EE3510

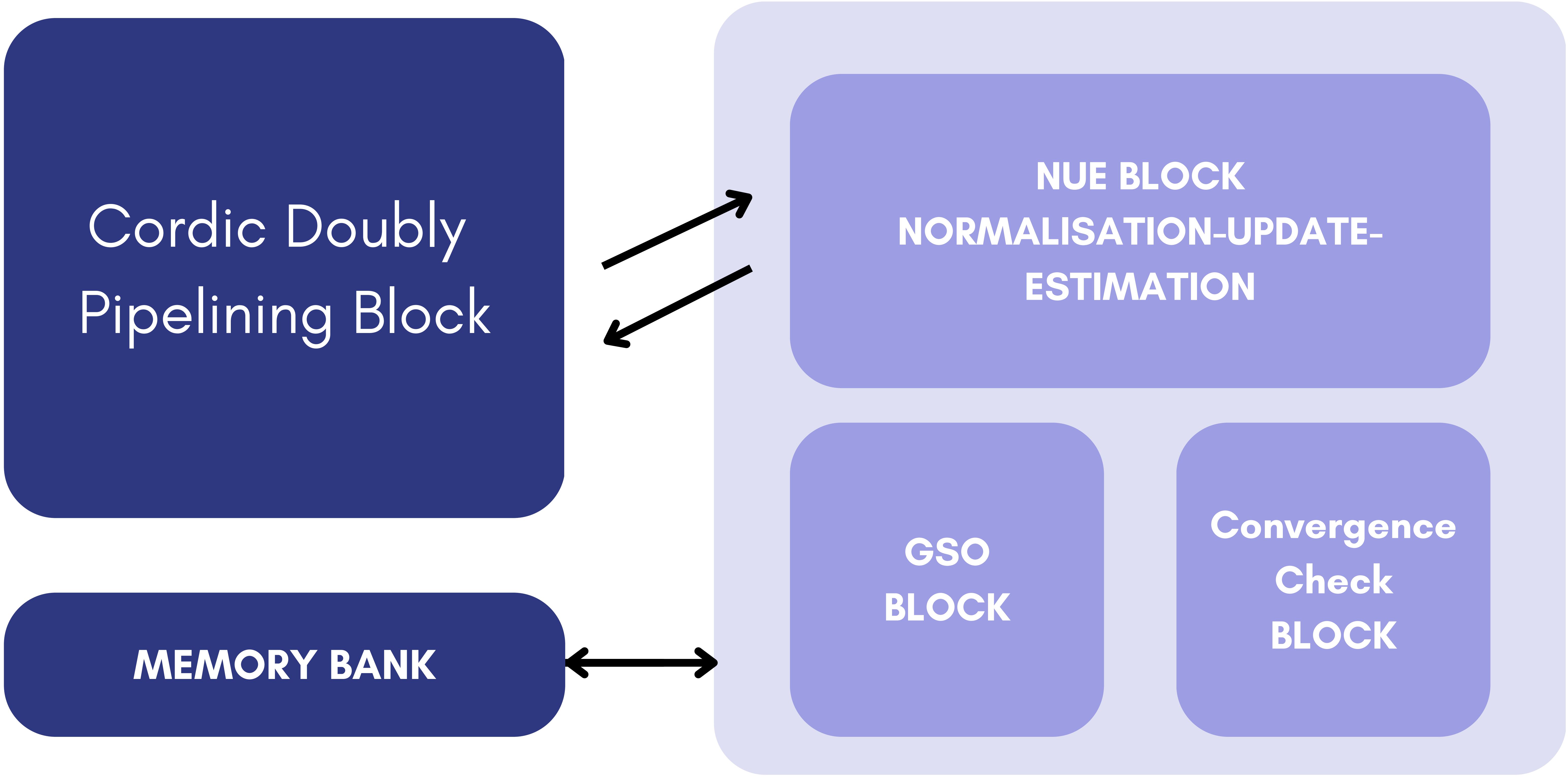
RTL Design and Verification

ICA (Independent Component Analysis)

Devansh Srivatsava - EE22BTECH11207
Gargi Behera - EE22BTECH11208
Rayi Giri Varshini - EE22BTECH11215
Talasu Sowmith - EE22BTECH11218

Independent Component Analysis

A computational technique used to separate a **multivariate signal** into additive, **independent components**. ICA, widely used for blind source separation extracting **individual signals from mixtures** by assuming different physical processes generate unrelated signals. It ranges from audio and image processing to biomedical signal analysis.



