Real-Time Digital Signal Processing
Prof. Ratna G. N.
Department of Electrical Engineering
Indian Institute of Science – Bengaluru

Lecture – 50
M3U32 - Discrete Cosine Transform - IV

Welcome back to Real-Time, Digital Signal Processing Course. So, in the last class we are discussing about the Discrete Cosine Transform.

(Refer Slide Time: 00:34)

So, we will continue today also on the discrete cosine transform. So, in the DCT III we covered a little on quantization and then how we are going to have the quantization matrix too. That is eliminate the lower part of the coefficients that is AC coefficients in DCT matrix.
And later on, how we did the run length coding and then in the process of recovering will be using the Huffman coding.

These are the things what we considered in the last class. So, today we will see how we are going to consider the DCT implementation in hardware.

(Refer Slide Time: 01:13)

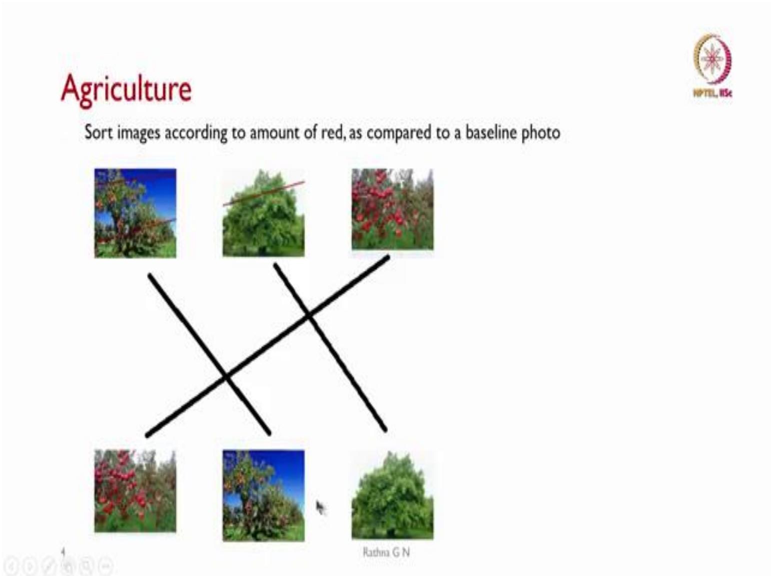
So, one of the application, we said it is in jpeg applications. So that is in the analysis program that will be running on the that is our UWB parallel computing labs cluster has the thing and then potential uses are in this lab what they use is for the application is in the agriculture. That is webcam is go images can be analysed for fruit that is ready to be harvested and workers can then save time by only going to areas that have fruit.

This is one of the examples or application what it is given in the web basically. And the other application what they use it is in the parking. So, what is that? It is, webcams can be placed in a packing parking garage and their images can be used to identify areas with open spaces. So, this is one of the example in the parking lot you can identify if there is any empty space then you can go and park your vehicles there.

The other one is in the web search. What is that? Images can be found over the web that are similar to an input image and images can be analysed then stored as keywords to be used in searches. So, all we usually type the basically for web searching and if you are interested in some part of the images, if you have like the thing, so, you want to see that similar ones. Then you can keep that as a reference and then you can find out how many of them are available.

As an example, in some of the applications if you want to run your machine learning algorithm, you need lot of images of that variety. So, it is in that case it helps us to give up search for typical those images.

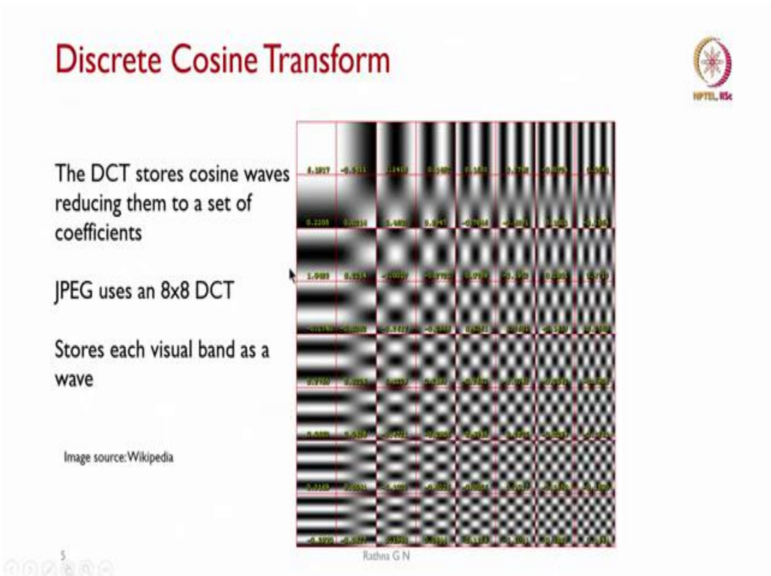
(Refer Slide Time: 03:26)



