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).

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

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:

$$[H_1]=\begin{pmatrix} h_{00}^1 & h_{01}^1 & h_{02}^1 \\ h_{10}^1 & h_{11}^1 & h_{12}^1 \\ h_{20}^1 & h_{21}^1 & h_{22}^1 \end{pmatrix}_{3\times3} \qquad [H_2]=\begin{pmatrix} h_{00}^2 & h_{01}^2 & h_{02}^2 \\ h_{10}^2 & h_{11}^2 & h_{12}^2 \\ h_{20}^2 & h_{21}^2 & h_{22}^2 \end{pmatrix}_{3\times3} \quad [H_3]=\begin{pmatrix} h_{00}^3 & h_{01}^3 & h_{02}^3 \\ h_{10}^3 & h_{11}^3 & h_{12}^3 \\ h_{20}^3 & h_{21}^3 & h_{22}^3 \end{pmatrix}_{3\times3}$$

Where, $[H_1]$, $[H_2]$ and $[H_3]$ represent the curve detection kernels/filters. Further, pixel values of the kernel are represented by h_{pq}^tWhere '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 [1]:

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

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 :

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

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:

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

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:

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

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

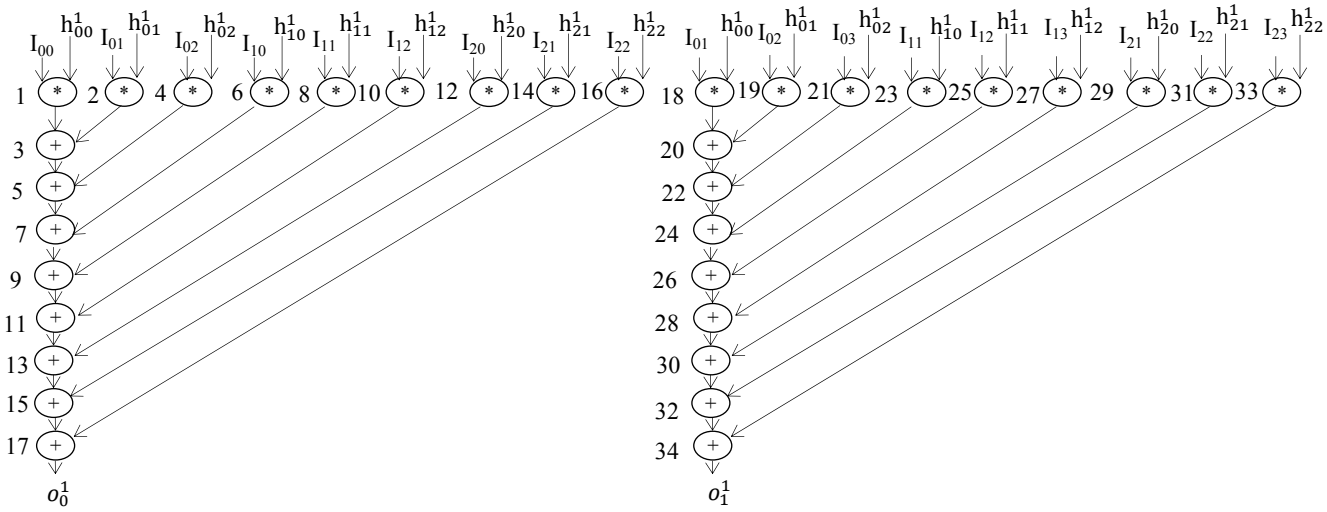
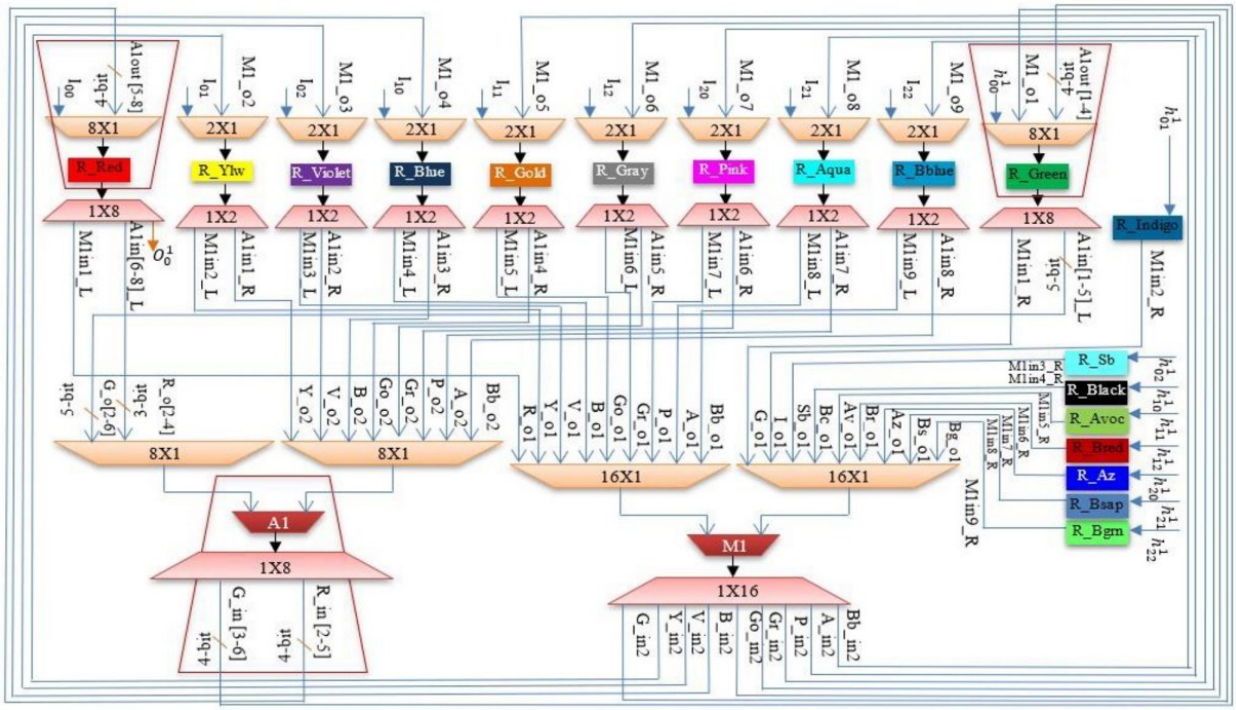
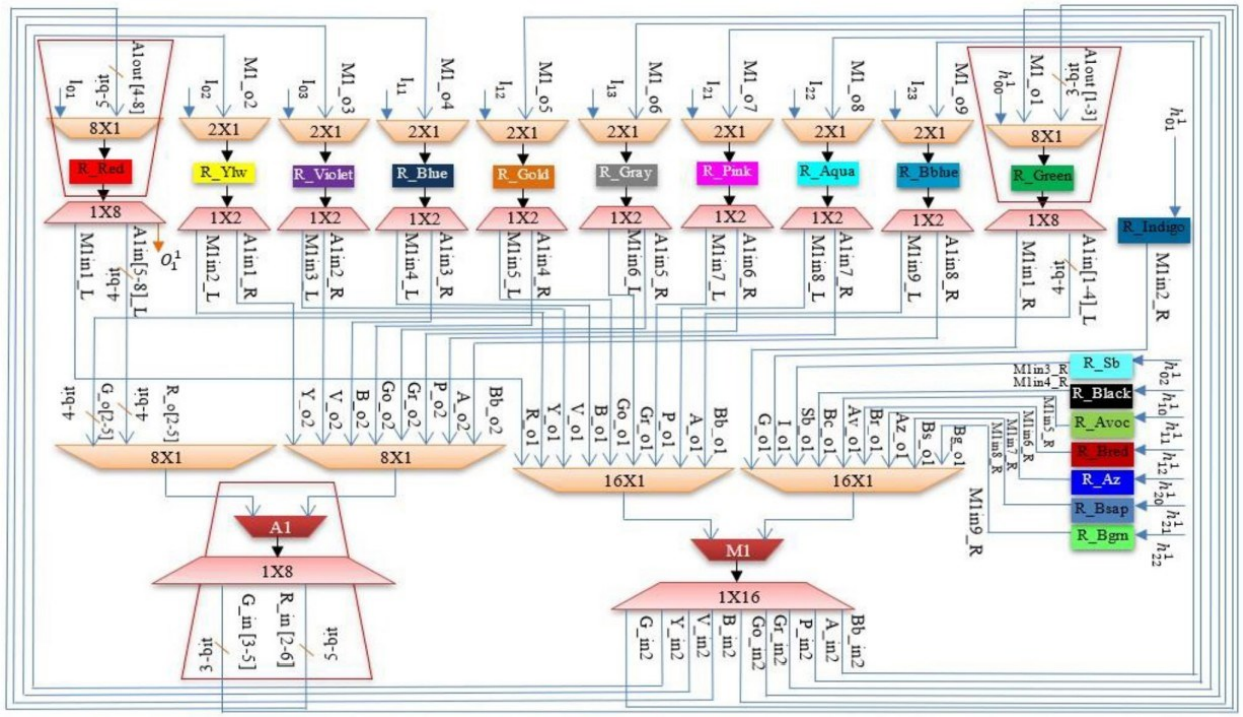


Fig. Data flow graph (DFG) of Convolutional Layer with filter kernel of size 3x3 and UF=2

The HLS generated register transfer level (RTL) datapath corresponding to CNN convolutional layer output pixel o_0^1 and o_1^1 is shown below:




8X1
2X1
1X8
1X2
16X1
1X16
R_Red
R_Ylw
R_Violet
R_Blue
R_Gold
R_Gray
R_Pink
R_Aqua
R_Bblue
R_Green
R_Indigo
R_Sb
R_Black
R_Avoc
R_Bred
R_Az
R_Bsap
R_Bgm
A1
M1

2) DCT BENCHMARK

Ref: A. Sengupta, R. Chaurasia and A. Anshul, "Robust Security of Hardware Accelerators Using Protein Molecular Biometric Signature and Facial Biometric Encryption Key," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 6, pp. 826-839, June 2023, doi: 10.1109/TVLSI.2023.3265559.

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 (8-point DCT) the first output signal X(0) can be expressed as follows:

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

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

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

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

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

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

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

$$= \frac{1}{2}\,x(0)\cos\left[\frac{\pi}{16}\right] + \frac{1}{2}\,x(1)\cos\left[\frac{3\pi}{16}\right] + \frac{1}{2}\,x(2)\cos\left[\frac{5\pi}{16}\right] + \frac{1}{2}\,x(3)\cos\left[\frac{7\pi}{16}\right] + \frac{1}{2}\,x(4)\cos\left[\frac{9\pi}{16}\right] + \frac{1}{2}\,x(5)\cos\left[\frac{11\pi}{16}\right] + \frac{1}{2}\,x(6)\cos\left[\frac{13\pi}{16}\right] + \frac{1}{2}\,x(7)\cos\left[\frac{15\pi}{16}\right]$$

$$= 0.4904\,x(0) + 0.4157\,x(1) + 0.2778\,x(2) + 0.0975\,x(3) + (-0.0975)\,x(4) + (-0.2778)\,x(5) + (-0.4157)\,x(6) + (-0.4904)\,x(7)$$

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

$$= \frac{1}{2}\,x(0)\cos\left[\frac{2\pi}{16}\right] + \frac{1}{2}\,x(1)\cos\left[\frac{6\pi}{16}\right] + \frac{1}{2}\,x(2)\cos\left[\frac{10\pi}{16}\right] + \frac{1}{2}\,x(3)\cos\left[\frac{14\pi}{16}\right] + \frac{1}{2}\,x(4)\cos\left[\frac{18\pi}{16}\right] + \frac{1}{2}\,x(5)\cos\left[\frac{22\pi}{16}\right] + \frac{1}{2}\,x(6)\cos\left[\frac{26\pi}{16}\right] + \frac{1}{2}\,x(7)\cos\left[\frac{30\pi}{16}\right]$$

$$= 0.4619\,x(0) + 0.1913\,x(1) + (-0.1913)\,x(2) + (-0.4619)\,x(3) + (-0.4619)\,x(4) + (-0.1913)\,x(5) + 0.1913\,x(6) + 0.4619\,x(7)$$

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

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

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

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 [2], [3], [4]:

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]

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]

