```
function result = vit_dec_hdl4print(enc_data, m, n, k, nu, state, generator, tb_length, radix, bitwise)
%function result = vit_dec_hdl(enc_data, m, n, k, nu, state, generator,
%tb_length, radix, bitwise)
%only support k = 1 now
%radix = 2: radix-2 ACS; radix = 4: radix-4 ACS

%param
global branch_metric tb_ram counter2b_wr counter2b_tb counter2b_dec counter6b counter6bn tb_state dec_state decode_tmp;

enc_length = length(enc_data);
if enc_length ~= radix
    disp('err length');
end;
r = log2(radix);
hamm_calc_index = (dec2bin([0 : 2^(n*r)-1]',n*r) - 48)*(2^bitwise-1);
hamm_dist = zeros(1, 2^(n*r));
branch_metric_calc = zeros(radix, state);
decision = zeros(1, state);
decision_bi = zeros(state, r);
hamm_temp = zeros((n*r), 2^(n*r));
wr_addr = counter2b_wr*tb_length + counter6b;
tb_addr = counter2b_tb*tb_length + counter6bn;
dec_addr = counter2b_dec*tb_length + counter6bn;

%Forward ACS Unit
input_temp = dec2bin(enc_data, bitwise) - 48;

%Hamming distance calculation
for cnt_ham = 1 : 2^(n*r)
    for cnt_ham1 = 1 : (n*r)
        hamm_calc_index_tmp = dec2bin(hamm_calc_index(cnt_ham, cnt_ham1), bitwise)-48;
        hamm_temp(cnt_ham1, cnt_ham) = bin2dec(char(xor(hamm_calc_index_tmp, input_temp(cnt_ham1, :))+48));
    end
    hamm_dist(cnt_ham) = sum(hamm_temp(:, cnt_ham), 1);
end;

%Add
for cnt1 = 0 : state-1
    for cnt2 = 0 : radix - 1
        branch_metric_index = mod(cnt1*radix+cnt2, state);
        hamm_dist_index = next_state_output(branch_metric_index, cnt1, generator, nu, radix, n);
        branch_metric_calc(cnt2 + 1, cnt1 + 1) = branch_metric(branch_metric_index + 1) + hamm_dist(hamm_dist_index + 1);
    end;
end;

%Compare and Select
for cnt1 = 0 : state-1
    [branch_metric(cnt1 + 1) decision(cnt1 + 1)] = min(branch_metric_calc(:, cnt1 + 1));
end;
decision_bi = dec2bin(decision-1, r) - 48;

%Write ram
tb_ram(:,[wr_addr+1:wr_addr+r]) = decision_bi;

%Backward Traceback
tb_ram_index = bin2dec(char(tb_state+48));
tb_temp = tb_ram(tb_ram_index+1, [tb_addr:-1:tb_addr-r+1]);
tb_state = [tb_state(1+r:end), tb_temp(end:-1:1)];
dec_ram_index = bin2dec(char(dec_state+48));
result_temp = tb_ram(dec_ram_index+1, [dec_addr:-1:dec_addr-r+1]);
dec_state = [dec_state(1+r:end), result_temp(end:-1:1)];


if counter6b == 64-r
    counter2b_wr = mod(counter2b_wr + 1, 4); %4 ram design
    result = [result_temp, decode_tmp];
    decode_tmp = [];
    dec_state = tb_state;
else
    result = [];
    decode_tmp = [result_temp, decode_tmp];
end;
counter2b_tb = mod(counter2b_wr + 3, 4);
counter2b_dec = mod(counter2b_wr + 1, 4);

counter6b = mod(counter6b + r, tb_length);
counter6bn = tb_length - counter6b;

```