So, as you can see in the agriculture, so that is how we are going to sort the image according to amount of red as a compared to a baseline photo. So, you will be seeing that whether you want to segregate them. So that is based on the red if it is intensity is more you want to have it as first and later on, the next one and then which has the minimum. So, this is one of it. So, this will show that some of the ripe fruits, if you want to go and then pluck it.

So, you can concentrate on these area to pluck them. So, little whatever later on, you know that how long it will take for this tree to get more ripened one and then here you are not seeing any red in the thing. So, you know that fruits are not ready in the thing. So, you can predict also and then see only those places what you can go and then pluck it. So that you will be saving some of the time of farmers.

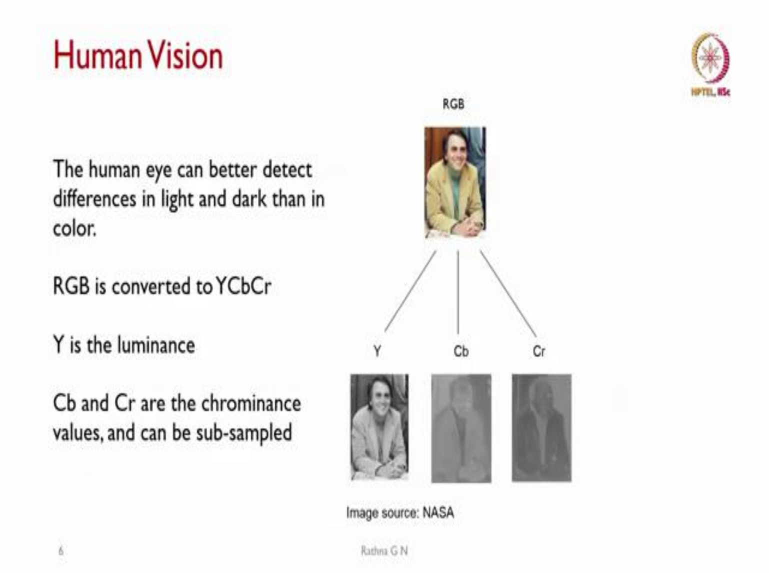
(Refer Slide Time: 04:32)



So, we know that this is how we have introduced our DCT that is discrete cosine transform. So, we know that it is going to store the cosine waves, reduce them to a set of coefficients what we have given the values here and we know that jpeg uses an 8 by 8 DCT. So, this thing what we will call it as 8 by 8 our reference image basically basis function what we call it. So, we will be having 8 by 8 basis function to find out our, will call it as coefficients.

So, visual band as a wave what it is shown here? So, this is from the Wikipedia source. What you will, you can go for more detailed one in that to see it.

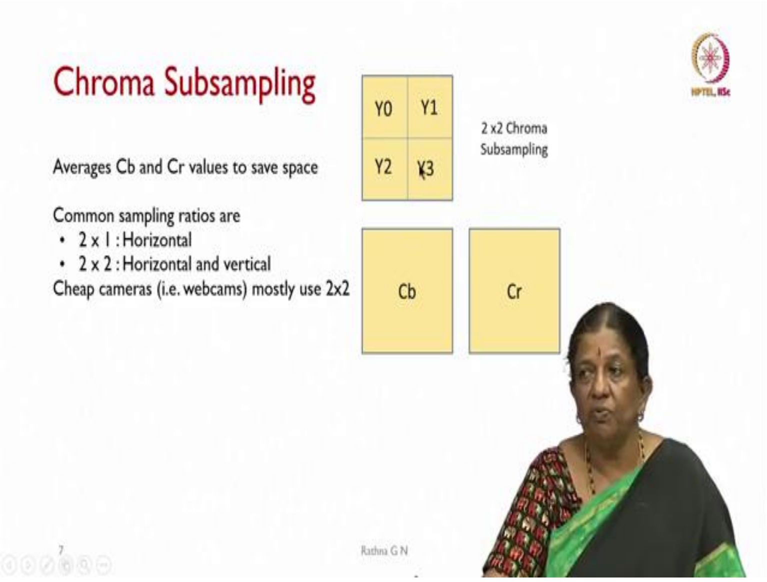
(Refer Slide Time: 05:26)



So now, coming to the human vision. So, what is it? We said already in the previous class, the better detect differences in light and dark than in colour images as you have this is a RGB image. So, we have red, green and then blue. So, if you bifurcate them as luminance as we discussed in the last class. So, you will be seeing that you are able to identify the person in that is Y component.