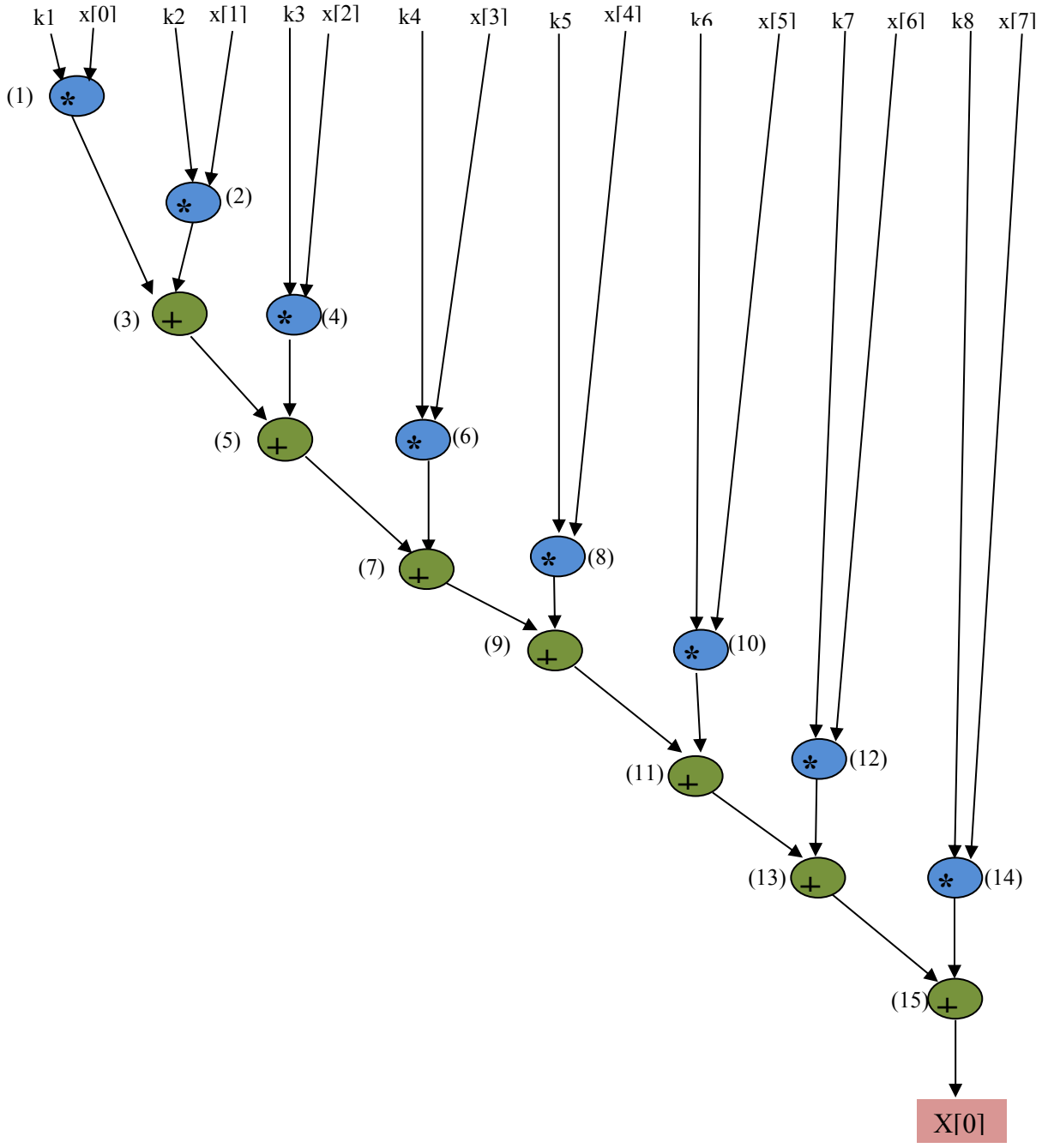


Fig. Data flow graph of an 8-point DCT

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]

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.

Similarly, for N=4 (4-point DCT) the first output signal T(0) can be expressed as follows:

$$T(0) = \frac{1}{\sqrt{2}}\sqrt{\frac{2}{4}}\sum_{i=0}^{i=3} t(i)\cos[0]$$

$$= \frac{1}{2}t(0) + \frac{1}{2}t(1) + \frac{1}{2}t(2) + \frac{1}{2}t(3)$$

$$= 0.5t(0) + 0.5t(1) + 0.5t(2) + 0.5t(3)$$

Similarly, the output signal values for T (1) to T (3) are computed.

$$T(1) = 0.65t(0) + 0.27t(1) - 0.27t(2) - 0.65t(3) \quad T(2) = 0.5t(0) - 0.5t(1) - 0.5t(2) + 0.5t(3)$$
$$T(3) = 0.27t(0) - 0.65t(1) + 0.65t(2) - 0.27t(3)$$

The final matrix expressing the output signals of 4-point DCT in matric multiplication form is given below.

$$\begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix} \begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \end{bmatrix}$$

The above matrix multiplication form expression can also be expressed as:

$$\begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \end{bmatrix} = \begin{bmatrix} C_1 & C_1 & C_1 & C_1 \\ C_2 & C_3 & -C_3 & -C_2 \\ C_1 & -C_1 & -C_1 & C_1 \\ C_3 & -C_2 & C_2 & -C_3 \end{bmatrix} \begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \end{bmatrix}$$

The general equation of 4-point DCT for computing the first output signal (T (0)) is expressed as (k_1 to k_4 are DCT coefficients):

$$T(0) = k_1 * t(0) + k_2 * t(1) + k_3 * t(2) + k_4 * t(3)$$

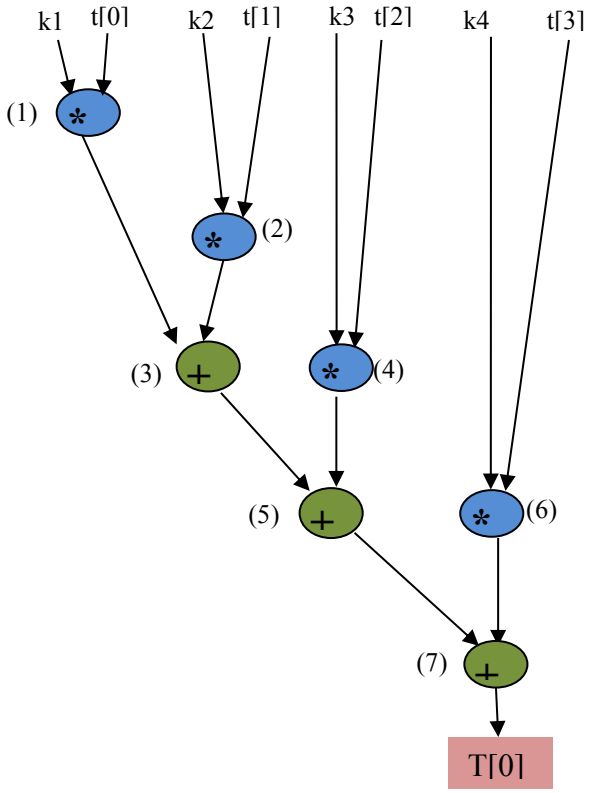
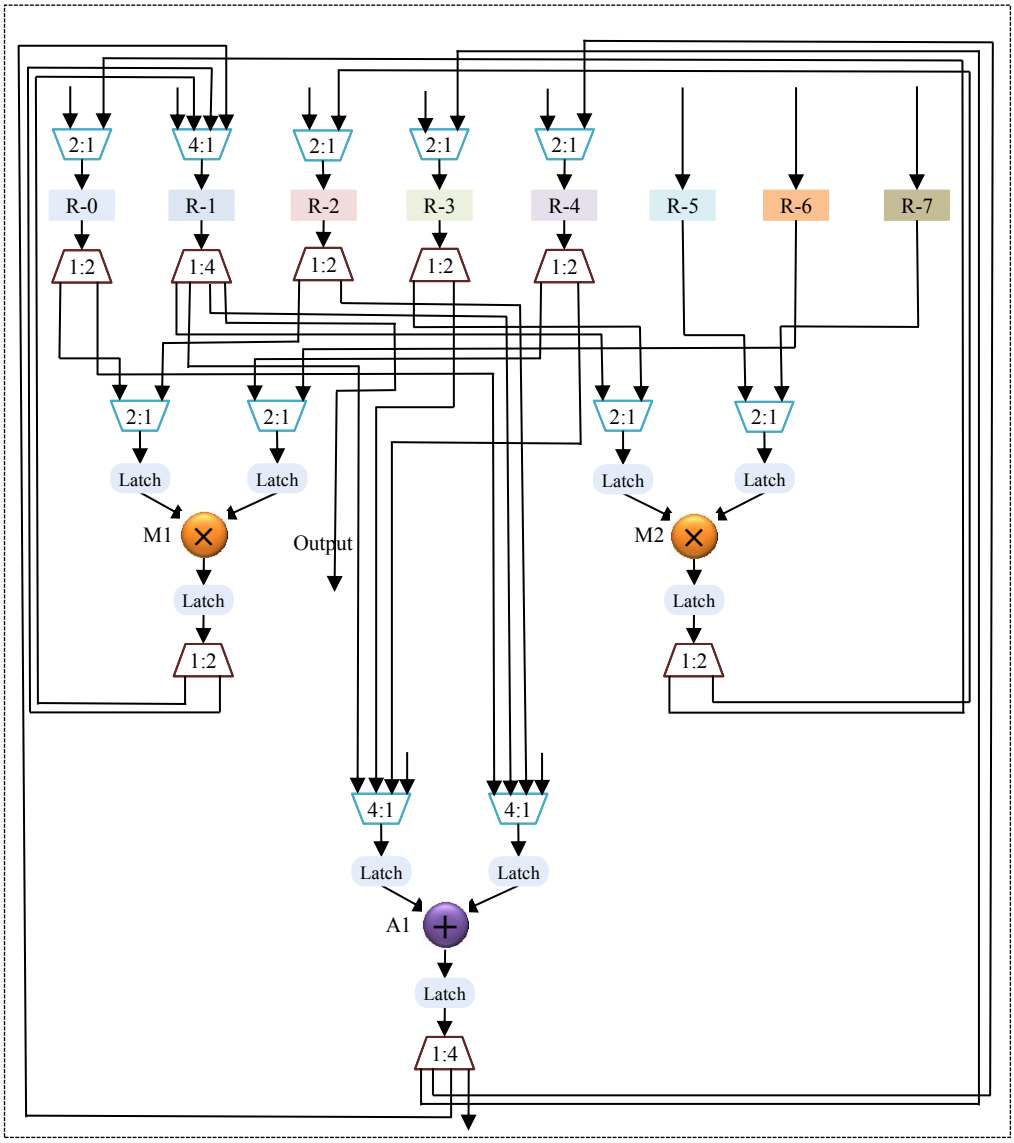


Fig. Data flow graph of 4-point DCT

The HLS generated register transfer level (RTL) datapath corresponding to 4-point DCT is shown below:



3) FINITE IMPULSE RESPONSE (FIR) FILTER

Ref: A. Sengupta, D. Kachave and D. Roy, "Low Cost Functional Obfuscation of Reusable IP Cores Used in CE Hardware Through Robust Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 604-616, April 2019, doi: 10.1109/TCAD.2018.2818720.

Ref: A. Sengupta and M. Rathor, "Enhanced Security of DSP Circuits Using Multi-Key Based Structural Obfuscation and Physical-Level Watermarking for Consumer Electronics Systems," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 163-172, May 2020, doi: 10.1109/TCE.2020.2972808.

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 [5], [6], [7], [25], [26].