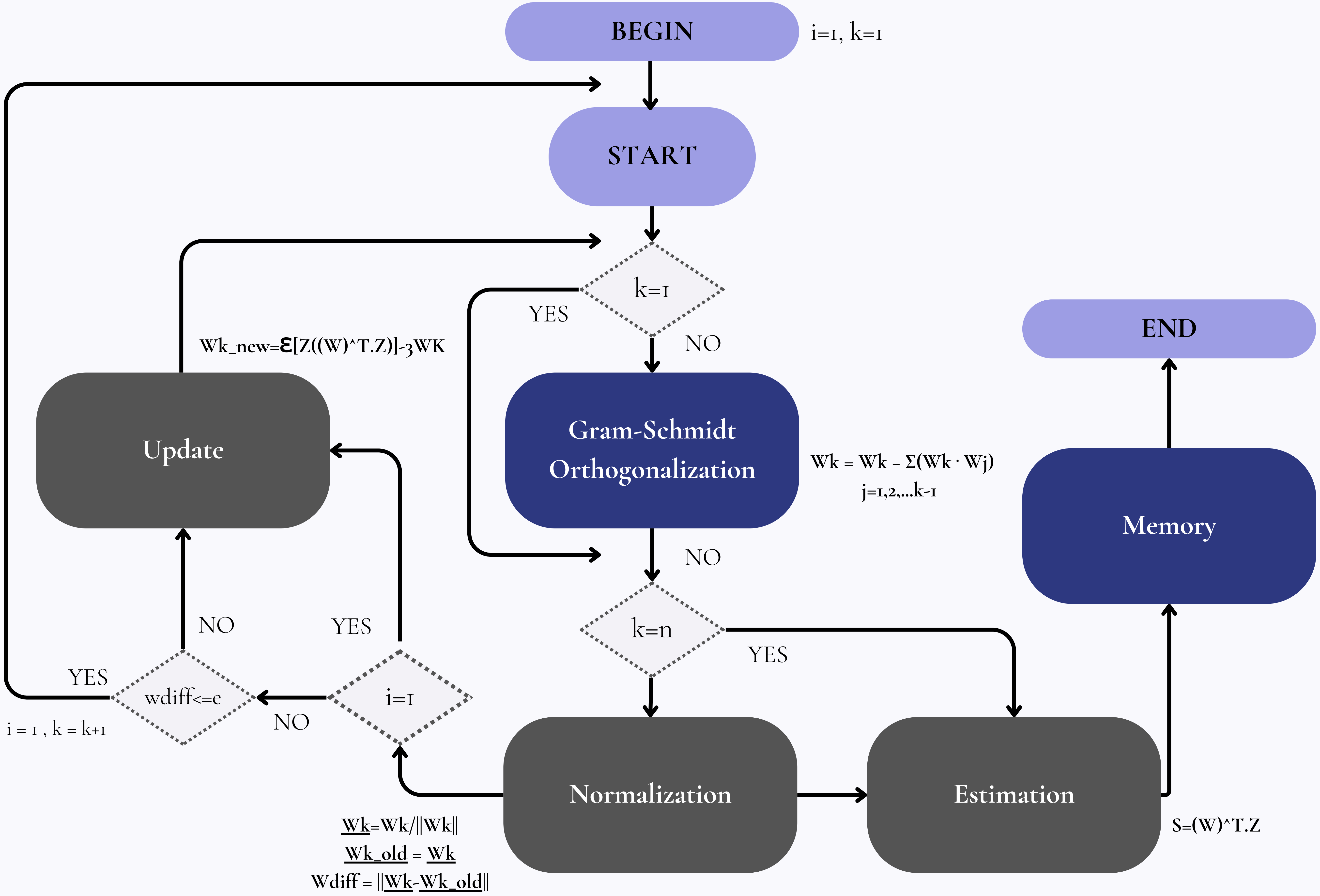
Cordic Doubly Pipelining Block

MEMORY BANK

NUE BLOCK NORMALISATION-UPDATE-ESTIMATION

GSO BLOCK

Convergence Check BLOCK



BEGIN

i=1, k=1

START

k=1

YES

NO

Gram-Schmidt Orthogonalization

Wk = Wk - Σ(Wk · Wj)
