LIST OF HLS BENCHMARKS

1) CONVOLUTIONAL LAYER (IN CNN)

Ref: A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network Using Facial Biometric," *IEEE Transactions on Consumer Electronics*, vol. 68, no. 3, pp. 291-306, 2022.

The image processing applications such as detection of curves, edges and objects, exploit filter/kernel (let say of size m×n) and generate convolved or filtered image. Further, process of designing secured convolutional layer IP core is presented here. Suppose an input image is of size P×Q (size of the input matrix is P×Q) where each pixel value is denoted by A_{ij} (i and j varying from 0 to P-1 and Q-1 respectively).

$$[I] = \begin{pmatrix} A_{00} & A_{01} & A_{02} & \cdots & A_{0(Q-1)} \\ A_{10} & A_{11} & A_{12} & \cdots & A_{1(Q-1)} \\ A_{20} & A_{21} & A_{22} & \cdots & A_{2(Q-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{(P-1)0} & A_{(P-1)1} & A_{(P-1)2} & \cdots & A_{(P-1)(Q-1)} \end{pmatrix}_{P\times Q}$$

Where, 'A' represents the intensity value corresponding to the pixels of input image. Further, a generic kernel/filter matrix of size m×n is denoted by$[H]_{mxn}$. In case of 3×3 size filters for curve detection, three kernel matrices [H] of size 3×3 are represented as follows:

$$[H_1] = \begin{pmatrix} h^1_{00} & h^1_{01} & h^1_{02} \\ h^1_{10} & h^1_{11} & h^1_{12} \\ h^1_{20} & h^1_{21} & h^1_{22} \end{pmatrix}_{3\times 3} \quad [H_2] = \begin{pmatrix} h^2_{00} & h^2_{01} & h^2_{02} \\ h^2_{10} & h^2_{11} & h^2_{12} \\ h^2_{20} & h^2_{21} & h^2_{22} \end{pmatrix}_{3\times 3} \quad [H_3] = \begin{pmatrix} h^3_{00} & h^3_{01} & h^3_{02} \\ h^3_{10} & h^3_{11} & h^3_{12} \\ h^3_{20} & h^3_{21} & h^3_{22} \end{pmatrix}_{3\times 3}$$

Where, $[H_1]$, $[H_2]$ and $[H_3]$ represent the curve detection kernels/filters. Further, pixel values of the kernel are represented by h^t_{pq}Where 'p, q' varies from 0 to 2 and't' denotes kernel/filter number.

In the proposed convolutional layer IP core, 'same convolution' is performed. In order to perform same convolution', the size of input matrix is augmented by adding zero-rows and zero-columns based on the following rule:

$$D = \frac{(S-1)}{2}$$

Where, 'S' is the size of kernel, i.e., S=3 for 3×3 kernels and 'D' is the number of zero rows/columns to be added on each side of input matrix (top, bottom, left and right). Therefore, post-padding size of the input matrix is increased by 2, as shown below:

$$[I] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & A_{00} & A_{01} & A_{02} & \cdots & A_{0(Q-1)} & 0 \\ 0 & A_{10} & A_{11} & A_{12} & \cdots & A_{1(Q-1)} & 0 \\ 0 & A_{20} & A_{21} & A_{22} & \cdots & A_{2(Q-1)} & 0 \\ 0 & \vdots & \vdots & \vdots & \ddots & \vdots & 0 \\ 0 & A_{(P-1)0} & A_{(P-1)1} & A_{(P-1)2} & \cdots & A_{(P-1)(Q-1)} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}_{M\times N}$$

Where, 'M×N' is the dimension of augmented input matrix which is equal of size (P+2)×(Q+2). Further, a generic representation of augmented matrix post applying padding using (14) is shown below:

$$[I]=\begin{pmatrix} I_{00} & I_{01} & I_{02} & \cdots & I_{0(N-1)} \\ I_{10} & I_{11} & I_{12} & \cdots & I_{1(N-1)} \\ I_{20} & I_{21} & I_{22} & \cdots & I_{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_{(M-1)0} & I_{(M-1)1} & I_{(M-1)2} & \cdots & I_{(M-1)(N-1)} \end{pmatrix}_{M\times N}$$

Pixel values from this matrix are denoted by I_{uv}, where 'u' and 'v' vary from 0 to M-1 and N-1 respectively. For an input matrix (augmented) of size M×N, and for 'K' filters of size m×n, size of the feature map can be calculated using the following equation:

$$\left[(M\text{-}m+1)\times(N\text{-}n+1)\right]\times K$$

Further, output matrix of the same convolution between input matrix and kernel matrix is denoted by [O] whose dimension are same as that of input matrix pre-padding (i.e., P×Q). Output pixel values of 2-D convolution are denoted by O_y^z, where 'y' varies from 0 to [(M-m+1)x(N-n+1)-1] and 'z' represents the number of output feature map corresponding to kernel.