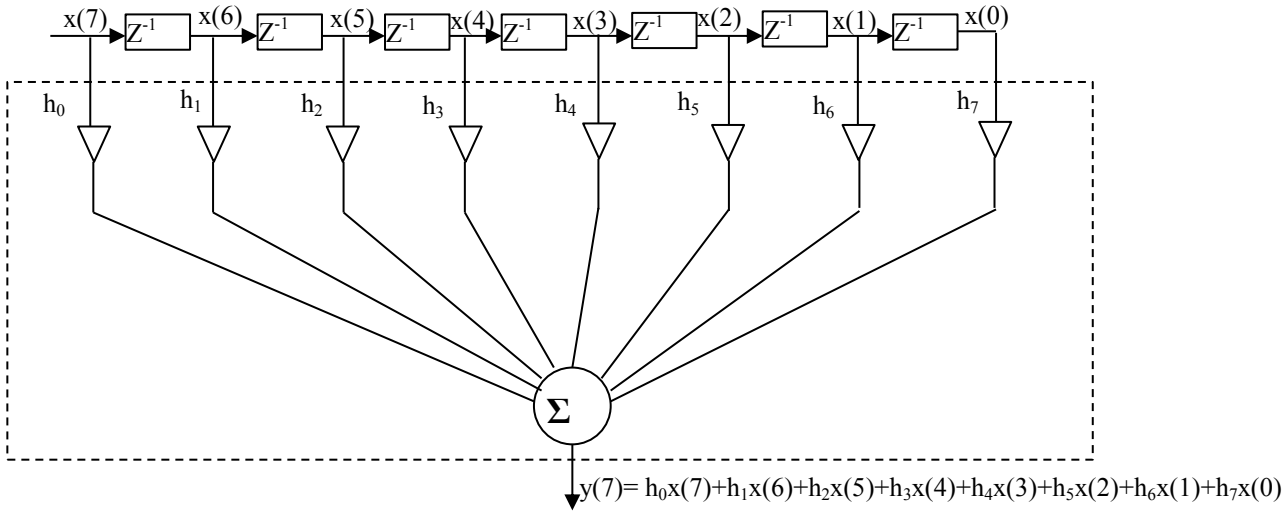


Fig. Block diagram of a 7th order FIR filter

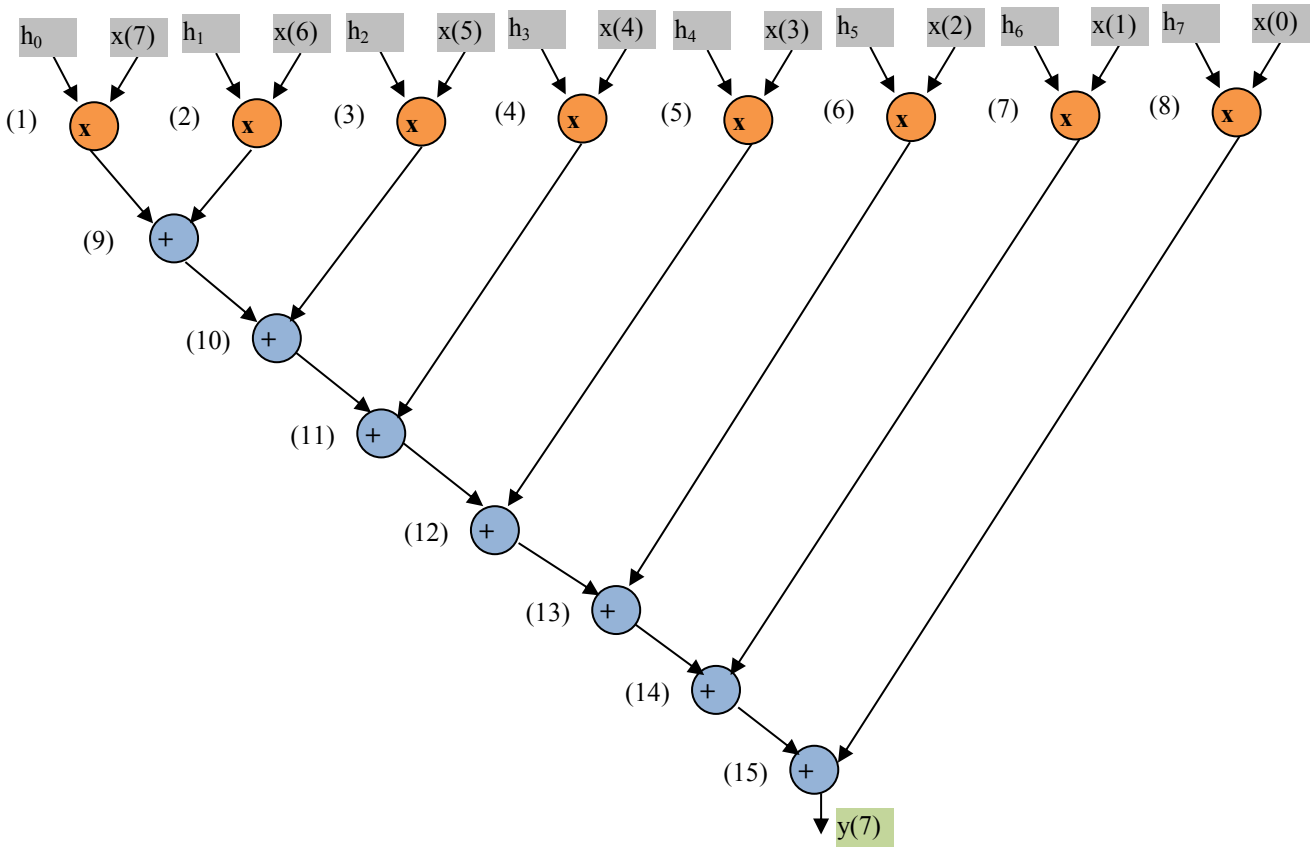
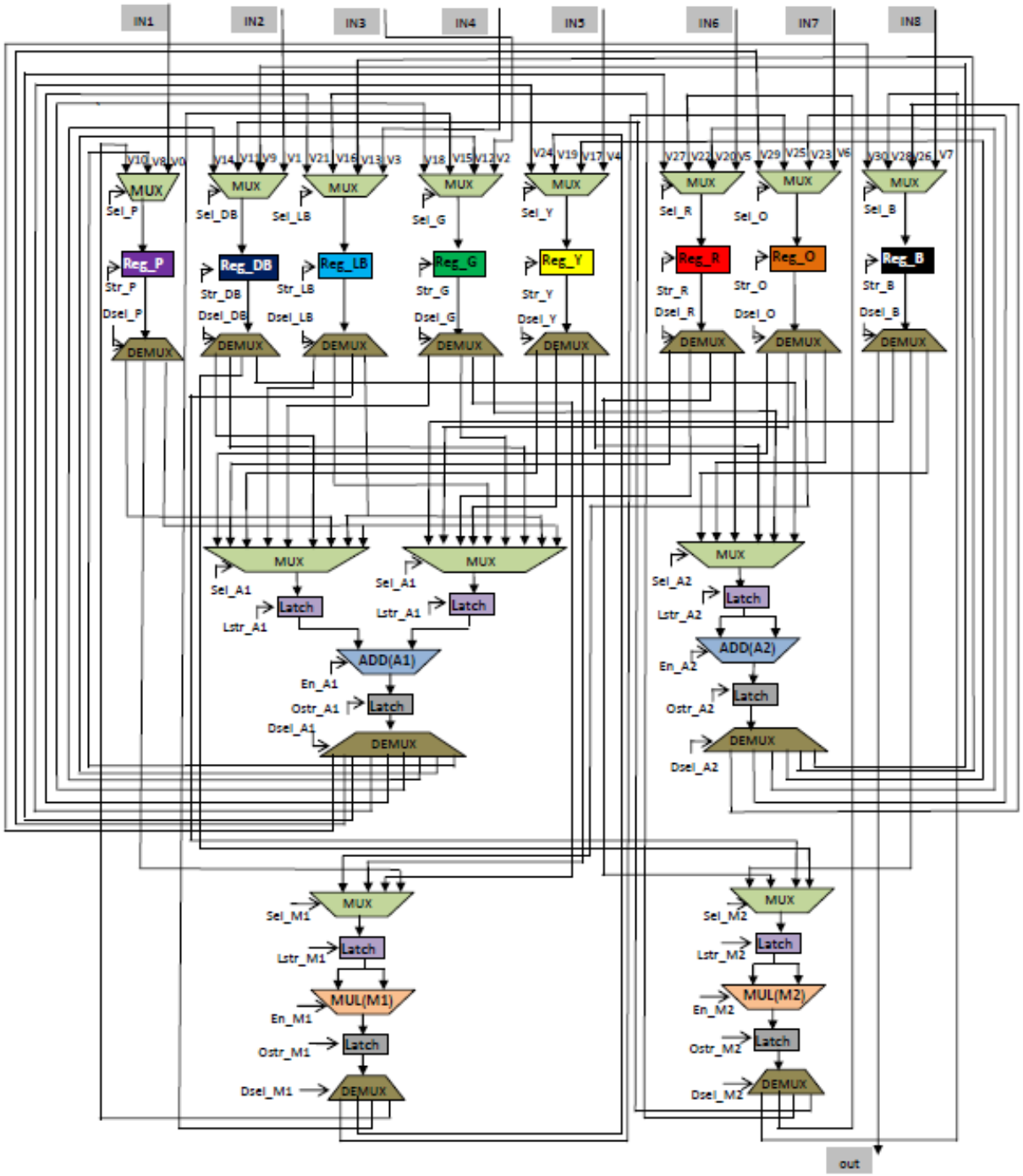


Fig. Data flow graph of a 7th order FIR filter

The HLS generated register transfer level (RTL) datapath corresponding to FIR filter is shown below:



4) INFINITE IMPULSE RESPONSE (IIR) FILTER

Ref: M. Rathor, A. Anshul and A. Sengupta, "Securing Reusable IP Cores Using Voice Biometric Based Watermark," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 2735-2749, July-Aug. 2024, doi: 10.1109/TDSC.2023.3315780.

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, Ω_2/Ω_1 = 2.414

Step 2: Determination of order of the filter:

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

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

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,

Ω_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 $c_1 = 1$

$B_0B_1 = 1$, 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, $H(z) = \left(\frac{2.5467}{2\left(\frac{z-1}{z+1}\right)+2.5467}\right)\left(\frac{6.4857}{\left[2\left(\frac{z-1}{z+1}\right)\right]^2+5.0934\left(\frac{z-1}{z+1}\right)+6.4857}\right)$

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

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

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

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

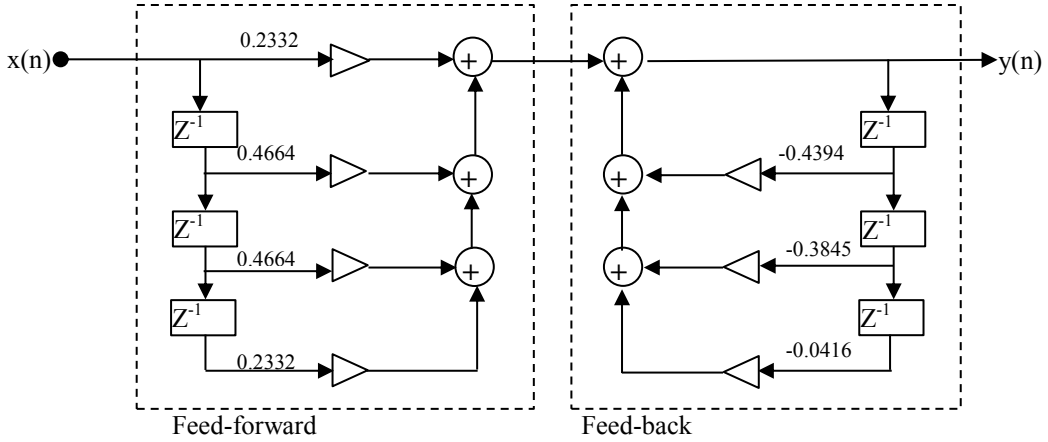


Fig. Block diagram of a 3rd order IIR Digital Butterworth Filter

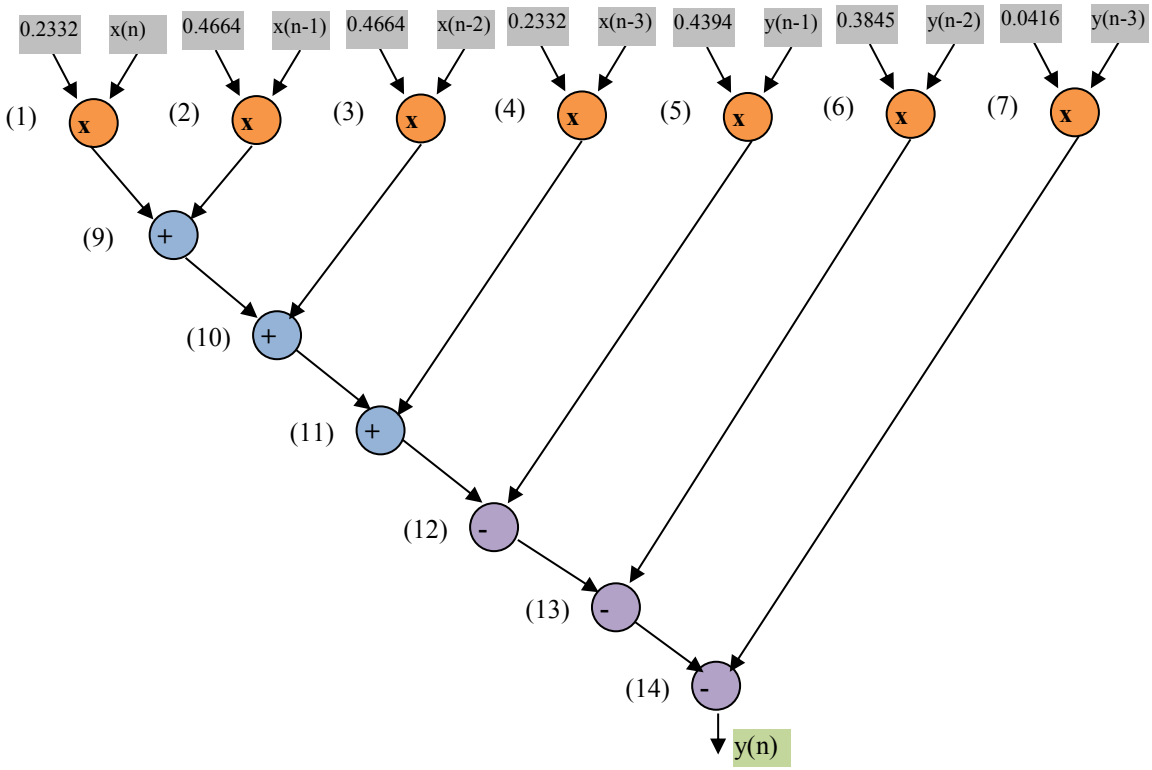
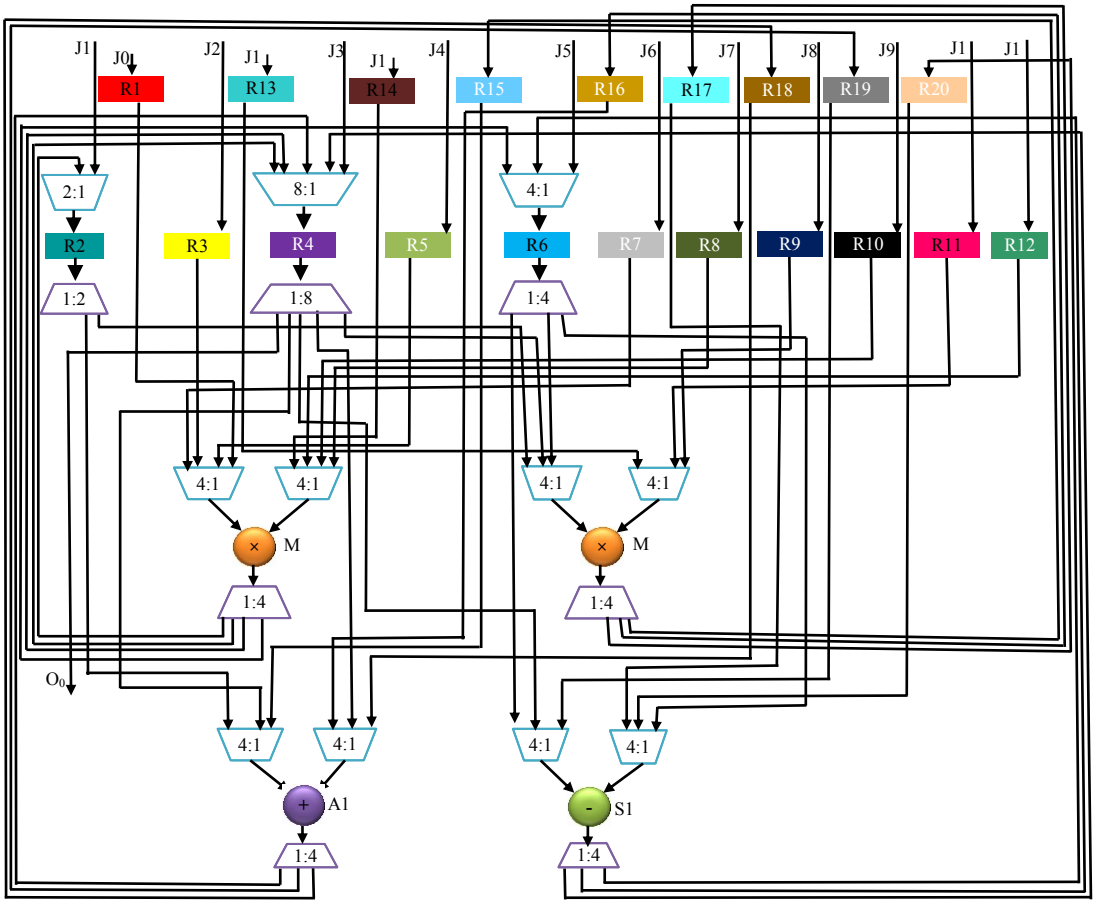