j=1,2,...k-1

NO

k=n

YES

Normalization

Wk=Wk/||Wk||
Wk_old = Wk
Wdiff = ||Wk-Wk_old||

Estimation

S=(W)^T.Z

Memory

END

i=1

YES

NO

wdiff<=e

YES

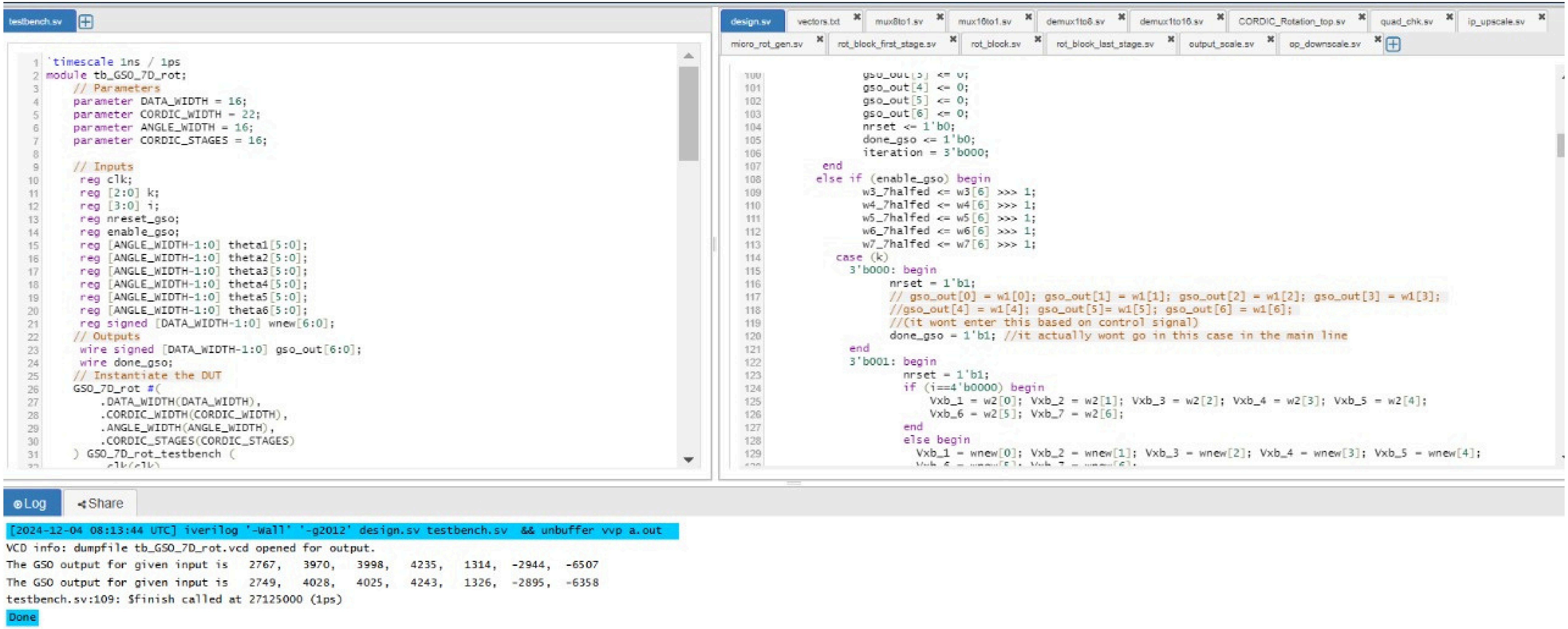
NO

i = 1 , k = k+1

Update

Wk_new=Ɛ[Z((W)^T.Z)]-3WK

Code:



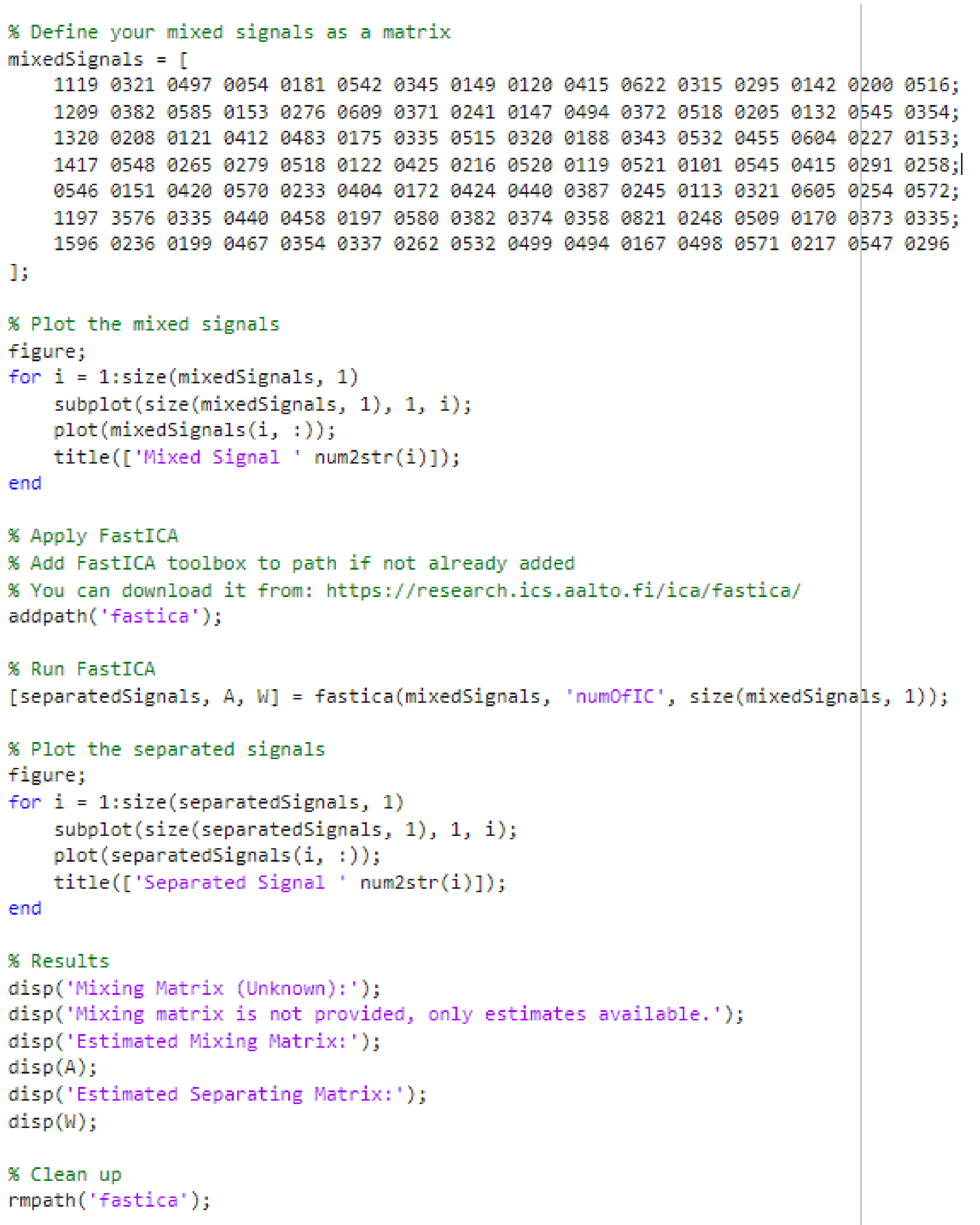
```
`timescale 1ns / 1ps
module tb_GSO_7D_rot;
    // Parameters
    parameter DATA_WIDTH = 16;
    parameter CORDIC_WIDTH = 22;
    parameter ANGLE_WIDTH = 16;
    parameter CORDIC_STAGES = 16;

    // Inputs
     reg clk;
     reg [2:0] k;
     reg [3:0] i;
     reg nreset_gso;
     reg enable_gso;
     reg [ANGLE_WIDTH-1:0] theta1[5:0];
     reg [ANGLE_WIDTH-1:0] theta2[5:0];
     reg [ANGLE_WIDTH-1:0] theta3[5:0];
     reg [ANGLE_WIDTH-1:0] theta4[5:0];
     reg [ANGLE_WIDTH-1:0] theta5[5:0];
     reg [ANGLE_WIDTH-1:0] theta6[5:0];
     reg signed [DATA_WIDTH-1:0] wnew[6:0];
    // Outputs
     wire signed [DATA_WIDTH-1:0] gso_out[6:0];
     wire done_gso;
    // Instantiate the DUT
    GSO_7D_rot #(
        .DATA_WIDTH(DATA_WIDTH),
        .CORDIC_WIDTH(CORDIC_WIDTH),
        .ANGLE_WIDTH(ANGLE_WIDTH),
        .CORDIC_STAGES(CORDIC_STAGES)
    ) GSO_7D_rot_testbench (
        .clk(clk)
```

```
              gso_out[3] <= 0;
              gso_out[4] <= 0;
              gso_out[5] <= 0;
              gso_out[6] <= 0;
              nrset <= 1'b0;
              done_gso <= 1'b0;
              iteration = 3'b000;
        end
       else if (enable_gso) begin
              w3_7halfed <= w3[6] >>> 1;
              w4_7halfed <= w4[6] >>> 1;
              w5_7halfed <= w5[6] >>> 1;
              w6_7halfed <= w6[6] >>> 1;
              w7_7halfed <= w7[6] >>> 1;
          case (k)
            3'b000: begin
                  nrset = 1'b1;
                  // gso_out[0] = w1[0]; gso_out[1] = w1[1]; gso_out[2] = w1[2]; gso_out[3] = w1[3];
                  //gso_out[4] = w1[4]; gso_out[5]= w1[5]; gso_out[6] = w1[6];
                  //(it wont enter this based on control signal)
                  done_gso = 1'b1; //it actually wont go in this case in the main line
            end
            3'b001: begin
                    nrset = 1'b1;
                    if (i==4'b0000) begin
                        Vxb_1 = w2[0]; Vxb_2 = w2[1]; Vxb_3 = w2[2]; Vxb_4 = w2[3]; Vxb_5 = w2[4];
                        Vxb_6 = w2[5]; Vxb_7 = w2[6];
                    end
                    else begin
                      Vxb_1 = wnew[0]; Vxb_2 = wnew[1]; Vxb_3 = wnew[2]; Vxb_4 = wnew[3]; Vxb_5 = wnew[4];
                      Vxb_6 = wnew[5]; Vxb_7 = wnew[6];
```