Output value of each element/pixel corresponding to output feature map is denoted by O_y and is evaluated as follows:

$$O_y=\sum_{M,\,m\,=\,\text{lower value}}^{M,\,m\,=\,\text{upper value}}\left(\sum_{N,\,n\,=\,\text{lower value}}^{N,\,n\,=\,\text{upper value}} I_{MN}\times H_{mn}\right)$$

In the proposed approach two sliding window of kernel matrix simultaneously convolves over input matrix to compute two-pixel outputs in parallel. Two-pixel outputs are computed as follows:

$$1^{st}\text{ output}: O_0=\sum_{\substack{M=0\\m=0}}^{\substack{M=2\\m=2}}\left(\sum_{\substack{N=0\\n=0}}^{\substack{N=2\\n=2}} I_{MN}\times H_{mn}\right)\qquad 2^{nd}\text{ output}: O_1=\sum_{\substack{M=1\\m=0}}^{\substack{M=3\\m=2}}\left(\sum_{\substack{N=0\\n=0}}^{\substack{N=2\\n=2}} I_{MN}\times H_{mn}\right)$$

By expanding the equation (17) to compute both output pixel values (assuming for kernel 1) will be calculated as:

$$\begin{aligned} O_0^1=&\left[\left(I_{00}\times h_{00}^1\right)+\left(I_{01}\times h_{01}^1\right)+\left(I_{02}\times h_{02}^1\right)\right]+\\ &\left[\left(I_{10}\times h_{10}^1\right)+\left(I_{11}\times h_{11}^1\right)+\left(I_{12}\times h_{12}^1\right)\right]+\\ &\left[\left(I_{20}\times h_{20}^1\right)+\left(I_{21}\times h_{21}^1\right)+\left(I_{22}\times h_{22}^1\right)\right]\\ O_1^1=&\left[\left(I_{01}\times h_{00}^1\right)+\left(I_{02}\times h_{01}^1\right)+\left(I_{03}\times h_{02}^1\right)\right]+\\ &\left[\left(I_{11}\times h_{10}^1\right)+\left(I_{12}\times h_{11}^1\right)+\left(I_{13}\times h_{12}^1\right)\right]+\\ &\left[\left(I_{21}\times h_{20}^1\right)+\left(I_{22}\times h_{21}^1\right)+\left(I_{23}\times h_{22}^1\right)\right] \end{aligned}$$

Where, each product term in (18) is represented as ($I_{ab}\times h_{pq}^t$); where each pixel value in the input matrix and each kernel value in kernel matrix is represented by I_{ab} and h_{pq}^t respectively. During the computation of the first two-pixel values O_0^1 and O_1^1 using 3×3 kernel, values of 'a' and 'p' varies

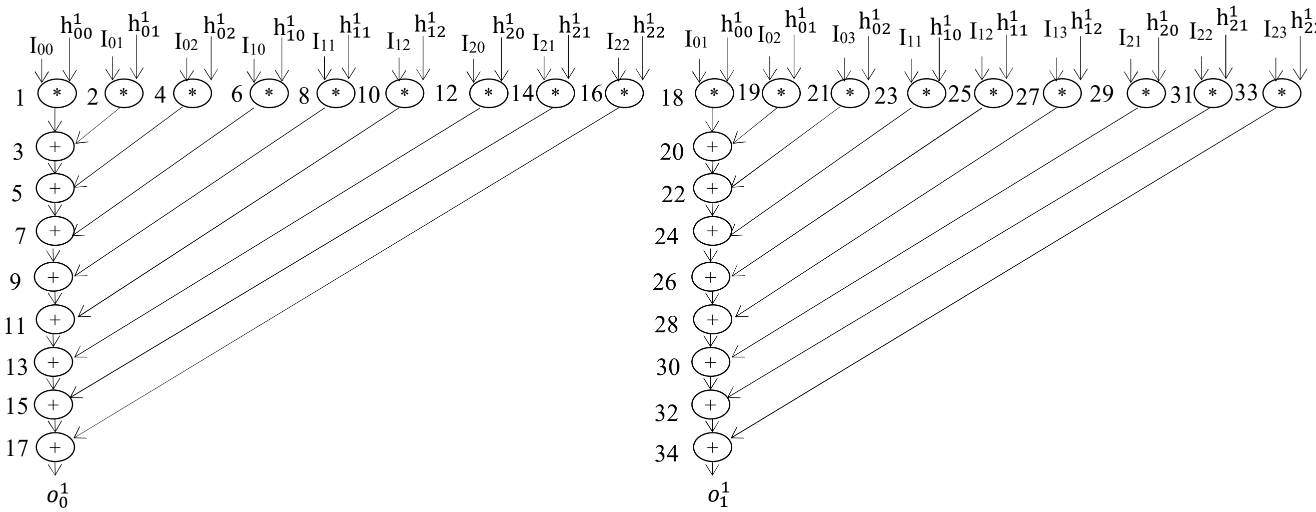

Fig. Data flow graph (DFG) of Convolutional Layer with filter kernel of size 3x3 and UF=2

2) DCT BENCHMARK

Ref: A. Sengupta "High-Level Synthesis based Methodologies for Hardware Security, Trust and IP Protection, " The Institute of Engineering and Technology (IET), 2024, ISBN-13: 978-1-83724-117-0.

Ref: A. Sengupta, S. P. Mohanty "IP Core Protection and Hardware-Assisted Security for Consumer Electronics", The Institute of Engineering and Technology (IET), 2019, Book ISBN: 978-1-78561-799-7, e-ISBN: 978-1-78561-800-0.