Fig. Data flow graph of an IIR Butterworth filter

The HLS generated register transfer level (RTL) datapath corresponding to IIR filter is shown below [8], [9]:



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: A. Sengupta and D. Roy, "Antipiracy-Aware IP Chipset Design for CE Devices: A Robust Watermarking Approach," *IEEE Consumer Electronics Magazine*, vol. 6, no. 2, pp. 118-124, April 2017, doi: 10.1109/MCE.2016.2640622.

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

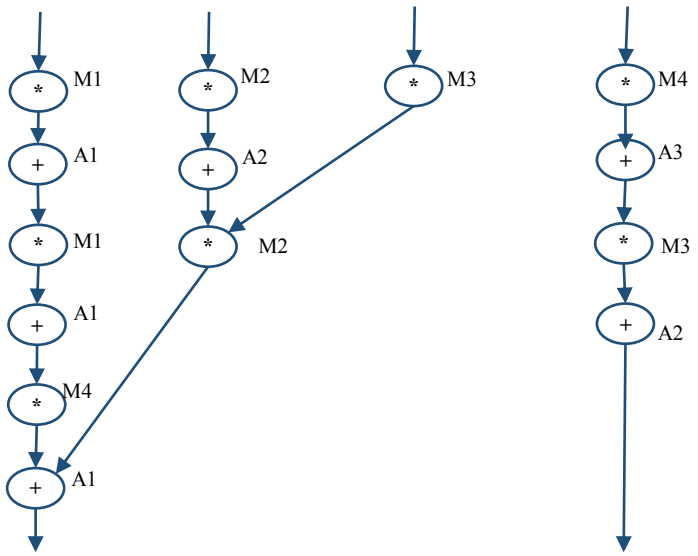
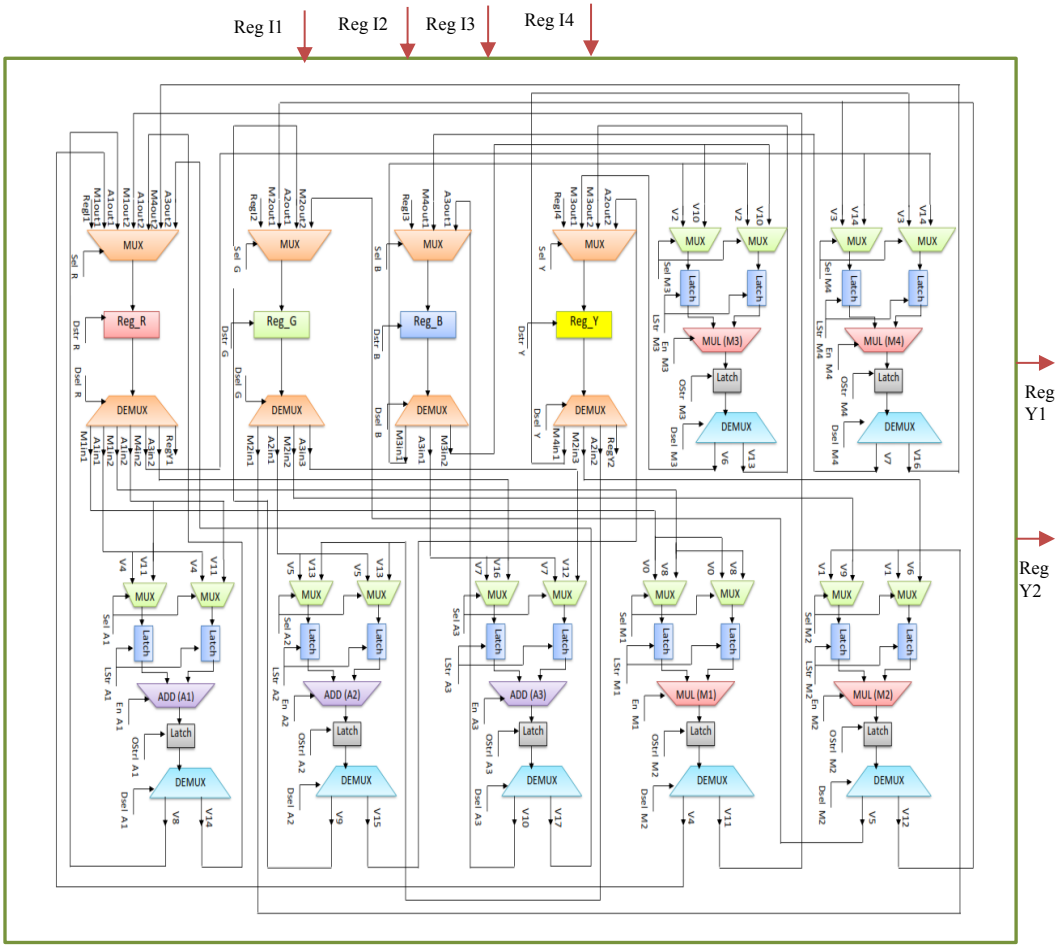


Fig. Data flow graph of MESA Horner Bezier

The HLS generated register transfer level (RTL) datapath corresponding to MESA Horner Bezier is shown below [10], [11], [12]:



6) SHARPENING FILTER (SF)

Ref: A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760.

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.

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

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

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

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

$$h^s=\begin{bmatrix}-1 & -1 & -1\\ -1 & 9 & -1\\ -1 & -1 & -1\end{bmatrix}_{3\times3}$$

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:

$$\begin{aligned}O_0 &= \left[\left(I_{00}+I_{01}+I_{02}+I_{10}+I_{12}+I_{20}+I_{21}+I_{22}\right)\times\left(-1\right)\right]+\left(I_{11}\times9\right)\\ O_1 &= \left[\left(I_{01}+I_{02}+I_{03}+I_{11}+I_{13}+I_{21}+I_{22}+I_{23}\right)\times\left(-1\right)\right]+\left(I_{12}\times9\right)\end{aligned}$$

The respective CDFG is shown below [13], [14]:

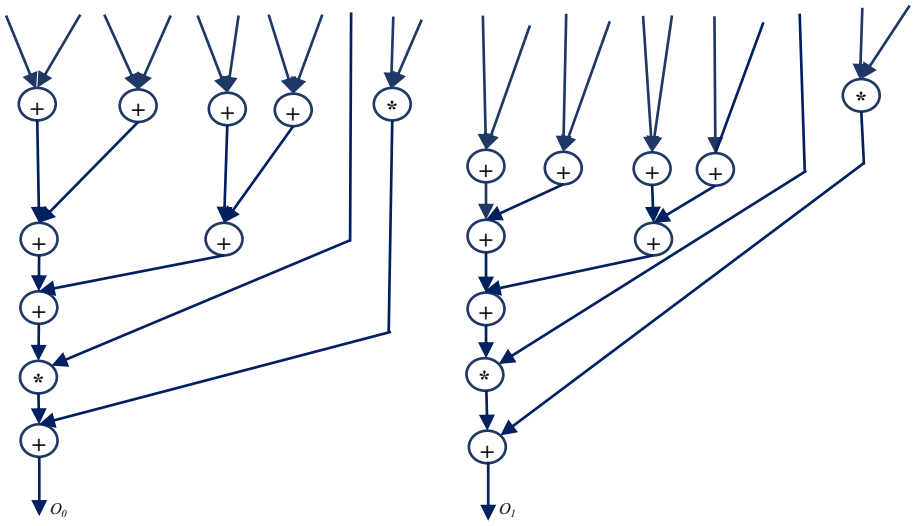


Fig. CDFG of sharpening filter

7) BLUR FILTER (BF)

Ref: A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760.

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.

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

$$h^b = \left(\frac{1}{9}\right) * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}_{3\times3}$$

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

$O_0 = [(I_{00}*(1/9)) + (I_{01}*(1/9)) + (I_{02}*(1/9))] +[(I_{10}*(1/9)) + (I_{11}*(1/9)) + (I_{12}*(1/9))] +[(I_{20}*(1/9)) + (I_{21}*(1/9)) + (I_{22}*(1/9))]$

$O_1 = [(I_{01}*(1/9)) + (I_{02}1/9)) + (I_{03}*(1/9))] +[(I_{11}*(1/9)) + (I_{12}*(1/9)) + (I_{13}*(1/9))] +[(I_{21}*(1/9)) + (I_{22}*(1/9)) + (I_{23}*(1/9))]$

The respective CDFG is shown below [15], [16]:

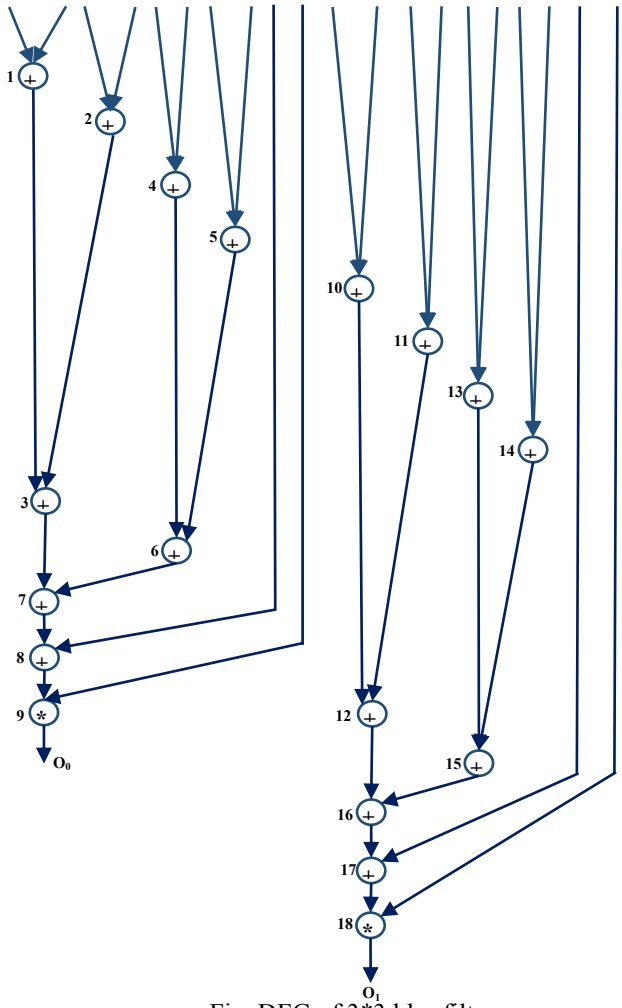


Fig. DFG of 3*3 blur filter

8) LAPLACE EDGE DETECTION FILTER (LED)

Ref: A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760.