And then chrominance that is Cb and Cr they are little faded, so, what we said was we can do the compression in this domain. So that is what it is given luminance and Cb and Cr are the chrominance values and can be sub-sampled first basically.

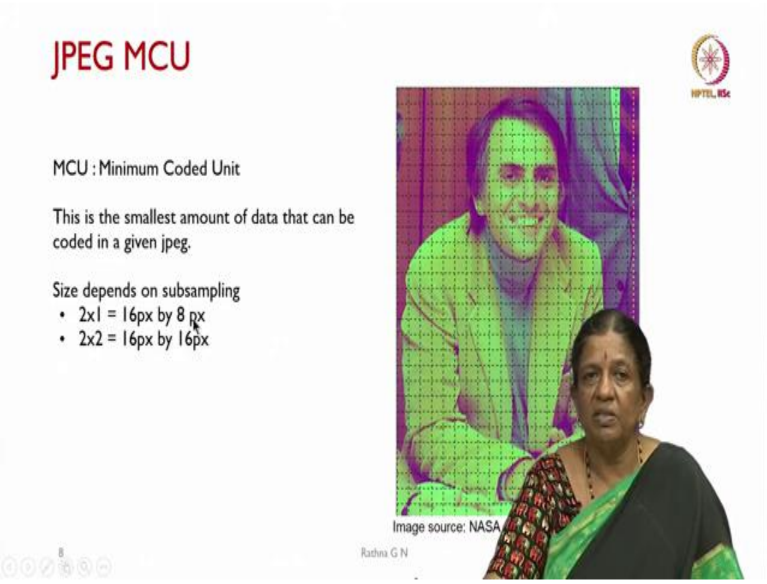
(Refer Slide Time: 06:17)



So, what is the subsampling? We are going to do it, so, we are going to keep the chrominance Y0, Y1, Y2, Y3 as it is that is. And then we can do 2 by 2 that is chrome's subsampling, chromo subsampling. So, how we are going to do? Averages Cb and Cr value to save space basically. Common sampling ratios are 2 by 1 in the horizontal or we can have 2 by 2 both horizontal and then vertical.

So, what we say is cheap cameras that is webcams, mostly use this 2 by 2 standard subsampling basically. So, these are the Cb Cr, so, you will be subsampling to either 2 by 1 or 2 by 2. So, as it says, webcams convert it into 2 by 2 for chip cameras.

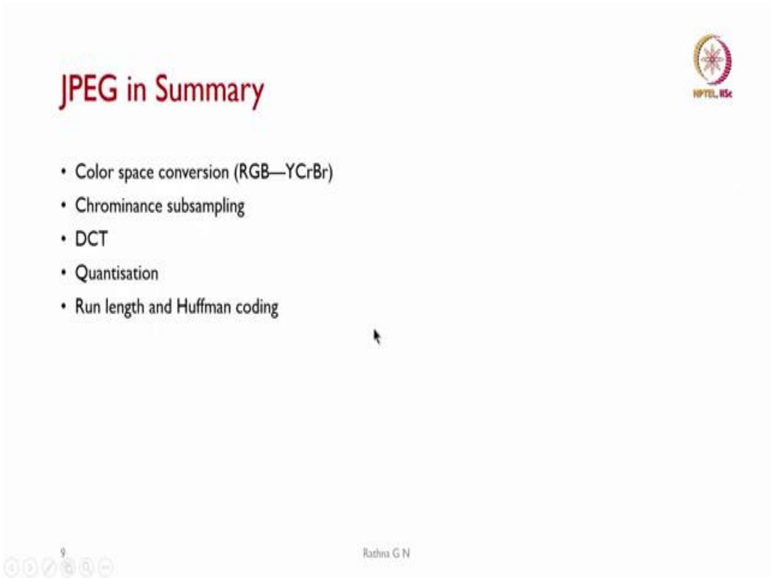
(Refer Slide Time: 07:13)



So, what is that? That is what it says, is jpeg minimum coded unit. So, this is the smallest amount of the data that can be coded in a given jpeg. So, how do you say that? That is size is going to depend on subsampling? So, if we are doing 2 by 1 then it is 16 pixel by 8 pixel what has to be stored. So, if we are considering 2 by 2 then it is going to be 16 pixel by 16 pixel what you will be taking it.