The generic equation of forward DCT can be expressed as :

$$X(m) = u(m)\sqrt{\frac{2}{N}}\sum_{i=0}^{N-1} x(i)\cos\left[\frac{(2i+1)m\pi}{2N}\right]$$

In the above expression,

$$u(m) = \begin{cases} \frac{1}{\sqrt{2}}; for\ m = 0 \\ 1\ \ ; for\ m\ \neq 0 \end{cases}; \text{ m = 0, 1, ..N-1}$$

x(i) is the input signal, X(m) is the output signal and N indicates number of data points. Therefore, for N=8 the first output signal X(0) can be expressed as follows:

$$X(0) = \frac{1}{\sqrt{2}}\sqrt{\frac{2}{8}}\sum_{i=0}^{7} x(i)\cos[0]$$

$$= \frac{1}{\sqrt{8}}\,x(0) + \frac{1}{\sqrt{8}}\,x(1) + \frac{1}{\sqrt{8}}\,x(2) + \frac{1}{\sqrt{8}}\,x(3) + \frac{1}{\sqrt{8}}\,x(4) + \frac{1}{\sqrt{8}}\,x(5) + \frac{1}{\sqrt{8}}\,x(6) + \frac{1}{\sqrt{8}}\,x(7)$$

$$= 0.3536\,x(0) + 0.3536\,x(1) + 0.3536\,x(2) + 0.3536\,x(3) + 0.3536\,x(4) + 0.3536\,x(5) + 0.3536\,x(6) + 0.3536\,x(7)$$

Similarly, for the same data point X(1) and X(2) can be expressed as:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}$$

Fig. Output signal of 8 point DCT in the form of matrix multiplication

$$X(1) = 1\sqrt{\frac{2}{8}}\sum_{i=0}^{7} x(i)\cos\left[\frac{(2i+1)1\pi}{16}\right]$$

$$= \frac{1}{2}\,x(0)\cos\left[\frac{\pi}{16}\right] + \frac{1}{2}\,x(1)\cos\left[\frac{3\pi}{16}\right] + \frac{1}{2}\,x(2)\cos\left[\frac{5\pi}{16}\right] + \frac{1}{2}\,x(3)\cos\left[\frac{7\pi}{16}\right] + \frac{1}{2}\,x(4)\cos\left[\frac{9\pi}{16}\right]$$

$$+ \frac{1}{2}\,x(5)\cos\left[\frac{11\pi}{16}\right] + \frac{1}{2}\,x(6)\cos\left[\frac{13\pi}{16}\right] + \frac{1}{2}\,x(7)\cos\left[\frac{15\pi}{16}\right]$$

$$= 0.4904\,x(0) + 0.4157\,x(1) + 0.2778\,x(2) + 0.0975\,x(3) + (-0.0975)\,x(4)$$

$$+ (-0.2778)\,x(5) + (-0.4157)\,x(6) + (-0.4904)\,x(7)$$

$$X(2) = 1\sqrt{\frac{2}{8}}\sum_{i=0}^{7} x(i)\cos\left[\frac{(2i+1)2\pi}{16}\right]$$

$$= \frac{1}{2}\,x(0)\cos\left[\frac{2\pi}{16}\right] + \frac{1}{2}\,x(1)\cos\left[\frac{6\pi}{16}\right] + \frac{1}{2}\,x(2)\cos\left[\frac{10\pi}{16}\right] + \frac{1}{2}\,x(3)\cos\left[\frac{14\pi}{16}\right] + \frac{1}{2}\,x(4)\cos\left[\frac{18\pi}{16}\right]$$

$$+ \frac{1}{2}\,x(5)\cos\left[\frac{22\pi}{16}\right] + \frac{1}{2}\,x(6)\cos\left[\frac{26\pi}{16}\right] + \frac{1}{2}\,x(7)\cos\left[\frac{30\pi}{16}\right]$$

$$= 0.4619\,x(0) + 0.1913\,x(1) + (-0.1913)\,x(2) + (-0.4619)\,x(3) + (-0.4619)\,x(4)$$

$$+ (-0.1913)\,x(5) + 0.1913\,x(6) + 0.4619\,x(7)$$

Similarly, X(3) to X(7) can be calculated.

We can represent this calculation in the form of matrix multiplication. Where the 8x1 output signal matrix can be derived by multiplying 8 point DCT coefficient matrix with 8x1 input signal matrix. The 8 point DCT coefficient can also be represented, where C_1 to C_7 indicates the DCT coefficients in ascending order (C_1 indicates the maximum positive DCT coefficient value and C_7 indicates the minimum positive DCT coefficient value). Based on matrix multiplication, the output signals X[0] to X[7] can be represented as:

X[0]=c4*x[0]+c4*x[1]+c4*x[2]+c4*x[3]+c4*x[4]+ c4*x[5]+ c4*x[6]+ c4*x[7]

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & -C_6 & C_2 \\ C_3 & -C_7 & -C_1 & -C_5 & C_5 & C_1 & C_7 & -C_3 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}$$

Fig. Another representation of output signal of 8 point DCT in the form of matrix multiplication

X[1]=c1*x[0]+c3*x[1]+c5*x[2]+c7*x[3]+(-c7)*x[4]+(- c5)*x[5]+ (-c3)*x[6]+ (-c1)*x[7]

X[2]=c2*x[0]+c6*x[1]+(-c6)*x[2]+(-c2)*x[3]+(-c2)*x[4]+(- c6)*x[5]+ c6*x[6]+ c2*x[7]

X[3]=c3*x[0]+(-c7)*x[1]+(-c1)*x[2]+(-c5)*x[3]+c5*x[4]+ c1*x[5]+ c7*x[6]+ (-c3)*x[7]

X[4]=c4*x[0]+(-c4)*x[1]+(-c4)*x[2]+c4*x[3]+c4*x[4]+ (-c4)*x[5]+ (-c4)*x[6]+ c4*x[7]

X[5]=c5*x[0]+(-c1)*x[1]+c7*x[2]+c3*x[3]+(-c3)*x[4]+ (-c7)*x[5]+ c1*x[6]+ (-c5)*x[7]