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.

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

$$h^{le} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}_{3\times3}$$

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

$O_0 = [(I_{01}*(-1))] +[(I_{10}*(-1)) + (I_{11}*(4)) + (I_{12}*(-1))] +[(I_{21}*(-1))]$

$O_1 = [((I_{02}*(-1))] +[(I_{11}*(-1)) + (I_{12}*(4)) + (I_{13}*(-1))] +[(I_{22}*(-1))]$

The respective CDFG is shown below [17], [18]:

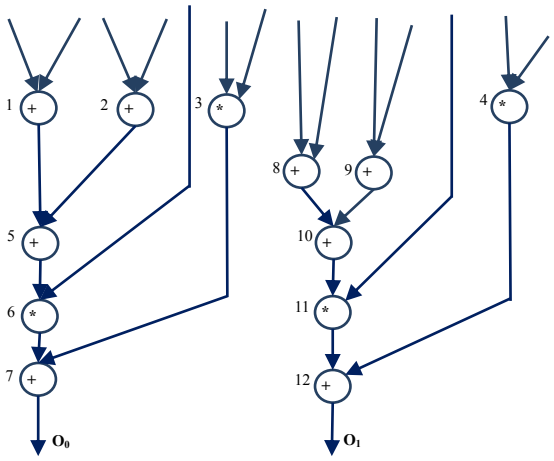


Fig. DFG of 3*3 LED image filter

9) VERTICAL EMBOSSMENT (VE) AND HORIZONTAL EMBOSSMENT (HE) FILTER

Ref: A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760.

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.

The 3×3 kernel for VE and HE filters are shown below:

$$h^{ev} = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}_{3\times3}$$

$$h^{eh} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}_{3\times3}$$

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

$O_0 = [(I_{10}*(-1)) + (I_{12}*(-1))]$

$O_1 = [(I_{11}*(-1)) +[(I_{22}*(-1))]$

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

$O_0 = [(I_{01}*(-1))] +[(I_{21}*(-1))]$

$O_1 = [(I_{11}*(-1)) + [(I_{22}*(-1))]$

The respective CDFG is shown below [19], [20]:

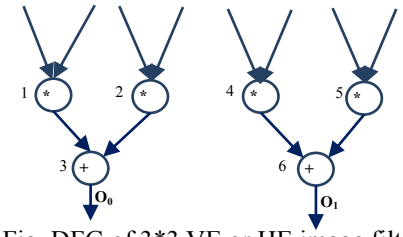
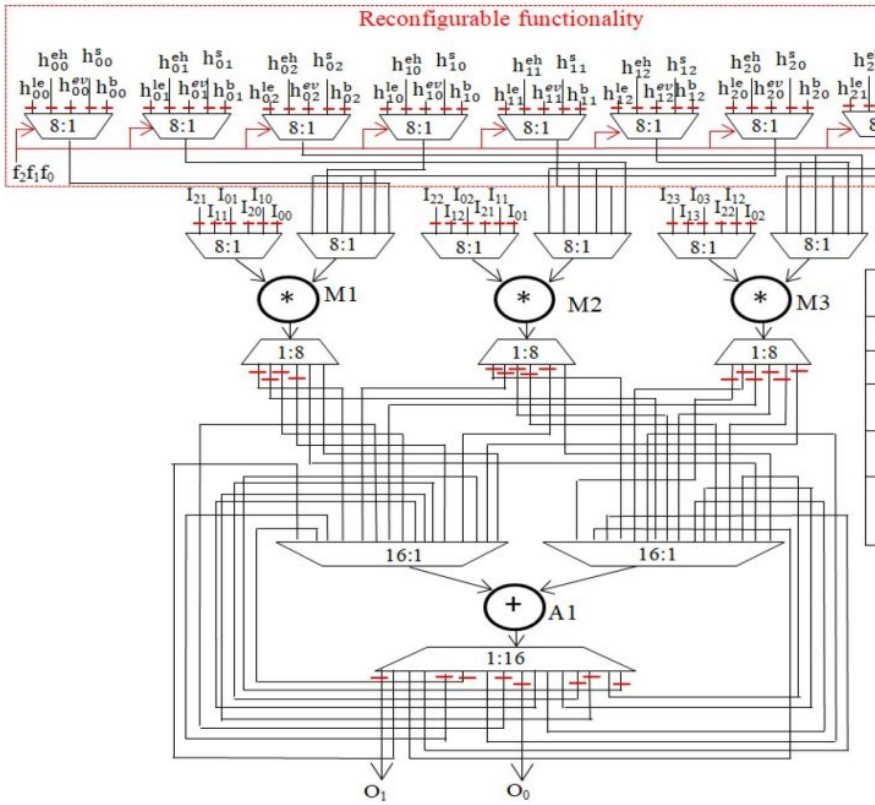


Fig. DFG of 3*3 VE or HE image filter

The HLS generated reconfigurable register transfer level (RTL) datapath corresponding to all above image filters (*i.e.,* SF, BF, LED, VE, and HE) is shown below:



$f_2f_1f_0$	Filter type	Kernel coefficient
000	Blur	h^{b}_{pq}
001	Sharpening	h^{s}_{pq}
010	Vertical embossment	h^{ev}_{pq}
011	Horizontal embossment	h^{eh}_{pq}
100	Laplace edge detection	h^{le}_{pq}

10) GENERALIZED LIKELIHOOD RATIO TEST (GLRT)

Ref: A. Sengupta, A. Anshul, Secure hardware IP of GLRT cascade using color interval graph based embedded fingerprint for ECG detector. *Nature Sci Rep* **14**, 13250 (2024). https://doi.org/10.1038/s41598-024-63533-7.

The GLRT dataflow graph is initially extracted from its corresponding transfer function. The transfer function of GLRT using Mallat's algorithm is as follows:

$Z(a) = s(t)^T H(H^T H)^{-1} H^T s(t)$

where, *s(t)* is the input to the filtering unit and '*H*' is the linear combination matrix of the representative function. Here, $(H^T H)^{-1}$ is a 6×6 matrix and *s(t)* is a one dimensional signal (16-bit for a single precision floating point representation in IEEE 754 standard) captured through the electrode for filtering. The extracted CDFG of GLRT is shown below. WF1, WF2, and WF3 are the outputs of the filtering unit, and C11-C33 is the coefficients of the linear combination matrix (H).

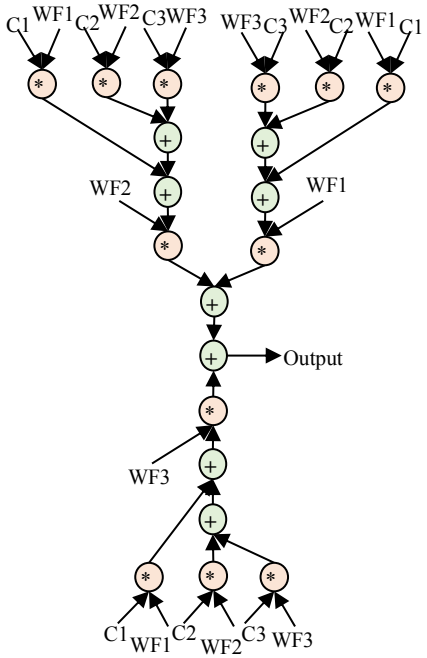
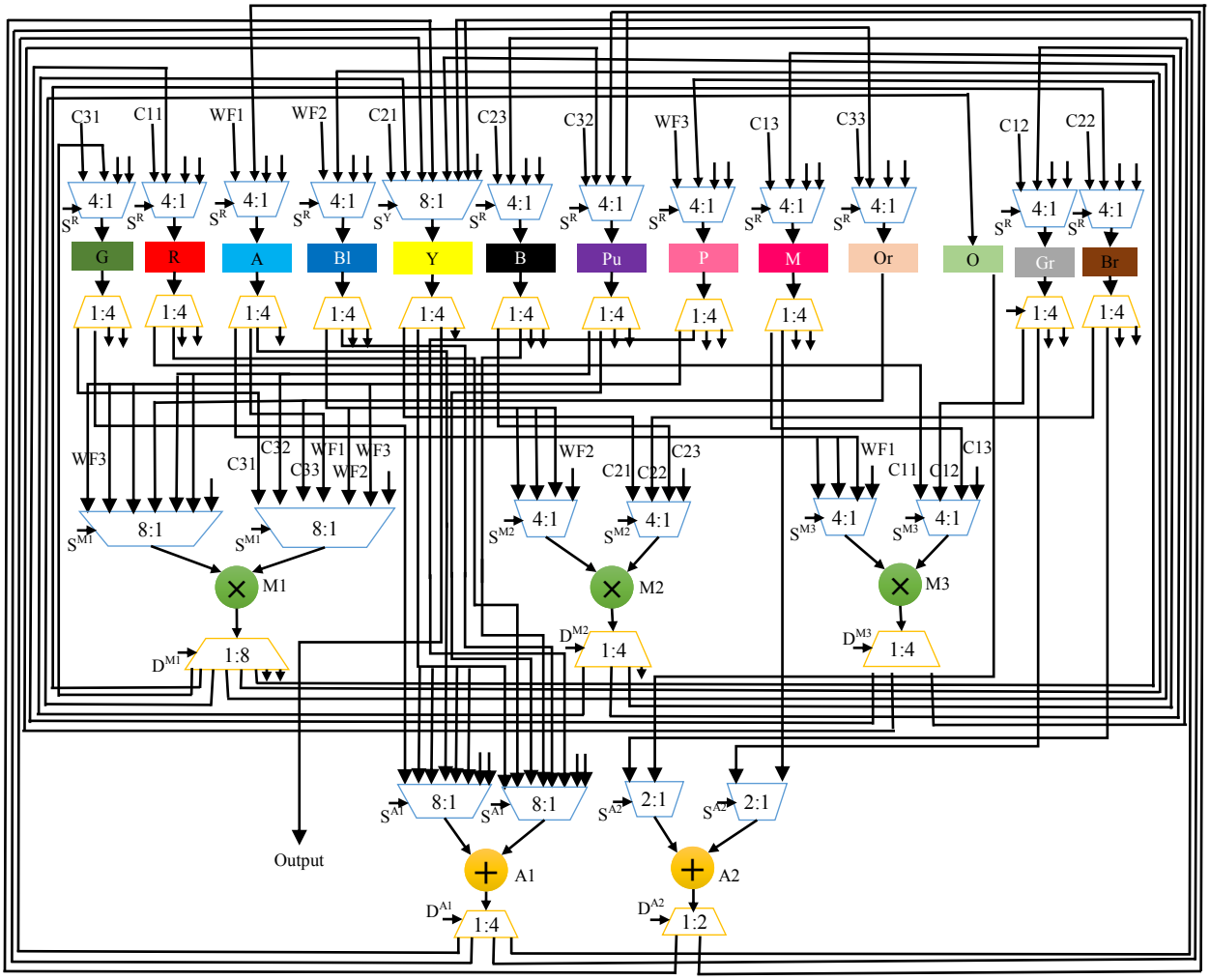


Fig. CDFG/DFG of GLRT IP

The HLS generated register transfer level (RTL) datapath corresponding to GLRT is shown below [21]:



11) CARDIAC PACEMAKER (CP)

Ref: A. Sengupta, R. Chaurasia, Secure implantable cardiac pacemaker for medical consumer electronics. *Nature npj Biomed. Innov.* **2**, 5 (2025). https://doi.org/10.1038/s44385-025-00008-y.

A cardiac pacemaker, comprises of filter banks as a wavelet decomposer and QRS complex, which are the computationally intensive counterparts in order to perform accurate pulse detection and enabling normal heart rhythm. Post receiving the pulse signal from pacemaker lead, it propagates to filter bank via ADC. Filter bank decompose the input pulse signal into sub bands with two outputs as W_i and W_{i+1} corresponding to monophasic and biphasic morphologies. The filter bank comprises of three branches (F1=2), (F2=3) and (F3=4), that are responsible for scaling and filtering the pulse signal. This is because to cover the frequency spectrum of an R wave, three scales are sufficient. Further, filter bank output is defined as:

$$y(n) = X^T(n)H$$

Where, X(n) is the input to the filter bank and H is defined as the matrix formed by linear combinations of representative functions which is computed using Mallat's algorithm. Finally, the decision signal Z(n) is computed by a generalized likelihood ratio test (GLRT) as follows:

$$Z(n) = X^T(n)H(H^TH)^{-1}H^TX(n)$$

Where,

$$(H^TH)^{-1} = \begin{bmatrix} 4.25 & -2.81 & 0.71 & 0 & 0 & 0 \\ -2.81 & 4.47 & -1.75 & 0 & 0 & 0 \\ 0.71 & -1.75 & 1.49 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4.84 & -2.31 & 0.6 \\ 0 & 0 & 0 & -2.31 & 4.29 & -1.49 \\ 0 & 0 & 0 & 0.6 & -1.49 & 1.77 \end{bmatrix}$$

The component $(H^TH)^{-1}$ in eqn. (2) is symmetric and sparse (comprising half of the elements as zero. Therefore, it eliminates the need for their implementation. Subsequently, the threshold level determines the existence of an R wave and signals to the pulse generator accordingly. Thus, GLRT with threshold function estimates the heart rate with the outputs of decomposed filter bank.

