y + 350Cr / 256 - 175 / 256$$

$$G = Y - 86Cb / 256 - 178Cr / 256 + 132 / 256$$

$$B = Y + 444Cb / 256 - 222 / 256$$

and

$$Y = (77R + 150G + 29B) / 256$$

$$Cb = (-44R - 87G + 131B) / 256 + 128$$

$$Cr = (131R - 110G - 21B) / 256 + 128$$

The color components are quantized to 8 bits. Derive an implementation based on

(a) Bit-serial multipliers

(b) Distributed arithmetic

(c) Compare the two implementations.

$$R = (256Y + 350Cr - 175) / 256$$

$$G = (256Y - 86Cb - 178Cr + 132) / 256$$

$$B = (256Y + 444Cb - 222) / 256$$

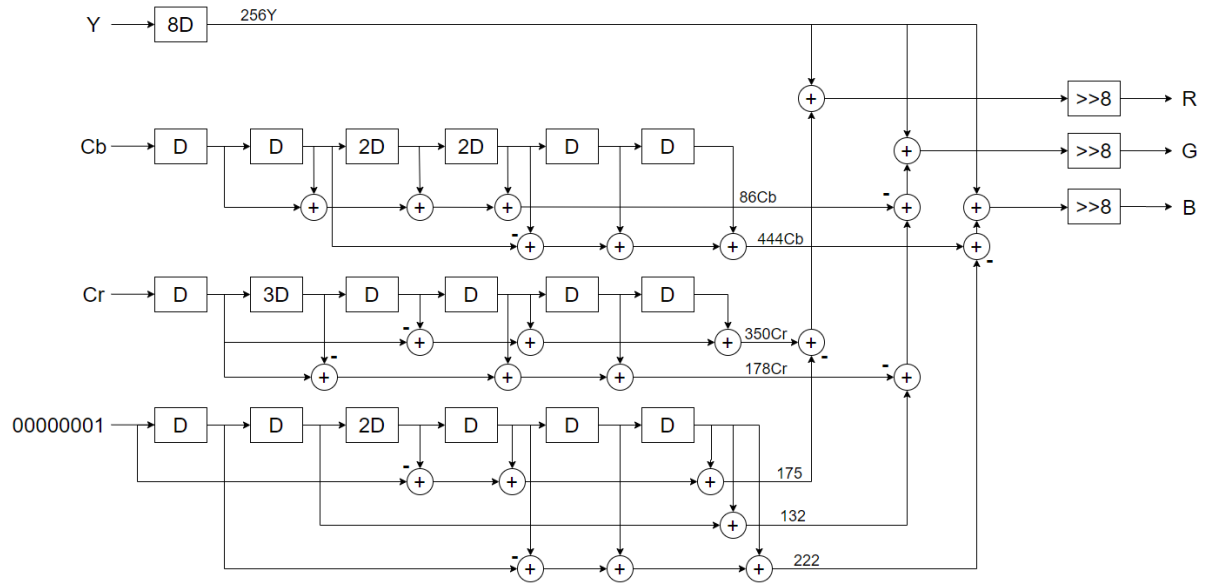
(a)

For the coefficient:

$$350 = 2^8 + 2^6 + 2^5 - 2^1, \quad 175 = 2^7 + 2^5 + 2^4 - 2^0$$

$$86 = 2^6 + 2^4 + 2^2 + 2^1, \quad 178 = 2^7 + 2^6 - 2^4 + 2^1, \quad 132 = 2^7 + 2^2$$

$$444 = 2^8 + 2^7 + 2^6 - 2^2, \quad 222 = 2^7 + 2^6 + 2^5 - 2^1$$



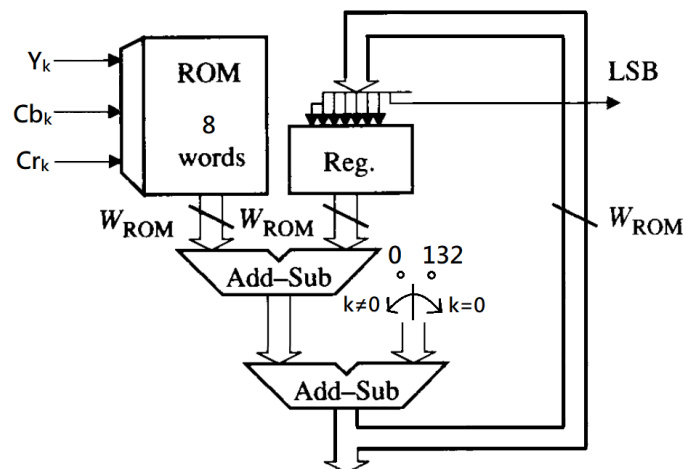
RGB to YCbCr is similar to this way.