So, this is an image taken from NASA, what it is projected to show that how the subsampling can be done.

(Refer Slide Time: 07:54)



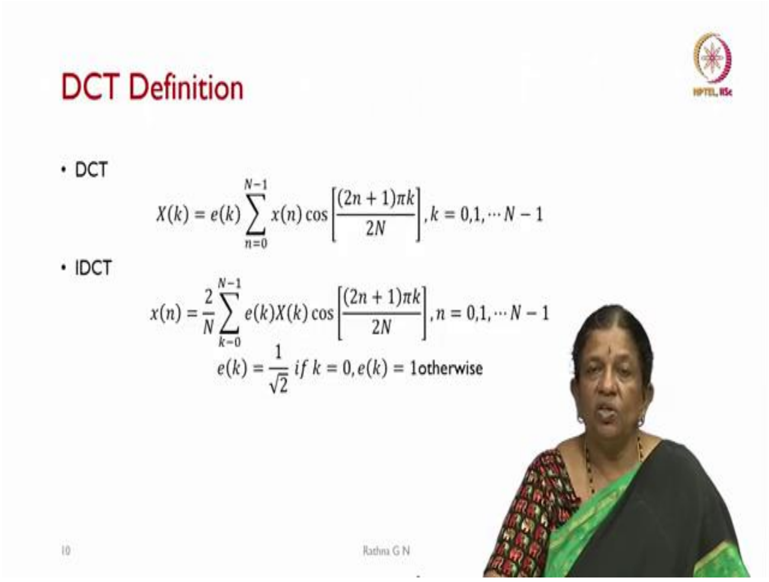
So, to give you a jpeg, the steps that are involved in summary, so, first, what we do is a colour space conversion. So, from RGB we will be converting it into luminance chrominance and the YCrBr basically and then will be doing this chrominance subsampling is going to happen. Then later on we are going to do the DCT. So, we know that just the DCT is not going to give us the reduced size of the image.

Later on, we will go for the quantization, so, as we said in the previous class, quantization we can use P = 1, 2 or 4, depending on how many values you want to store it, so that is decides the quantization. How much you want to have it? One has to be careful that near the edges, so, we may lose the information so, one has to look into that when you are doing the quantization.

Then once the quantization is done so, will be in the DCT we will be going for the run length coding in the IDCT because we have to reconstruct the image. So, we will go for the Huffman coding what we had discussed in the last classes and then the reverse process is going to happen. That is whatever the context signal has to be requantized and then IDCT. What will be doing it and then we will go for the chrominance subsampling in the reverse direction.

And then we will convert from YCrBr to RGB image and then we will be displaying it. So, this is how jpeg in summary looks like.

(Refer Slide Time: 09:49)

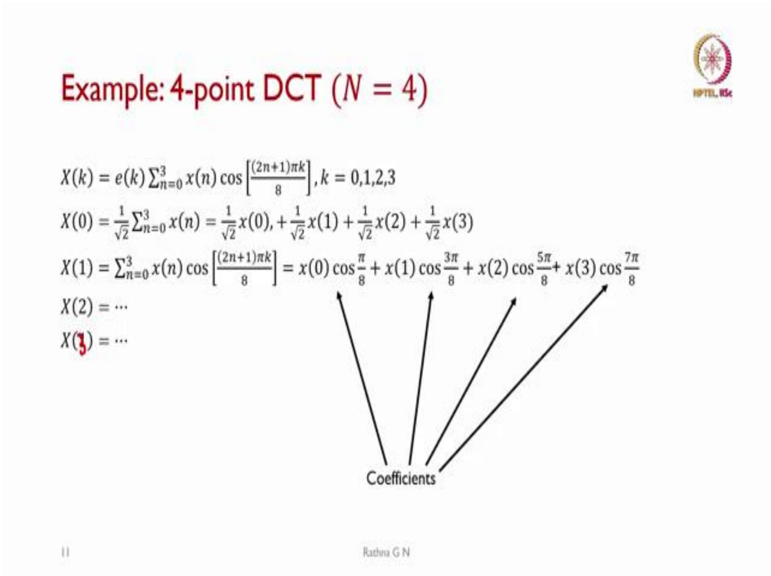


So now, we will see that how we are going to implement this in hardware. So, to give again a flavour of what is DCT, IDCT the code has been given it is one dimensional what we are considering it? So, we have e of k into n = 0 to N – 1 x of n into cos 2n + 1 pi k divided by 2N, N is we can choose it as 8 by 8 or it will be the size of the image what we are considering. So, if it is, we are using it in the direct DCT form, k will be going 0 to n – 1.