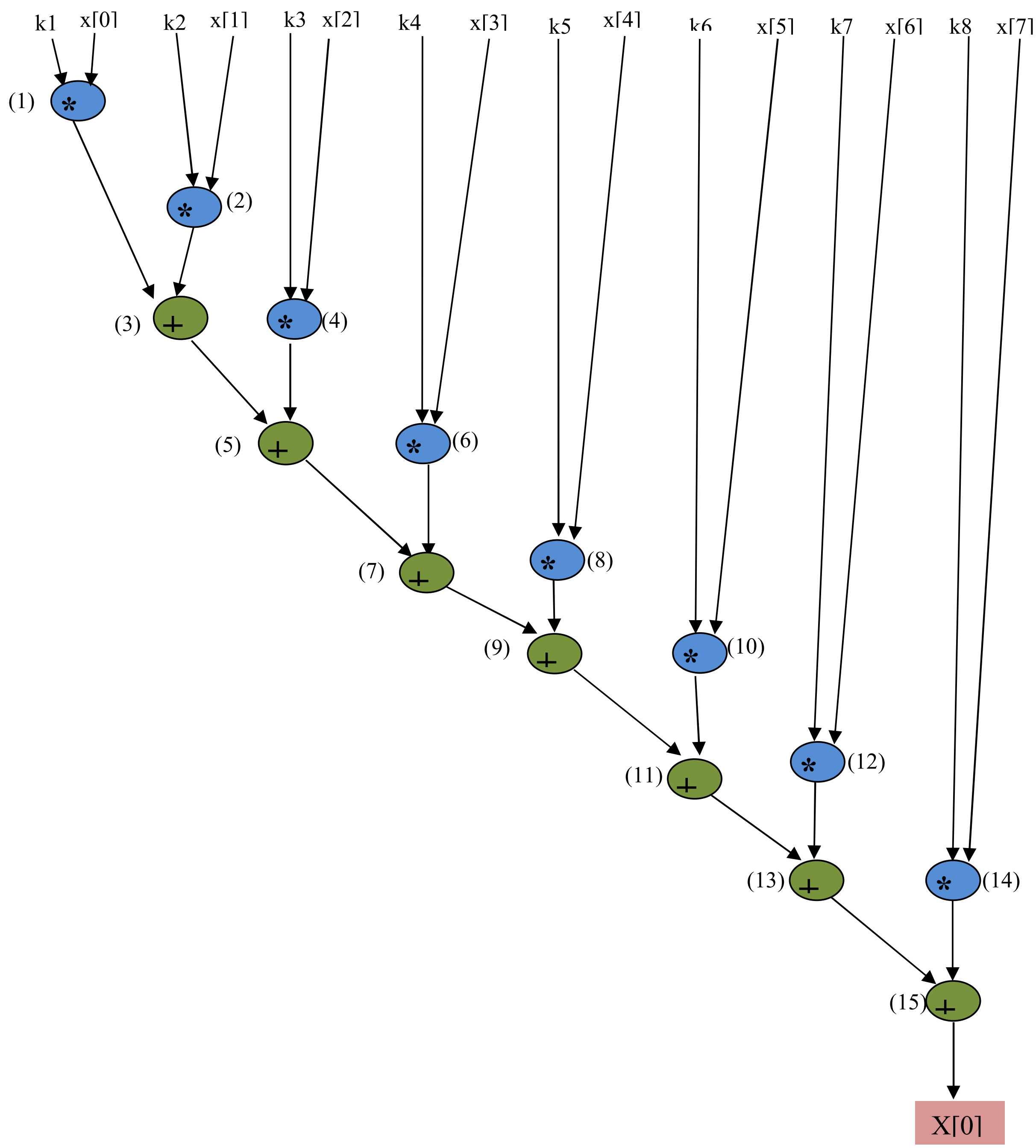

Fig. Data flow graph of an 8-point DCT

X[6]=c6*x[0]+(-c2)*x[1]+c2*x[2]+(-c6)*x[3]+(-c6)*x[4]+ c2*x[5]+ (-c2)*x[6]+ c6*x[7]

X[7]=c7*x[0]+(-c5)*x[1]+c3*x[2]+(-c1)*x[3]+c1*x[4]+ (-c3)*x[5]+ c5*x[6]+ (-c7)*x[7]

Generic equation of 8-Point DCT to compute 1st sample is:

X[0]=k1*x[0]+ k2*x[1]+ k3*x[2]+ k4*x[3]+ k5*x[4]+ k6*x[5]+ k7*x[6]+ k8*x[7]

The equation of 8-Point forward DCT consists of 8 multiplication operations and 7 addition operations. The equivalent DFG of 8-Point forward DCT, where primary inputs are two types, k1 to k8 indicates DCT coefficients and x[0] to x[7] indicates input signals primary inputs, blue nodes indicate multiplication operation and green nodes indicate addition operations, corresponding operation/node number are mentioned using integer value (1 to 15) and finally the output is shown in a red box. It can be observed that all the output signal X[0] to X[7] can be computed by changing the k1

3) FINITE IMPULSE RESPONSE (FIR) FILTER

Ref: A. Sengupta, S. P. Mohanty "IP Core Protection and Hardware-Assisted Security for Consumer Electronics", The Institute of Engineering and Technology (IET), 2019, Book ISBN: 978-1-78561-799-7, e-ISBN: 978-1-78561-800-0.

The generic equation of FIR filter can be expressed as follows:

$y(n) = \sum_{i=0}^{M} h_i * x(n-i)$,

where M is the filter order of a digital FIR, h_i is the FIR coefficient, x(n) is input impulse and y(n) is output impulse.