The CDFG corresponding to filter bank and QRS complex are generated by transforming their mathematical description/transfer function. The transfer function of employed low pass and high pass filter in the cardiac pacemaker filter bank is given as follows:

Transfer function of low pass filter:

$$H(Z) = 1 + 3Z^{-1} + 3Z^{-2} + Z^{-3}$$

$$\frac{Y(Z)}{X(Z)} = 1 + 3Z^{-1} + 3Z^{-2} + Z^{-3}$$

$$Y(Z) = X(Z) + X(Z).3Z^{-1} + X(Z).3Z^{-2} + X(Z).Z^{-3}$$

Now, the above signal corresponding to low pass filter can be represented in time domain as follows (by applying the Z transformation):

$$Y(n) = x(n) + 3x(n-1) + 3x(n-2) + x(n-3)$$

Further, the transfer function of high pass filter:

$$H(Z) = 1 + Z^{-1}$$

$$\frac{Y(Z)}{X(Z)} = 1 + Z^{-1}$$

$$Y(Z) = X(Z) + X(Z).Z^{-1}$$

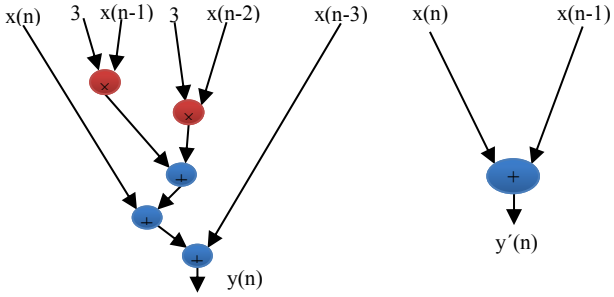
Now, the above signal corresponding to high pass filter can be represented in time domain as follows (by applying the Z transformation):

$$y'(n) = x(n) + x(n-1)$$

Subsequently, the CDFG corresponding to low pass and high pass filters is derived, where, x(n) indicates the current input and x(n-1), x(n-2) and x(n-3) represents input values corresponding to previous stages and the integer values represent the filter coefficient.

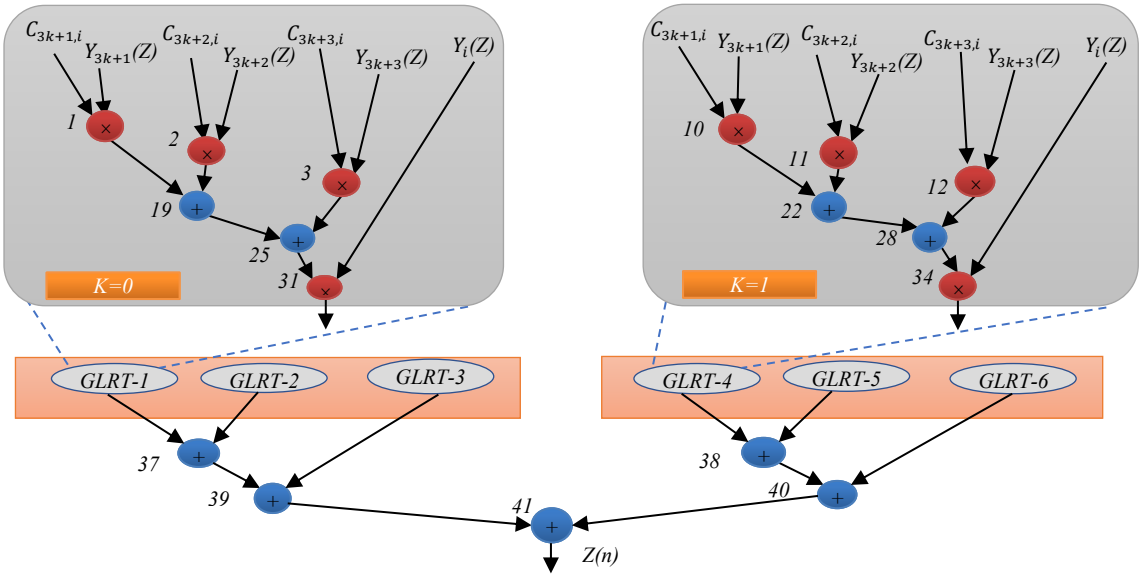
Further, the transfer function for QRS complex is obtained, where the multiplication of y(n) with the first column of $(H^TH)^{-1}$ and the first element of $H^TX(n)$ is executed, where $C_{3k+j,i}$ are elements of $(H^TH)^{-1}$ and $y_{3k+j}(n)$ the

output of the filter bank, with K=0,1 and j= 1,2,3. The CDFG corresponding to QRS complex is shown below, where $C_{3k+1,i}$, $C_{3k+2,i}$ and $C_{3k+3,i}$ represents the coefficient of the matrix corresponding to first column and $Y_{3k+1}(Z)$, $Y_{3k+2}(Z)$, $Y_{3k+3}(Z)$ and $Y_i(Z)$ represents the outputs of the filter bank. Similarly, for the other GLRT block, their corresponding CDFG is constructed. Finally, by performing the sum of these decomposed bands (GLRT outputs), pulse signal is constructed. It is then analyzed against threshold function to determine whether the incoming pulse is adjudged as a cardiac signal or as noise. The CDFG corresponding to QRS complex requires a total of 41 operations including 24 multiplications and 17 additions [22].



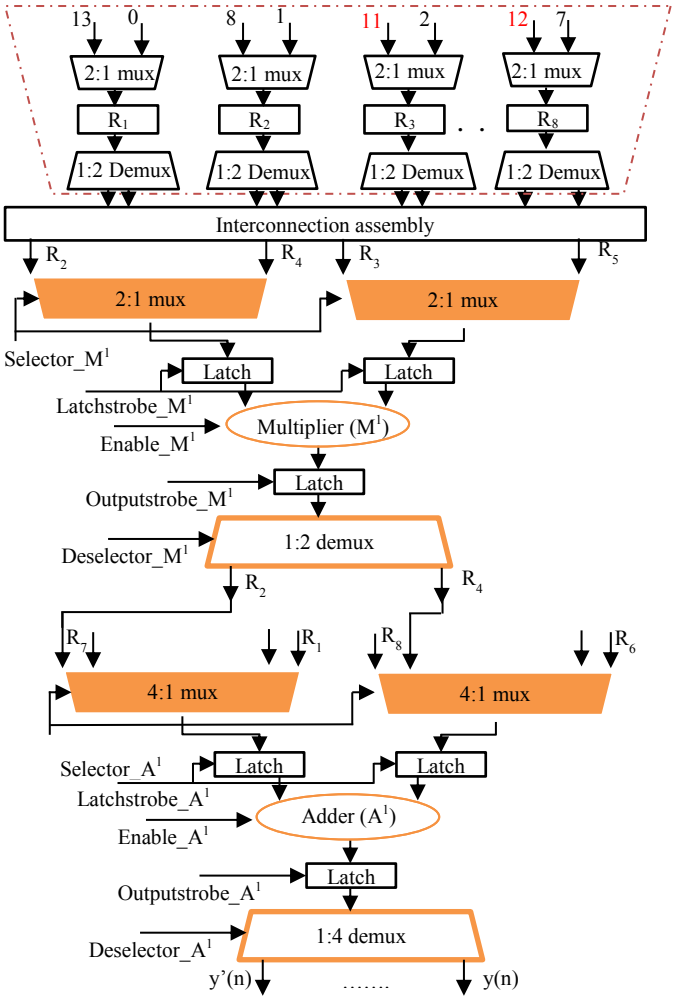
(a) CDFG corresponding to transfer function of low pass filter (LPF)

(b) CDFG corresponding to transfer function of high pass filter (HPF)

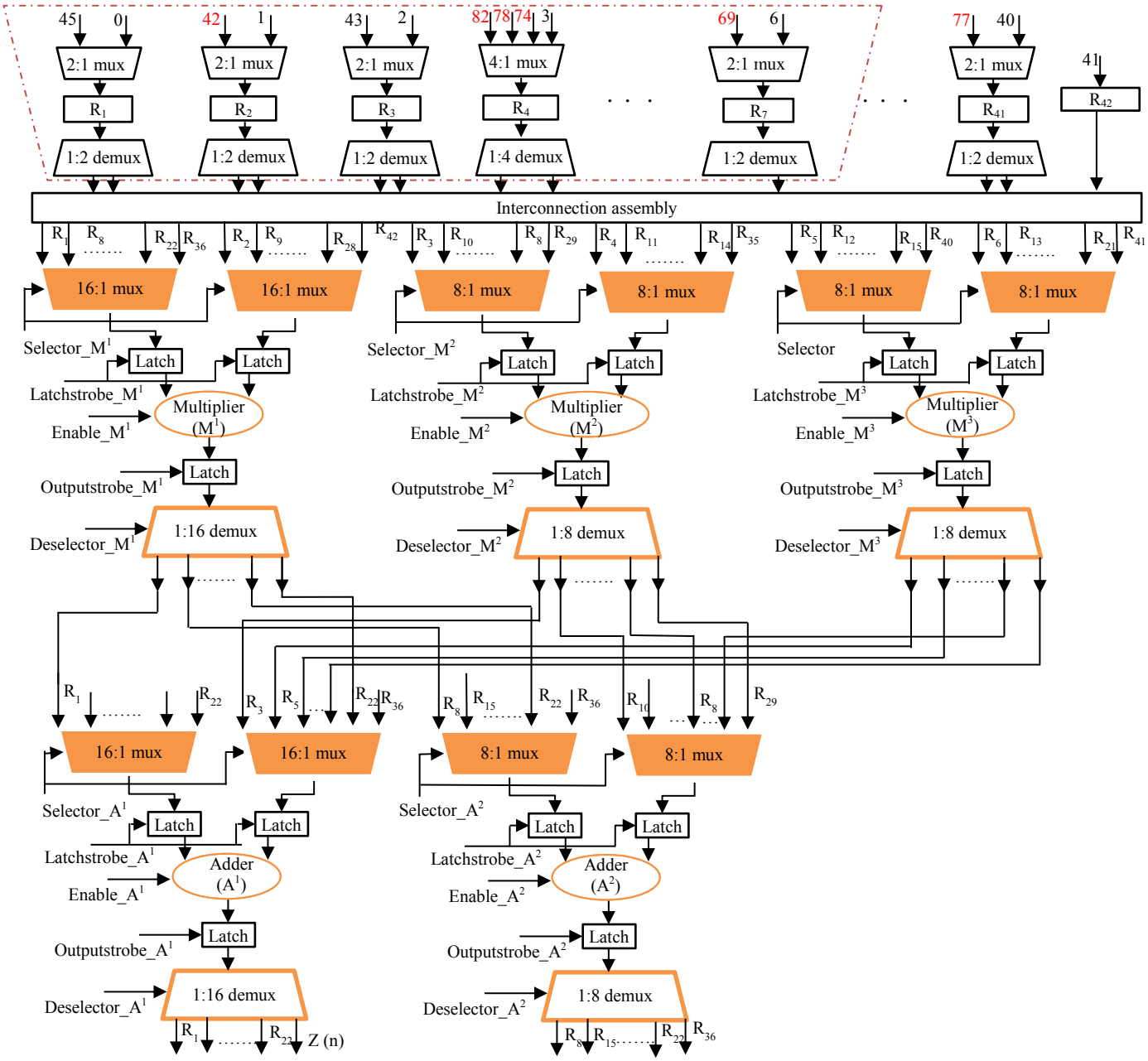


(c) CDFG of QRS complex

The HLS generated register transfer level (RTL) datapath corresponding to filter bank is shown below [22]:



The HLS generated register transfer level (RTL) datapath corresponding to QRS complex is shown below [22]:



12) LINEAR REGRESSION (LR)

Ref: A. Sengupta, R. Chaurasia, M. Rathor, HLS-based swarm intelligence driven optimized hardware IP core for linear regression-based machine learning. *IET Journal of Eng.* 2023, e12299 (2023). https://doi.org/10.1049/tje2.12299.

The cost function (f) of linear regression, in terms of mean square error (MSE), is given as follows:

$$f = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - y\right)^2$$

Where, N indicates total number of data points, y_i indicates actual value of i^{th} data point and y indicates the predicted value obtained using following function of linear equation, which is the basis of linear regression based ML.

$$y = mx + b$$

Where, x and y denote input (independent) and output (dependent) variable respectively, m denotes slop of the line and b denotes intercept.