(b)

Take $G = (256Y - 86Cb - 178Cr + 132)/256$ for example.

For every Y, Cb, Cr bit (Y_k means the k th bit in Y), there are 8 possible cases:

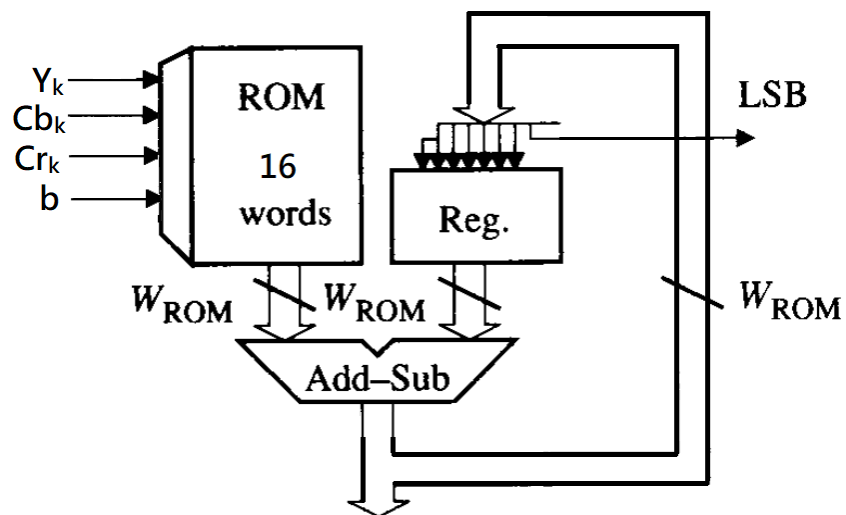
Y_k	Cb_k	Cr_k	$256Y_k - 86Cb_k - 178Cr_k$
0	0	0	0
0	0	1	-178
0	1	0	-86
0	1	1	-264
1	0	0	256
1	0	1	78
1	1	0	170
1	1	1	-8



We can save the result data in ROM, and Y_k, Cb_k, Cr_k act as address.

Or you can combine the bias coefficient to ROM:

Y_k	Cb_k	Cr_k	b	$256Y_k - 86Cb_k - 178Cr_k + 132b$
0	0	0	0	0
0	0	1	0	-178
0	1	0	0	-86
0	1	1	0	-264
1	0	0	0	256
1	0	1	0	78
1	1	0	0	170
1	1	1	0	-8
0	0	0	1	132
0	0	1	1	-46
0	1	0	1	46
0	1	1	1	132
1	0	0	1	-388
1	0	1	1	210
1	1	0	1	302
1	1	1	1	124



$b=1$ if k is 0.

Finally, the output need right shift 8.

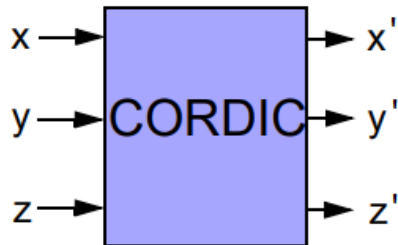
Other coefficient is the same way.

(c)

Bit-serial multiplier	Distributed arithmetic
Need $n(W_d + W_c - 1)$ cycles for n coefficients, but can be parallel	Need $W_d + W_{ROM}$ cycles (ROM is vector sum of n)
Need more adders and registers	Need more ROMs
Coefficient is fixed	Change data in ROMs can revise coefficient, more flexibility

3. Describe how to use CORDIC to efficiently compute the distance of a point (x, y) to the origin.

A CORDIC can be shown as following figure:



A point (x, y) will become (x', y') after pass through CORDIC, and angle z will become z'.