Using the above expression, a 7th order (8-tap) FIR filter can be represented as,

$$y(7) = h_0 * x(7) + h_1 * x(6) + h_2 * x(5) + h_3 * x(4) + h_4 * x(3) + h_5 * x(2) + h_6 * x(1) + h_7 * x(0)$$

The equation of 7th order FIR filter consists of 8 multiplication operations and 7 addition operations. The equivalent DFG of 7th order FIR filter, where primary inputs are shown in grey boxes, orange nodes indicate multiplication operation and blue nodes indicate addition operations, corresponding operation/node number are mentioned using integer value (1-15) and finally the output is shown in a green box.

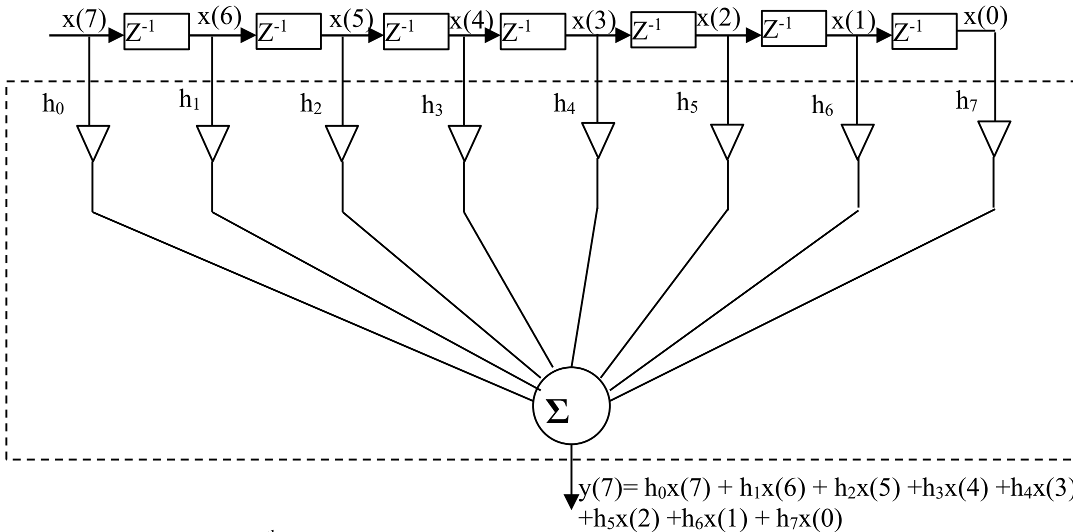

Fig. Block diagram of a 7th order FIR filter

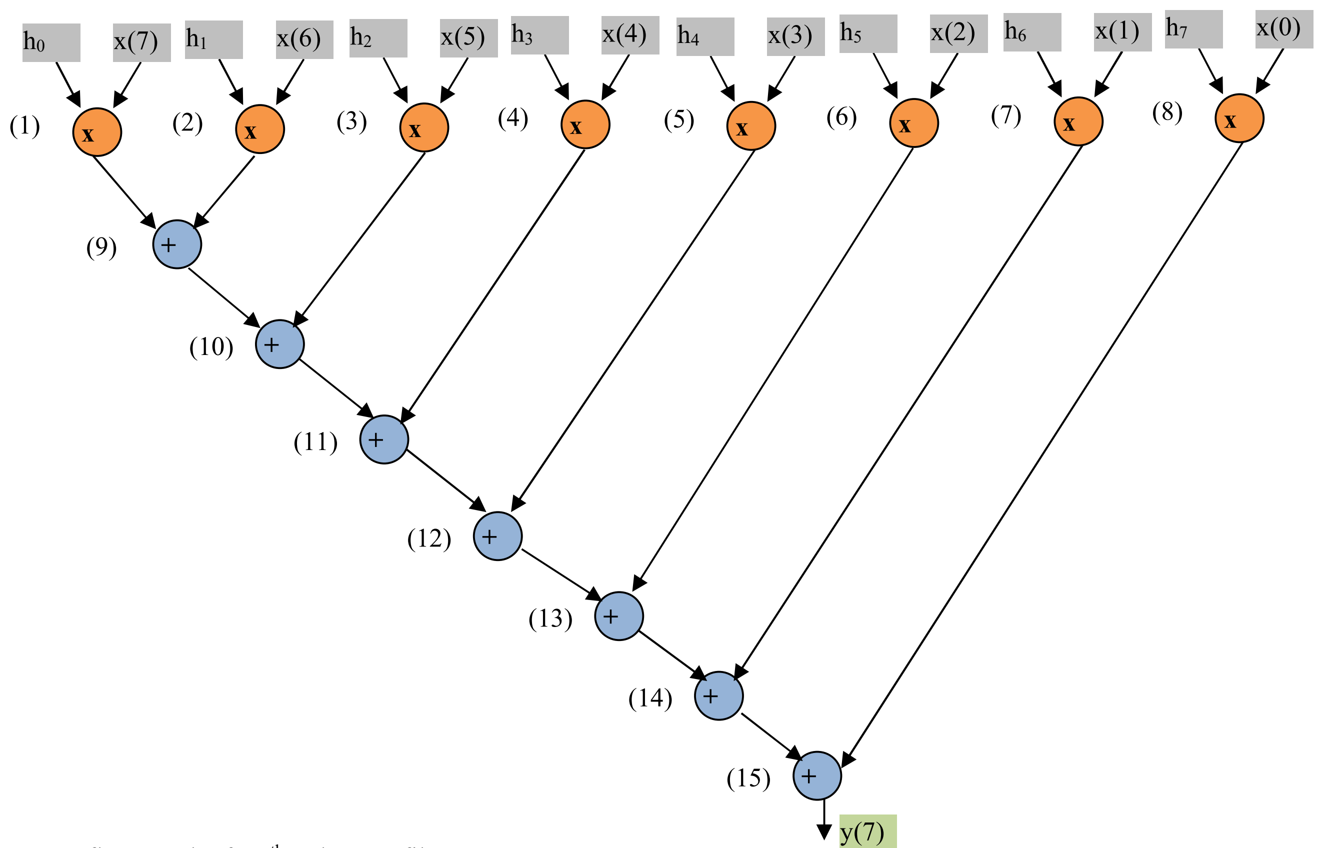

Fig. Data flow graph of a 7th order FIR filter

4) INFINITE IMPULSE RESPONSE (IIR) FILTER

Ref: A. Sengupta, S. P. Mohanty "IP Core Protection and Hardware-Assisted Security for Consumer Electronics", The Institute of Engineering and Technology (IET), 2019, Book ISBN: 978-1-78561-799-7, e-ISBN: 978-1-78561-800-0.

$0.9 \leq |H(e^{j\omega})| \leq 1$, for $0 \leq \omega \leq \pi/2$

$|H(e^{j\omega})| \leq 0.2$, for $3\pi/4 \leq \omega \leq \pi$

Solution: Given δ_1= 0.9, δ_2= 0.2, ω_1= $\pi/2$ and ω_2=$3\pi/4$; where, $\delta_1$and δ_2 are the parameters specifying *allowable pass-band* and *stop-band*, respectively.

Step 1: Determination of *edge frequencies* of analog filter:

Ω_1=2/T tan(ω_1/2) = 2/T tan(π/4)=2; Ω_2=2/T tan(ω_2/2) = 2/T tan(3π/8)=4.828

Therefore, $\Omega_2/\Omega_1 = 2.414$

Step 2: Determination of order of the filter:

Order of a filter (N) can be determined using the following equation,

$$N \geq \frac{1}{2}\frac{\log\frac{\left(\frac{1}{\delta_2^2}\right)-1}{\left(\frac{1}{\delta_1^2}\right)-1}}{\log\left(\Omega_2/\Omega_1\right)}$$

Using the above expression, we can obtain the following:

N ≥ ½ log(24/0.2346)/log(2.414) =2.626,

Thus, order N = 3.

Step 3: Determination of -3dB cut-off frequency:

Cut-off frequency (Ω_c) can be calculated using following equation,

$$\Omega_c = \frac{\Omega_1}{[\left(\frac{1}{\delta_1^2}\right)-1]^{1/2N}}$$

Using the above expression, we can obtain,

$\Omega_c = 2/[(1/0.9^2)-1]^{1/6} = 2.5467$

Step 4: Determination of transfer function (H(s)):

As N is odd, transfer function of IIR Butterworth can be derived using following equation,

$$H(s) = \frac{B_0\Omega_c}{s+c_0\Omega_c}\prod_{k=1}^{(N-1)/2}\frac{B_k\Omega_c^2}{s^2+b_k\Omega_c s+c_k\Omega_c^2}$$

Therefore, the *transfer function* of the current filter is,

$$H(s) = \left(\frac{B_0 \Omega_c}{s + c_0 \Omega_c}\right)\left(\frac{B_1 \Omega_c^2}{s^2 + b_1 \Omega_c s + c_1 \Omega_c^2}\right)$$

$b_1 = 2\sin(\pi/6) = 1$, $c_0 = 1$ and $c_1 = 1$

$B_0 B_1 = 1$, therefore, $B_0 = B_1 = 1$,

Therefore, $H(s) = \left(\frac{2.5467}{s+2.5467}\right)\left(\frac{6.4857}{s^2+2.5467s+6.4857}\right)$

Step 5: Determination of equivalent *system function (H(z))* using bilinear transformation:

For bilinear transformation, $s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$,

Therefore, $H(z) = \left(\frac{2.5467}{2\left(\frac{z-1}{z+1}\right)+2.5467}\right)\left(\frac{6.4857}{\left[2\left(\frac{z-1}{z+1}\right)\right]^2+5.0934\left(\frac{z-1}{z+1}\right)+6.4857}\right)$

$$H(z) = \left(\frac{16.5171(z+1)^3}{70.83z^3 + 31.1205z^2 + 27.2351z + 2.948}\right)$$

$$= \left(\frac{16.5171z^3 + 49.5513z^2 + 49.5513z + 16.5171}{70.83z^3 + 31.1205z^2 + 27.2351z + 2.948}\right)$$

$$H(z) = \frac{Y(z)}{X(z)} = \left(\frac{16.5171z^3 + 49.5513z^2 + 49.5513z + 16.5171}{70.83z^3 + 31.1205z^2 + 27.2351z + 2.948}\right)$$

$$= \left(\frac{0.2332 + 0.4664z^{-1} + 0.4664z^{-2} + 0.2332z^{-3}}{1 + 0.4394z^{-1} + 0.3845z^{-2} + 0.0416z^{-3}}\right)$$