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (mx_i + b)\right)^2$$

To determine the impact of change of m on y, partial derivative of cost function with respect to 'm' is performed.

$$\frac{\partial f}{\partial m} = \frac{\partial\left(\frac{1}{N}\sum_{i=1}^{N}\left(y_i - (mx_i + b)\right)^2\right)}{\partial m}$$

$$\frac{\partial f}{\partial m} = \frac{\partial\left(\frac{1}{N}\sum_{i=1}^{N}\left(y_i^2 + m^2x_i^2 + b^2 + 2mx_ib - 2y_imx_i - 2y_ib\right)\right)}{\partial m}$$

$$\frac{\partial f}{\partial m} = \frac{2}{N}\sum_{i=1}^{N}\left((-x_i)(y_i - (mx_i + b)\right)$$

Similarly, to determine the impact of change of b on y, partial derivative of cost function with respect to 'b' is performed.

$$\frac{\partial f}{\partial b} = \frac{\partial\left(\frac{1}{N}\sum_{i=1}^{N}\left(y_i - (mx_i + b)\right)^2\right)}{\partial b}$$

$$\frac{\partial f}{\partial b} = \frac{2}{N}\sum_{i=1}^{N}\left((-1)(y_i - (mx_i + b)\right)$$

The new values of m and b are obtained as follows:

$$m^{new} = m^{old} - \phi_1\frac{\partial f}{\partial m}$$

$$b^{new} = b^{old} - \phi_2\frac{\partial f}{\partial b}$$

Where, ϕ_1 and ϕ_2 are learning factors for m and b respectively. The equation of m^{new} and b^{new} are further written as follows:

$$m^{new} = m^{old} + \phi_1\frac{2}{N}\sum_{i=1}^{N}\left(x_i(y_i - (m^{old}x_i + b^{old})\right)$$

$$b^{new} = b^{old} + \phi_2\frac{2}{N}\sum_{i=1}^{N}\left((y_i - (m^{old}x_i + b^{old})\right)$$

Hence, new cost is given as follows:

$$f^{new}(m,b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (m^{new}x_i + b^{new})\right)^2$$

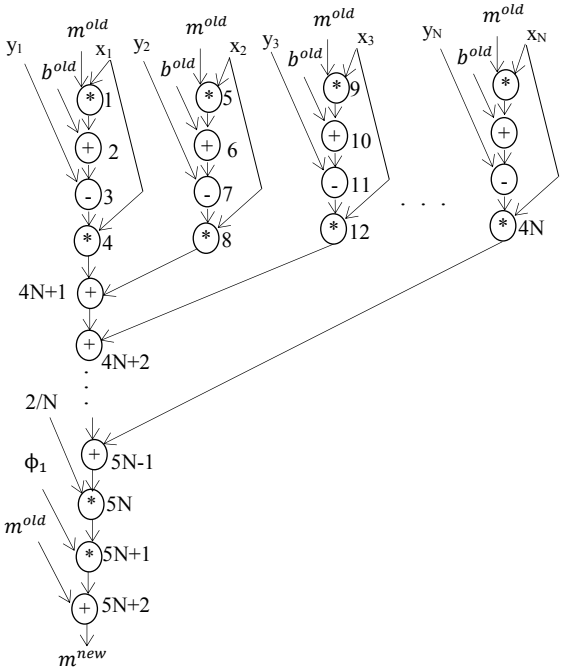


Fig. DFG of computing m^{new} for linear regression based ML co-processor

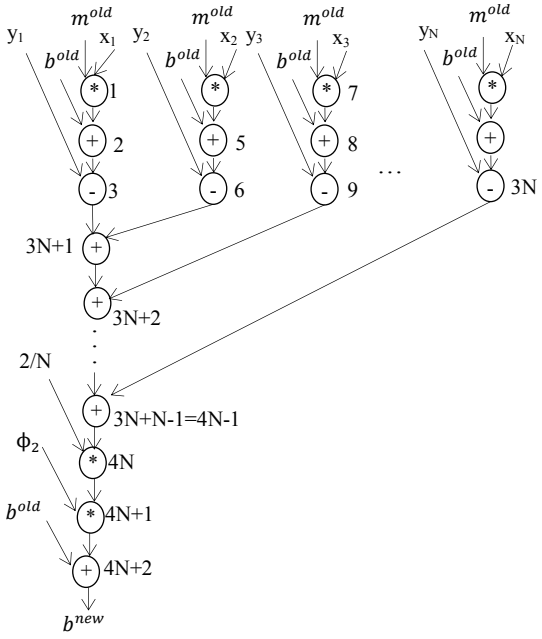


Fig. DFG of computing b^{new} for linear regression based ML co-processor

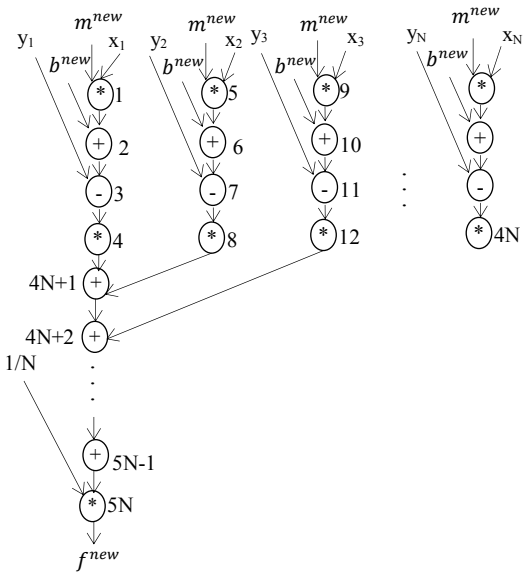


Fig. DFG of computing f^{new} for linear regression based ML co-processor

The HLS generated register transfer level (RTL) datapath corresponding to linear regression based ML co-processor is shown below [23]:

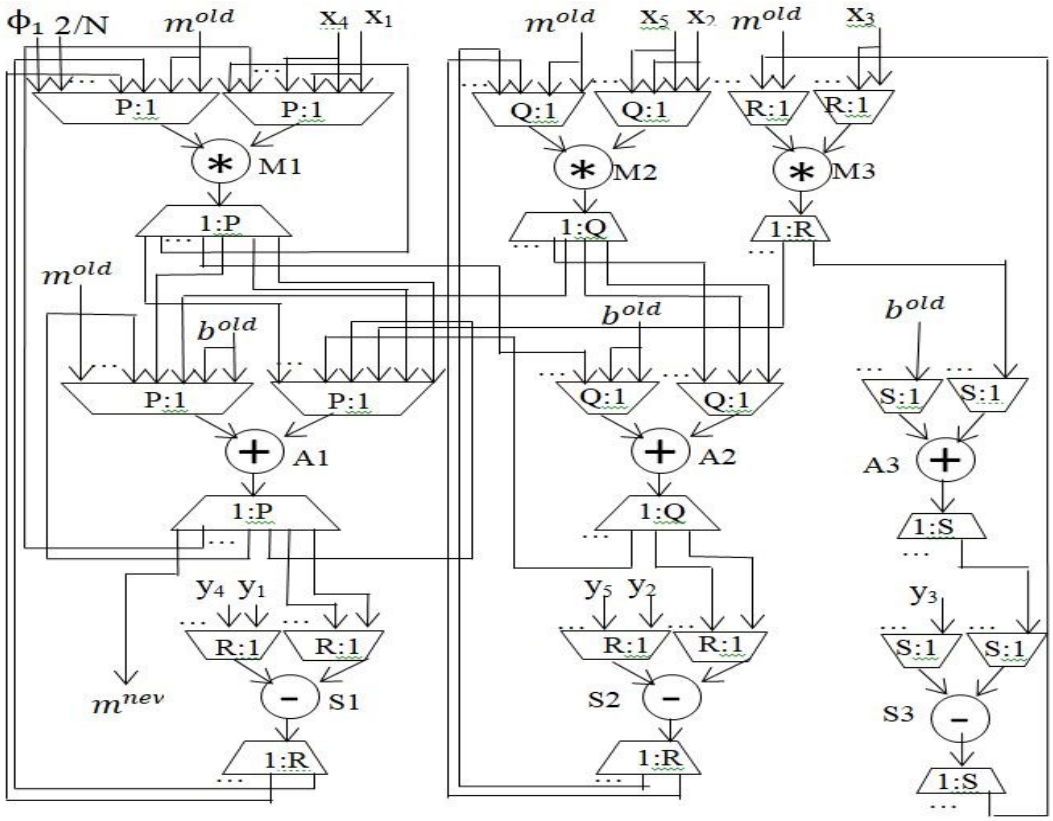


Fig. RTL design with THT for m^{new} for linear regression based ML co-processor

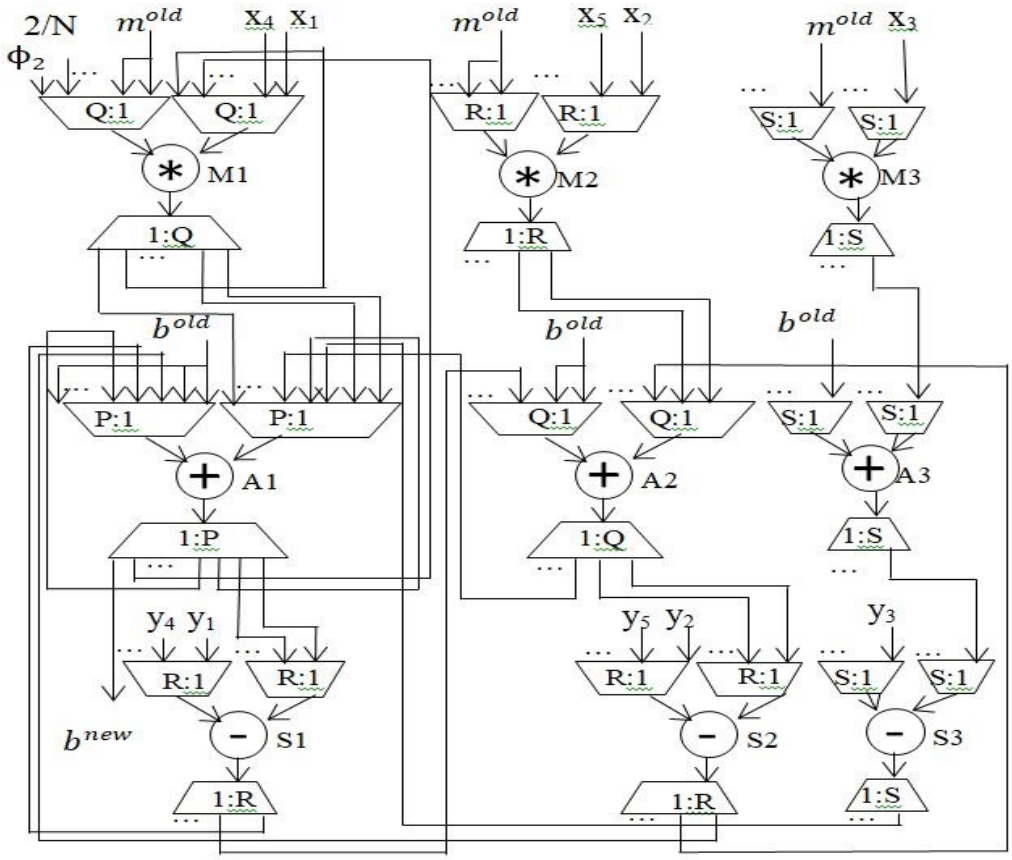


Fig. RTL design with THT for b^{new} for linear regression based ML co-processor

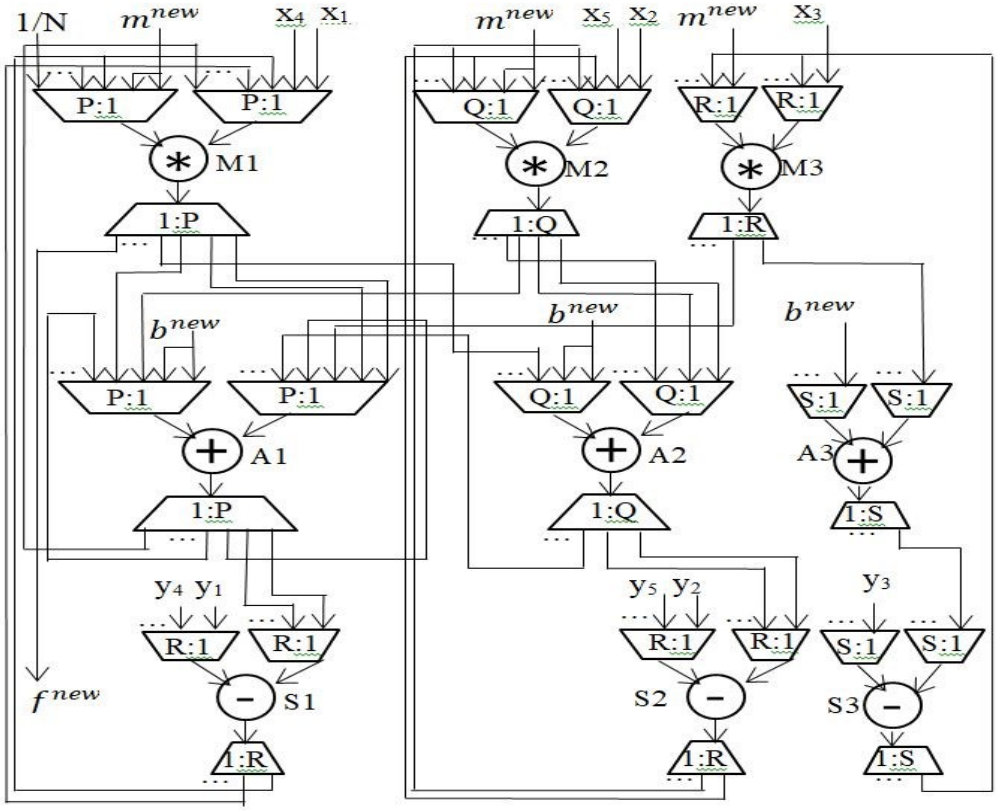


Fig. RTL design with THT for f^{new} for linear regression based ML co-processor

13) JOINT PHOTOGRAPHIC EXPERT GROUP COMPRESSION DECOMPRESSION (JPEG-CODEC)

Ref: A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "Low-Cost Obfuscated JPEG CODEC IP Core for Secure CE Hardware," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 3, pp. 365-374, Aug. 2018, doi: 10.1109/TCE.2018.2852265.