That is our image size, n by N what we have considered here. So, in the reverse process IDCT. We consider 2 by N instead of multiplying by root 2 by N in both the cases we consider only in the IDCT form 2 by N. So, this will be going from 0 to n – 1. So, it will be e of k into x of k what you have computed and cos 2n + 1 pi k divided by N and N will be varying between 0 to N – 1 in this case.

And our constant e of k is given by 1 by root 2, if k = 0 and it is going to be 1 for other coefficients.

(Refer Slide Time: 11:14)

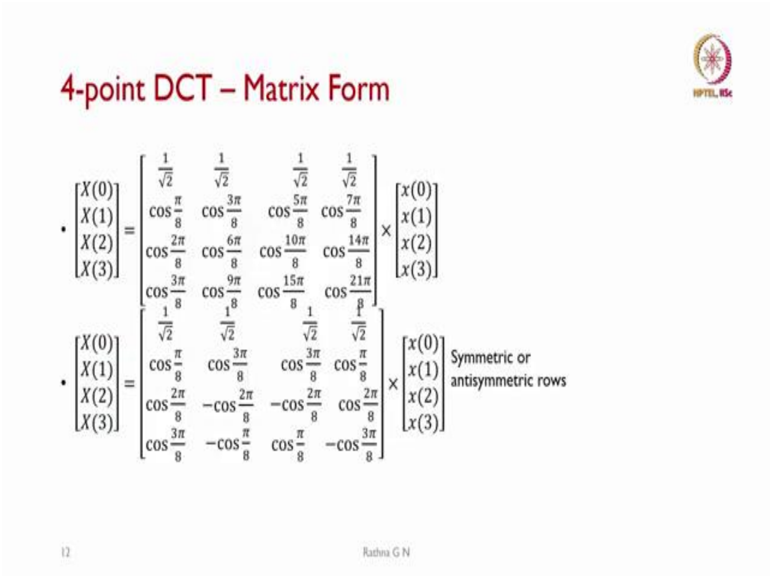


Now, we will see how we are going to derive the thing? So, the first is will take a 4 point DCT that is simpler just how we did for N = 4 for RDFT. So, we are considering this and then we will be expanding our summation, so that we can combine our coefficients. So, the first is x of k is e of k = n = 0 to 3, x of n this equation and k will be varying between 0 to 3. So now, x of 0 we know that 1 by root 2 summation of x of n.

So, which is given by 1 by root 2 into x of 0 + 1 by root 2 and x of 1 so on 1 by root 2 into x of 3. So, all the x0 to x3 has to be sum and then we have to multiply by root 2 to get x of 0. To get x of 1 so, we will expand the thing this equation by substituting n = 0 to 3 in this case. So, which is going to be x of 0 into cos pi by 8 and x of 1 into cos 2 pi by 8 + x of 2 into cos 3 pi by 8 + x of 3 into cos 7 pi by 8.

So, we are substituting n = 0 to 3 in this, assuming our k = 1. So now, what we call? All these are cos functions what we have we call it as coefficients. So, if you want, you can expand x of 2 and x of 3 also, to get 0 to 3 coefficients. So, here this is x of 3 what you have to expand it and then write down those coefficients.

(Refer Slide Time: 13:05)



So now, coming to in the matrix form, if we substitute these values as we have got it expanded one. So, we will see how it is going to look like. So, my capital X of 0, X of 1, X of 2, X of 3 are written here and we know that the first coefficients is going to be 1 by root 2. So which is going to be multiplied by x of 0, x of 1, x of 2 into x of 3, as input. And then the next second row is going to be pi by 8, 3 pi by 8, 5 pi by 8.