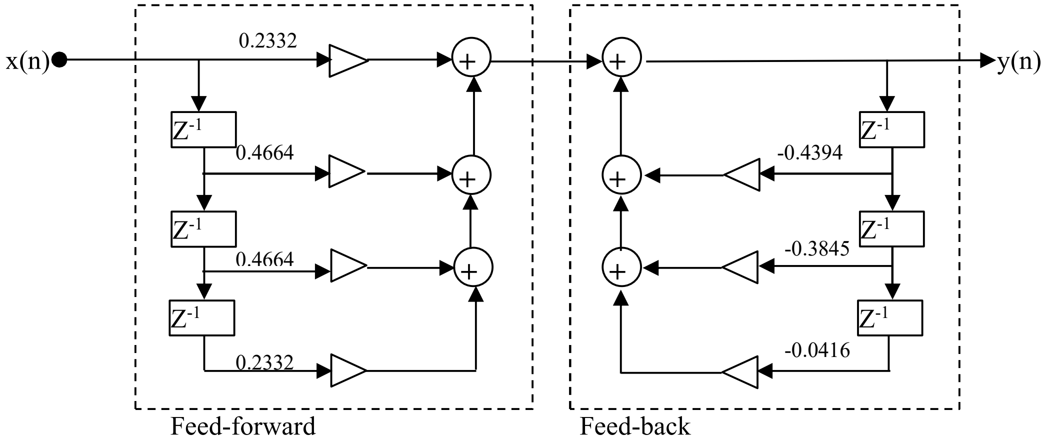

Fig. Block diagram of a 3rd order IIR Digital Butterworth Filter

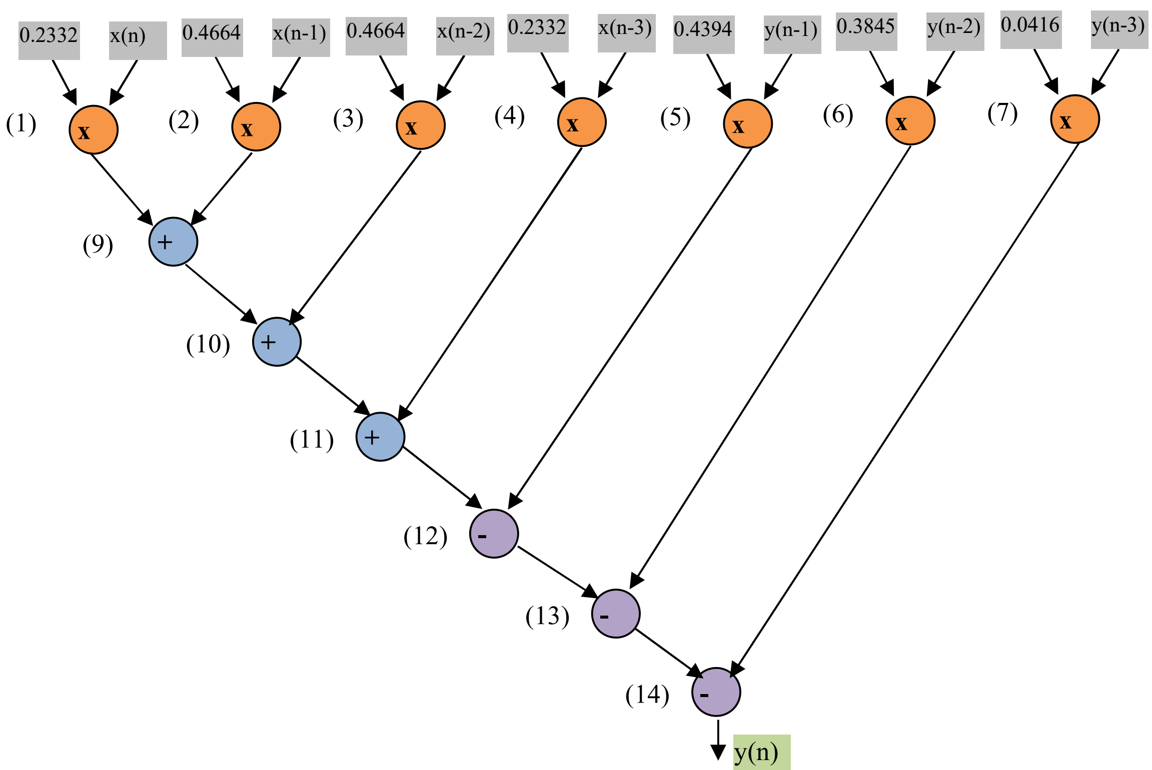

Fig. Data flow graph of an IIR Butterworth filter

5) MESA

Ref: A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," *IEEE Embedded Systems Letters*, 2024, doi: 10.1109/LES.2024.3416422.

Ref: University of California Santa Barbara Express Group, accessed on May 2025, Available: http://express.ece.ucsb.edu/benchmark/.

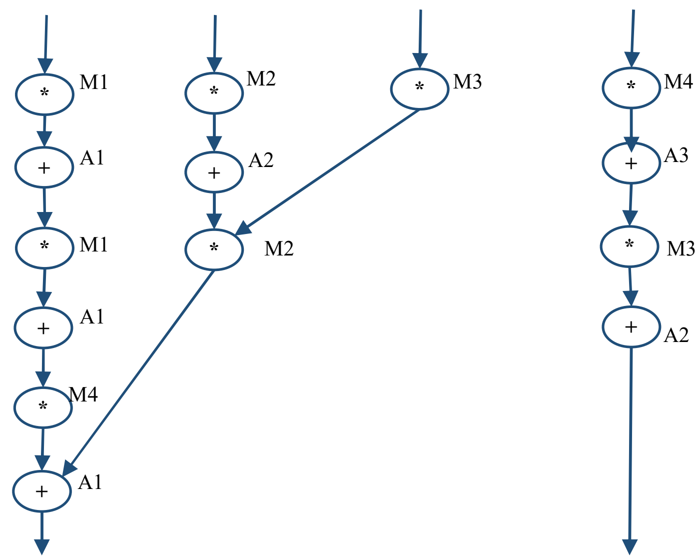

Fig. Data flow graph of MESA Horner Bezier

6) SHARPENING FILTER (SF)