The JPEG-CODEC application is responsible for performing JPEG compression on the images and is commonly used in several multimedia and consumer electronic devices. The input of the JPEG-CODEC application is a pre-processed image. Mathematically a grayscale image is represented using a pixel intensity matrix ranging from 0 to 255, where 0 denotes pure black, and 255 denotes pure white (this scale is for 8-bit depth grayscale images). Further, *'F'* (given below) denotes a generic 2D- discrete cosine transform (DCT) matrix used to process input grayscale images. As *'F'* is an 8×8 matrix and can process a maximum of 8×8 pixel values at one time. Further, f_n indicates the elements of the *F* matrix. Therefore, the input image data is divided and grouped into 8×8 matrix blocks. *'Z'* (given below)represents an 8×8 matrix block of the input image in a generic form. The standard quantization matrix *(Q)* is also an important input component. Next, each pixel intensity value from the input 8×8 matrix block is subtracted with 128 as discrete cosine transform coefficient matrix can only handle pixel values from range -128 to 127. Moreover, the JPEG algorithm comprises of steps such as zigzag scanning and run-length encoding to generate a compressed image from an input image. Similarly, the process for decompression is the reverse of the compression process algorithm. The first compressed image pixel data is shown below.

$I_{11} = (f_4 * W_{11} + f_4 * W_{12} + f_4 * * W_{13} + f_4 * W_{14} + f_4 * W_{15} + f_4 * W_{16} + f_4 * W_{17} + f_4 * W_{18})$

where, W_{11}, W_{12}, W_{13}, ……W_{18} is evaluated as follows:

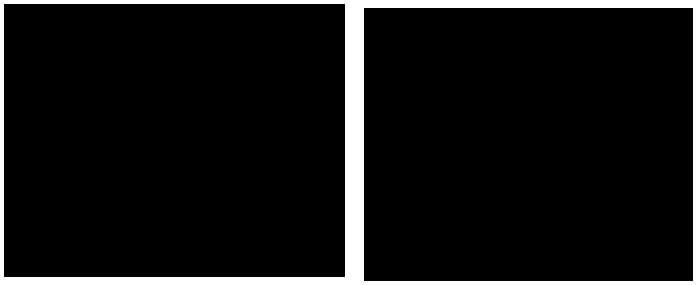
$W_{11} = (f_4 * Z_{11} + f_4 * Z_{21} + f_4 * * Z_{31} + f_4 * Z_{41} + f_4 * Z_{51} + f_4 * Z_{61} + f_4 * Z_{71} + f_4 * Z_{81})$

$W_{12} = (f_4 * Z_{12} + f_4 * Z_{22} + f_4 * * Z_{32} + f_4 * Z_{42} + f_4 * Z_{52} + f_4 * Z_{62} + f_4 * Z_{72} + f_4 * Z_{82})$

Similarly,

$W_{18} = (f_4 * Z_{18} + f_4 * Z_{28} + f_4 * * Z_{38} + f_4 * Z_{48} + f_4 * Z_{58} + f_4 * Z_{68} + f_4 * Z_{78} + f_4 * Z_{88})$

Next, the remaining pixel value output is computed in a similar fashion. A control data flow graph corresponding to the JPEG image compression IP core is shown below. There are eight micro IP components (*namely from IP1 to IP8*). Moreover, a zoomed image of one of the micro IP cores is also shown below. It also highlights the quantization operation as each final generated output is multiplied with *'Q'*. JPEG algorithm uses different quantization matrixes to generate better results.



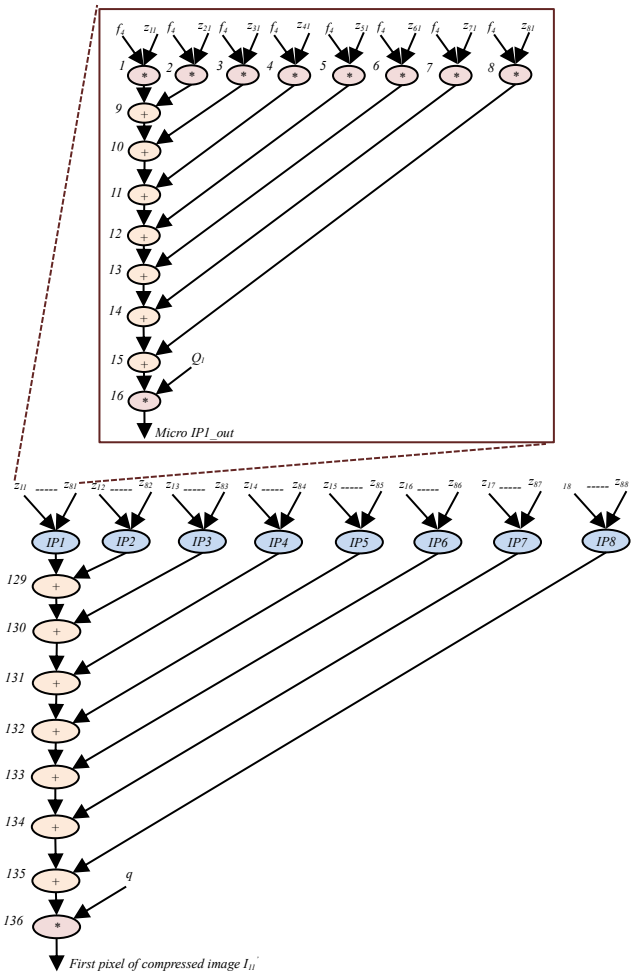
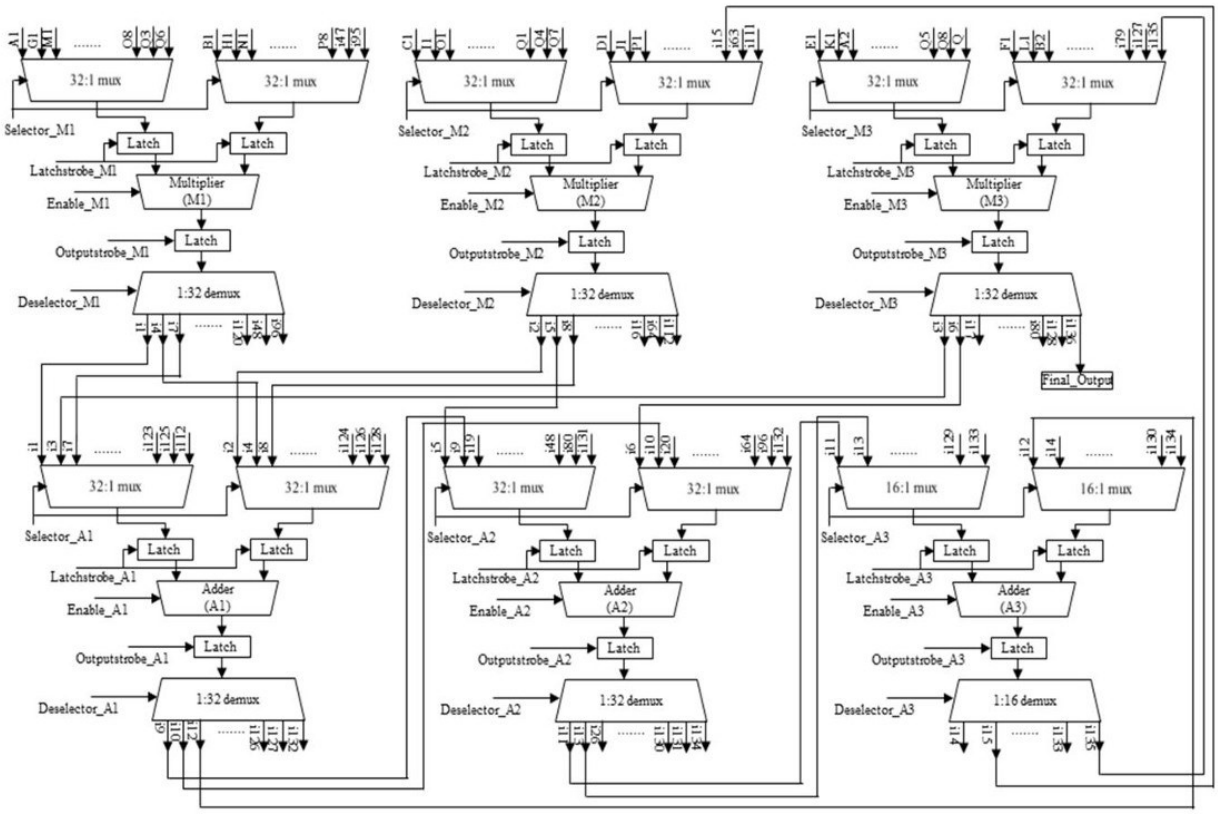


Fig. DFG of JPEG-CODEC for determining first pixel of the compressed image I_{11}'

The HLS generated register transfer level (RTL) datapath corresponding to JPEG-CODEC is shown below [24]:



References

[1] 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.

[2] A. Sengupta, R. Chaurasia and A. Anshul, "Robust Security of Hardware Accelerators Using Protein Molecular Biometric Signature and Facial Biometric Encryption Key," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 6, pp. 826-839, June 2023, doi: 10.1109/TVLSI.2023.3265559

[3] 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

[4] 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.

[5] A. Sengupta, D. Kachave and D. Roy, "Low Cost Functional Obfuscation of Reusable IP Cores Used in CE Hardware Through Robust Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 604-616, April 2019, doi: 10.1109/TCAD.2018.2818720

[6] A. Sengupta and M. Rathor, "Enhanced Security of DSP Circuits Using Multi-Key Based Structural Obfuscation and Physical-Level Watermarking for Consumer Electronics Systems," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 163-172, May 2020, doi: 10.1109/TCE.2020.2972808

[7] 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

[8] M. Rathor, A. Anshul and A. Sengupta, "Securing Reusable IP Cores Using Voice Biometric Based Watermark," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 2735-2749, July-Aug. 2024, doi: 10.1109/TDSC.2023.3315780

[9] 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

[10] 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

[11] A. Sengupta and D. Roy, "Antipiracy-Aware IP Chipset Design for CE Devices: A Robust Watermarking Approach," *IEEE Consumer Electronics Magazine*, vol. 6, no. 2, pp. 118-124, April 2017, doi: 10.1109/MCE.2016.2640622

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

[13] A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760

[14] 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

[15] A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760

[16] 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

[17] A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760
[18] 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
[19] A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters—Application Specific and Functionally Reconfigurable Processors," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 4, pp. 386-395, Nov. 2020, doi: 10.1109/TCE.2020.3027760
[20] 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
[21] A. Sengupta, A. Anshul, Secure hardware IP of GLRT cascade using color interval graph based embedded fingerprint for ECG detector. *Nature Sci Rep* **14**, 13250 (2024). https://doi.org/10.1038/s41598-024-63533-7
[22] A. Sengupta, R. Chaurasia, Secure implantable cardiac pacemaker for medical consumer electronics. *Nature npj Biomed. Innov.* **2**, 5 (2025). https://doi.org/10.1038/s44385-025-00008-y
[23] A. Sengupta, R. Chaurasia, M. Rathor, HLS-based swarm intelligence driven optimized hardware IP core for linear regression-based machine learning. *IET Journal of Eng.* 2023, e12299 (2023). https://doi.org/10.1049/tje2.12299
[24] A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "Low-Cost Obfuscated JPEG CODEC IP Core for Secure CE Hardware," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 3, pp. 365-374, Aug. 2018, doi: 10.1109/TCE.2018.2852265
[25] A. Sengupta, S. Bhadauria and S. P. Mohanty, "TL-HLS: Methodology for Low Cost Hardware Trojan Security Aware Scheduling With Optimal Loop Unrolling Factor During High Level Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 4, pp. 655-668, April 2017, doi: 10.1109/TCAD.2016.2597232
[26] A. Sengupta and S. Bhadauria, "Exploring Low Cost Optimal Watermark for Reusable IP Cores During High Level Synthesis," *IEEE Access*, vol. 4, pp. 2198-2215, 2016, doi: 10.1109/ACCESS.2016.2552058