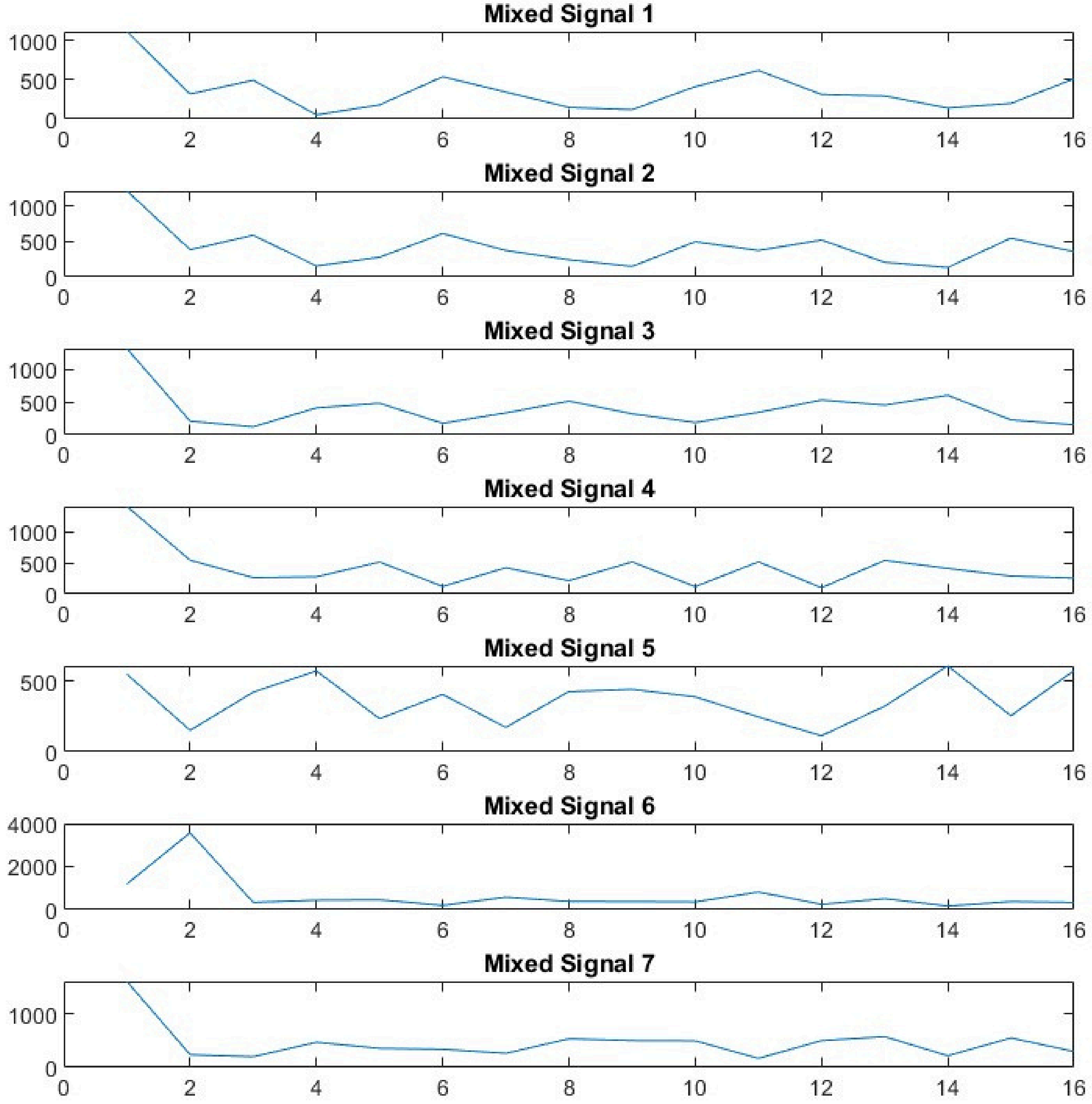
```
[2024-12-04 08:13:44 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile tb_GSO_7D_rot.vcd opened for output.
The GSO output for given input is    2767,   3970,   3998,   4235,   1314,  -2944,  -6507
The GSO output for given input is    2749,   4028,   4025,   4243,   1326,  -2895,  -6358
testbench.sv:109: $finish called at 27125000 (1ps)
Done
```