Ref: A. Sengupta "Secured Hardware Accelerators for DSP and Image Processing Applications", The Institute of Engineering and Technology (IET), 2021, Print: 978-1-83953-306-8, eBook: 978-1-83953-307-5.

$$O_0 = \sum_{a=0,p=0}^{a=4,p=4} \left[(I_{a0} \times h_{p0}) + (I_{a1} \times h_{p1}) + (I_{a2} \times h_{p2}) + (I_{a3} \times h_{p3}) + (I_{a4} \times h_{p4}) \right]$$

Above equation can be expanded for computing 1st pixel value O_0 of output image using 5×5 filters.

$$\begin{aligned} O_0 = & \left[(I_{00} \times h_{00}) + (I_{01} \times h_{01}) + (I_{02} \times h_{02}) + (I_{03} \times h_{03}) + (I_{04} \times h_{04}) \right] \\ & + \left[(I_{10} \times h_{10}) + (I_{11} \times h_{11}) + (I_{12} \times h_{12}) + (I_{13} \times h_{13}) + (I_{14} \times h_{14}) \right] \\ & + \left[(I_{20} \times h_{20}) + (I_{21} \times h_{21}) + (I_{22} \times h_{22}) + (I_{23} \times h_{23}) + (I_{24} \times h_{24}) \right] \\ & + \left[(I_{30} \times h_{30}) + (I_{31} \times h_{31}) + (I_{32} \times h_{32}) + (I_{33} \times h_{33}) + (I_{34} \times h_{34}) \right] \\ & + \left[(I_{40} \times h_{40}) + (I_{41} \times h_{41}) + (I_{42} \times h_{42}) + (I_{43} \times h_{43}) + (I_{44} \times h_{44}) \right] \end{aligned}$$

The 3×3 kernel for sharpening filter is shown below:

$$H^S = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}_{3\times 3}$$

Using the above filter kernel and the equation for computing 1st and 2nd pixel value O_0 and O_1, the sharpening filter transfer function can be written as:

$$O_0 = [(I_{00} + I_{01} + I_{02} + I_{10} + I_{12} + I_{20} + I_{21} + I_{22}) \times (-1)] + (I_{11} \times 9)$$
$$O_1 = [(I_{01} + I_{02} + I_{03} + I_{11} + I_{13} + I_{21} + I_{22} + I_{23}) \times (-1)] + (I_{12} \times 9)$$

The respective CDFG is shown below:

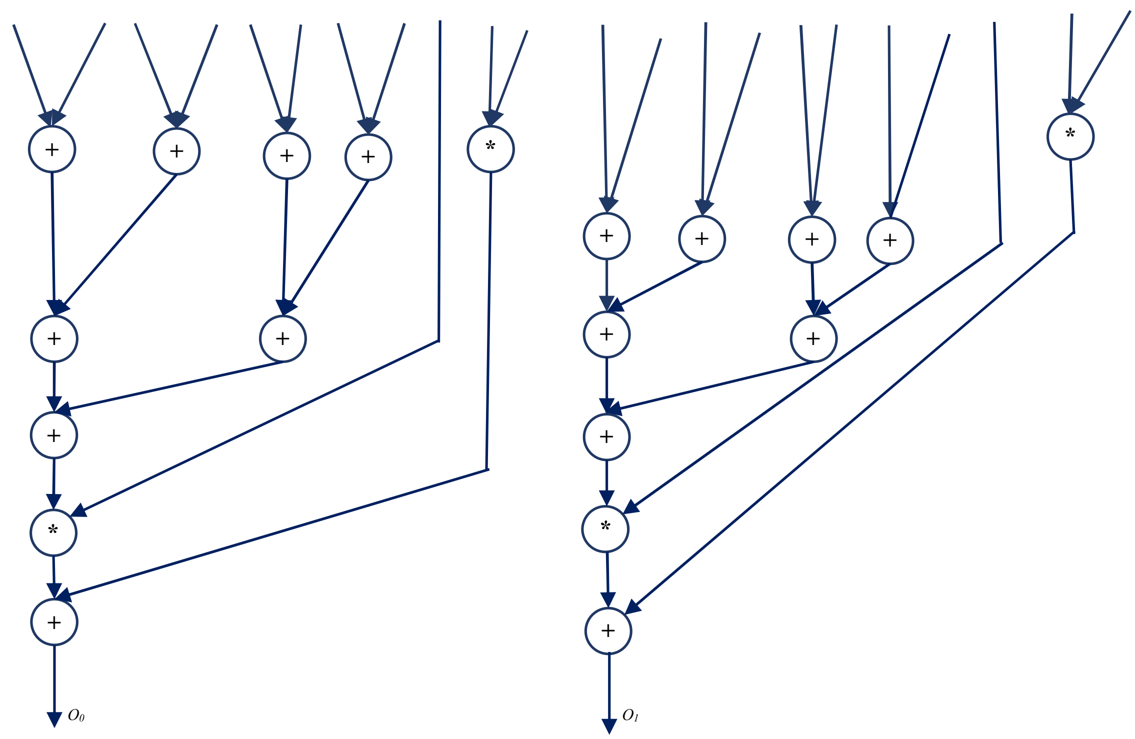

Fig. CDFG of sharpening filter