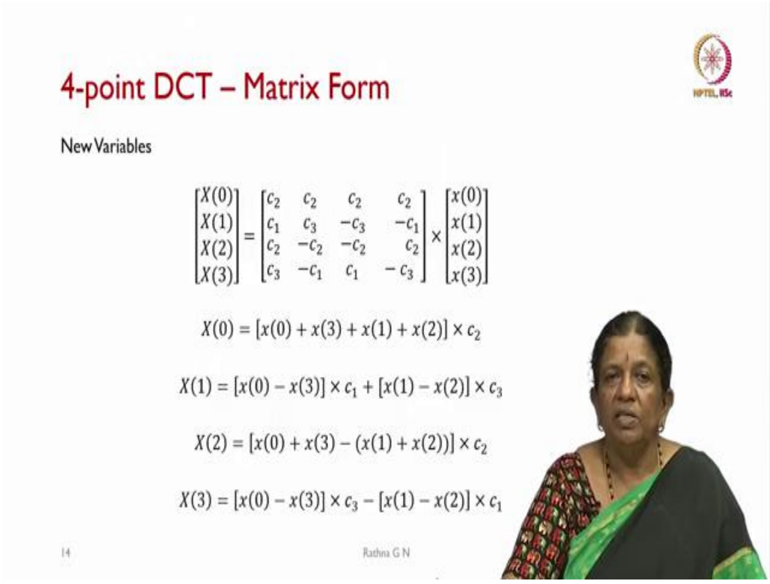
So, you will be adding 2 pi for all the things. So, the last one will be cos 7 pi by 8. So, you will be seeing in the third row it is going to start from 2 pi by 8. And then, you are going to add this thing what is that? 4 pi to this so, you get it as 6 pi + 4 is 10 pi and then 14 pi. So, because that is multiplication what we have it 2 into 2 because X of 2 what we are doing it 2 into 2 is 4. So, it will be + 4 pi.

So, it is going to be 6 pi same as that. Now, the next one is cos 3 pi by 8 and then you will be adding 6 pi to that. So, 3 into 2 is 6 so, it will be 9 pi by 8 and so on. So now, because we know that we can simplify this, so, when you write the after simplification, so, what we are going to get it? So, in terms of negative what you will be representing because you will be doing 2 pi + 4 pi so which is nothing but – cos 2 pi by 8.

So, by substituting all this, these are the coefficients what we will get it. So, we say whether it is symmetric or asymmetric rows what we are going to have it. So now, coming further, continuing with the matrix form, so, we will represent we call this is a c 2 coefficients what we can represent for 1 by root. So, the next one is pi by 8 is c 1 coefficient. 3 pi by 8 is c 3 coefficient. Now, we can represent this as –c 3 and then this is –c 1.

So, the next stage is going to be c 2 coefficients, cos 2 pi by 8 and then –c 2, –c 2 and then c 2. So, the same way, the last row, what we can represent it as c 3, –c 1, c 1 and then –c 3. So, we will know that 4-point DCT needs 4 coefficients, so that is c 1 to c 3 n – 1 coefficients. That is what it is going to be used. It is shown here, n – 1 is 3 coefficients that is c 1, c 2, c 3 what we need to represent our coefficients here.

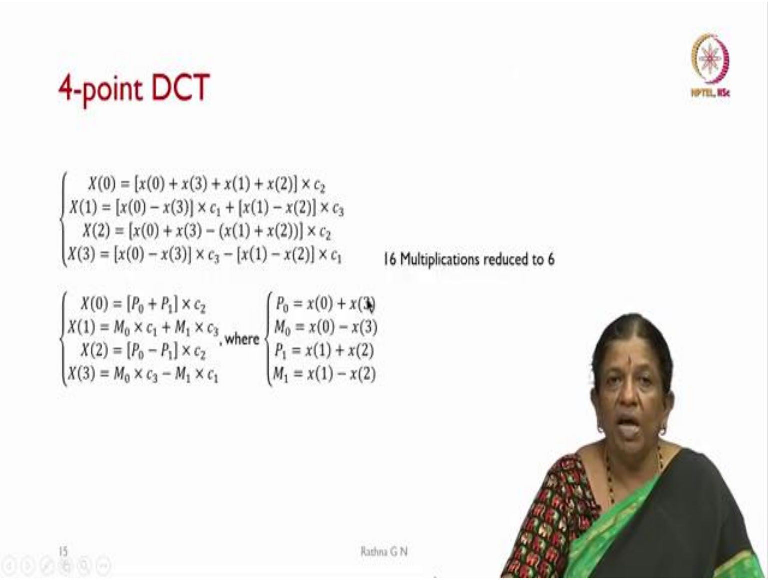
(Refer Slide Time: 16:25)



So, the some of the new variables how we can modify the thing is so, what we have is here we know that it is c 2. This is x of 0 and we have rewritten the thing. Now, what is our x of 0? We have to sum up all this and multiply by c 2 and same way what we will be doing for X of 1 is, it is x of 0 – x of 3 into c 1 + x of 1 – x of 2 into c 3. Because we will be combining this c 1 and then this c 1 and this c 3 and this c 3 so that our number of multiplication is going to reduce.