In **Circular Coordinate System, Vector Rotation Mode** will result in follow output:

$$\begin{aligned} x' &= K_n(x \cos z - y \sin z) \\ y' &= K_n(x \sin z + y \cos z) \\ z' &= 0 \end{aligned} \quad , \quad K_n = \prod_n (\sqrt{1 + 2^{-2i}})$$

In handout P91 ~ P94 , where $m = 1$, $\mu_i^2 = 1$, $s(m, i) = i$

The table below shows the rotation angles z that can be used for each iteration (i) of the CORDIC algorithm

i	2^{-i}	angle $\tan^{-1}(2^{-i})$	$\sqrt{1 + 2^{-2i}}$	scaling factor $\frac{1}{K_n} = \prod \frac{1}{\sqrt{1 + 2^{-2i}}}$
0	1	45°	1.4142	0.7071
1	1/2	26.5651°	1.1180	0.6325
2	1/4	14.0362°	1.0308	0.6136
3	1/8	7.1250°	1.0078	0.6088
4	1/16	3.5763°	1.0020	0.6076
5	1/32	1.7899°	1.0005	0.6074
6	1/64	0.8952°	1.0001	0.60725
7	1/128	0.4476°	1.00003	0.60726

So if we want to compute the distance of a point (x, y) to the origin, we can iteratively pass through CORDIC by the rotation angle 45°, 26.5651°, 14.0362° ...to make $y' = 0$, and the x' is the distance.

Indeed, to make $y' = 0$ is called **Angle Accumulation Mode (Vectoring Mode)** in handout P93, the output is:

$$\begin{aligned} x' &= K_n \sqrt{x^2 + y^2} \\ y' &= 0 \\ z' &= z + \tan^{-1}\left(\frac{y}{x}\right) \end{aligned}, K_n = \prod_n (\sqrt{1 + 2^{-2i}})$$

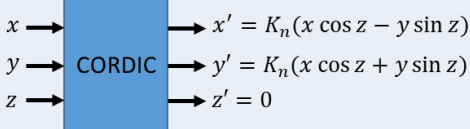
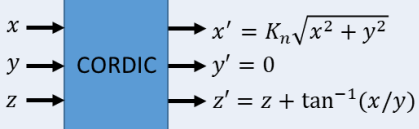
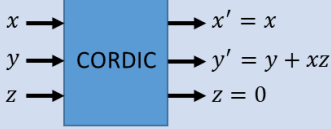
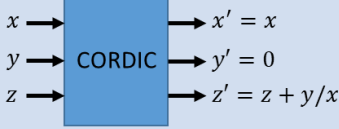
This is an example to compute the distance of a point $(1, \sqrt{3})$ to the origin:

$$\begin{aligned} x_{i+1} &= x_i - \mu_i y_i 2^{-i} = \sqrt{1 + 2^{-2i}} (x_i \cos \theta_i - y_i \sin \theta_i) \\ y_{i+1} &= \mu_i x_i 2^{-i} + y_i = \sqrt{1 + 2^{-2i}} (x_i \cos \theta_i + y_i \sin \theta_i) \end{aligned}$$

i	x_i	y_i	z_i	θ_i	μ_i $= -\text{sign}(x_i y_i)$
0	1	$\sqrt{3}$	0	45°	-1
1	2.732	0.732	45°	26.6°	-1
2	3.098	-0.634	71.6°	14°	1
3	3.257	0.141	57.6°	7.1°	-1
4	3.274	-0.267	64.7°	3.6°	1
5	3.291	-0.062	61.1°	1.8°	1
6	3.293	0.041	59.3°	0.9°	-1
7	3.293	-0.011	60.2°	0.4°	1

$$\begin{aligned} x_7 &= \prod_{i=0}^6 \sqrt{1 + 2^{-2i}} \sqrt{x^2 + y^2} = K_6 \sqrt{x^2 + y^2} = 3.293 \\ \sqrt{x^2 + y^2} &= \frac{1}{K_6} 3.293 = 0.60725 \times 3.293 = 1.999899 \end{aligned}$$

CORDIC operation can be summarized as below table:

Mode	Rotation $\mu_i = \text{sign}(z); z \rightarrow 0$	Vectoring $\mu_i = -\text{sign}(xy); y \rightarrow 0$
Circular $K_n = \prod_{i=0}^n (\sqrt{1 + 2^{-2i}})$		
Linear $K_n = 1$		
Hyperbolic $K_n = \prod_{i=0}^n (\sqrt{1 - 2^{-2i}})$	