Matlab FastICA

(With the test bench Zin Input)



```
% Define your mixed signals as a matrix
mixedSignals = [
    1119 0321 0497 0054 0181 0542 0345 0149 0120 0415 0622 0315 0295 0142 0200 0516;
    1209 0382 0585 0153 0276 0609 0371 0241 0147 0494 0372 0518 0205 0132 0545 0354;
    1320 0208 0121 0412 0483 0175 0335 0515 0320 0188 0343 0532 0455 0604 0227 0153;
    1417 0548 0265 0279 0518 0122 0425 0216 0520 0119 0521 0101 0545 0415 0291 0258;
    0546 0151 0420 0570 0233 0404 0172 0424 0440 0387 0245 0113 0321 0605 0254 0572;
    1197 3576 0335 0440 0458 0197 0580 0382 0374 0358 0821 0248 0509 0170 0373 0335;
    1596 0236 0199 0467 0354 0337 0262 0532 0499 0494 0167 0498 0571 0217 0547 0296
];

% Plot the mixed signals
figure;
for i = 1:size(mixedSignals, 1)
    subplot(size(mixedSignals, 1), 1, i);
    plot(mixedSignals(i, :));
    title(['Mixed Signal ' num2str(i)]);
end

% Apply FastICA
% Add FastICA toolbox to path if not already added
% You can download it from: https://research.ics.aalto.fi/ica/fastica/
addpath('fastica');

% Run FastICA
[separatedSignals, A, W] = fastica(mixedSignals, 'numOfIC', size(mixedSignals, 1));

% Plot the separated signals
figure;
for i = 1:size(separatedSignals, 1)
    subplot(size(separatedSignals, 1), 1, i);
    plot(separatedSignals(i, :));
    title(['Separated Signal ' num2str(i)]);
end

% Results
disp('Mixing Matrix (Unknown):');
disp('Mixing matrix is not provided, only estimates available.');
disp('Estimated Mixing Matrix:');
disp(A);
disp('Estimated Separating Matrix:');
disp(W);

% Clean up
rmpath('fastica');
```


Mixed Signal 1
Mixed Signal 2
Mixed Signal 3
Mixed Signal 4
Mixed Signal 5
Mixed Signal 6
Mixed Signal 7