So, it will be X0 – X3 will subtract this and multiply with 1 coefficient. Same way for the other thing, so, you will be seeing for X2 and X3 the same way you can combine them.

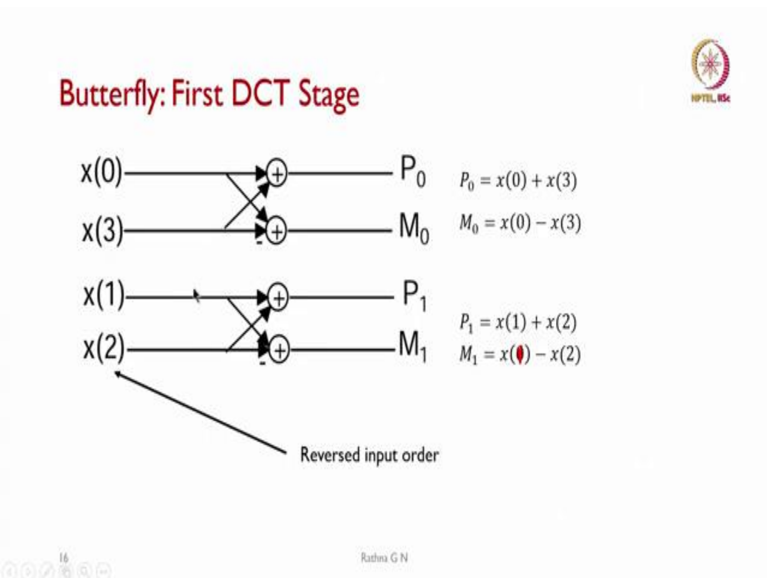
(Refer Slide Time: 17:23)



So now, what happens? So, we will write it as from what we have is original we need because it is a 4 by 4 matrix. We need 6 multiplication, so, what we can reduce this one to 6 multiplications what you are seeing it that is. 1, 2, 3, 4, 5 and then 6. So, we have reduced from 16 multiplications to 6 multiplications but addition is going to remain the same. So now, we will see intermediate products how we will be representing it.

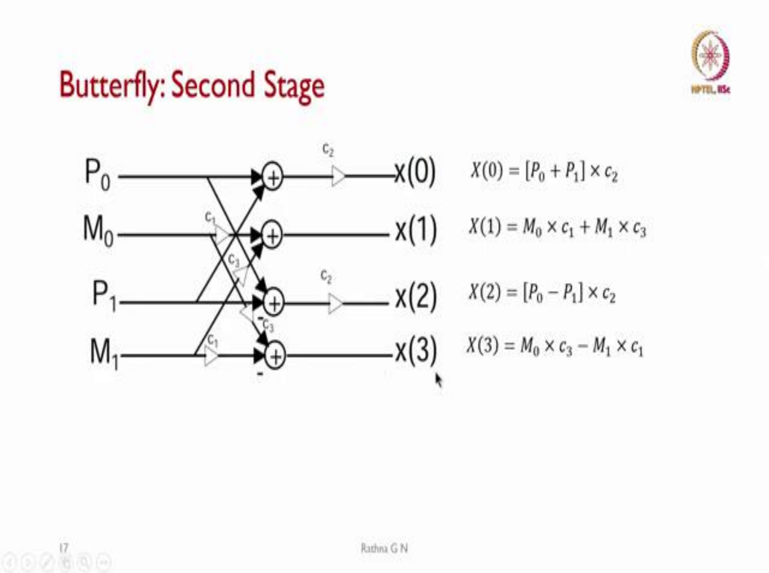
We call it as x of 0 is given by P 0 + P 1 into our c 2. So, what is P 0? Which is going to be x0 + x3 and then M 0 that is because we are calculating the negative part of it x0 – x3 and P 1 will be x of 1 + x of 2 and M 1 is represented as x of 1 – x of 2. Then we will be reordering. Thus, so, we will be seeing x of 0 is given by this equation x of 1, x of 2, x of 3 in terms of P 0, M 0, P 1 and M 1 is given by this equation.

(Refer Slide Time: 18:45)



So, how we are going to represent in the butterfly form. So, we know that X0 and then X3 is going to be minus, first one is X0 we are going to take this one X3 to this x0 + x3 is my P 0 output and the later on x0 – x3 will be my M 0. Same way we will put it for X1 and then X2. So that is X1 + X2 will be p 1 and x1 – x2 will be my M 1 here. So, what is happened here? We have the reversed input order just like RDFT, so, we are having x 0, x 3, x1 and then x2.

(Refer Slide Time: 19:41)

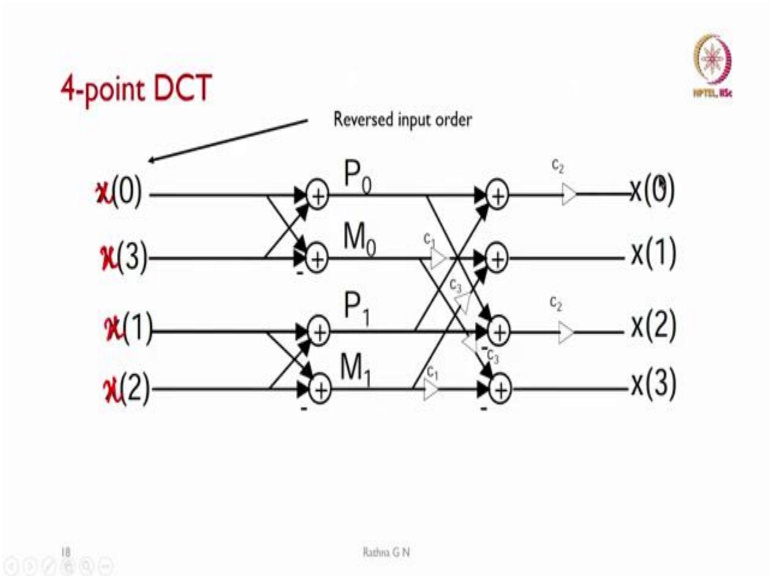


So, coming to the second stage. What is the thing is going to happen? So, we have P 0 and M 0 and P 1 and then M 1. So, next stage is, we have X0 is P 0 + P 1 into c 2. So, we will be multiplying with c 2 coefficients in the last stage. After adding with P 0 M 0, we will be getting X0. So, the next one is what we have is x of 1 is M 0 into c 1 + M 1 into c 3. So, you have to have M 0 with c 1 coefficient and then M 1 with c 3 coefficient.

So, this is what you will be c 3 coefficient, so, you multiply both of them, add them you will be getting your X1. Now, x2 is what P 0 – P 1 into c 2. You are taking P 0 here minus what you are going to do. P 1 multiplied by c 2 will be given output as X2 and X3 we know that M 0 into c 3 – M 1 into c 1. So, what we have? M 0 into what we have is here it is multiplied by c 3 here and then M 1 is multiplied by c 1.

So, after this is a subtraction that is, minus what we have it. So, we will be getting X of 3 as the output. This is how we can write our butterfly structure.

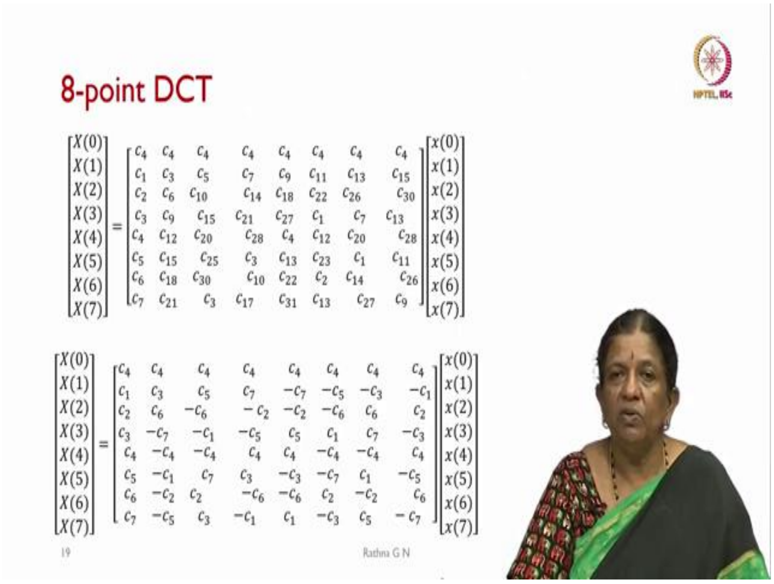
(Refer Slide Time: 21:13)



So, the final thing what we have 4-point DCT is given by this complete butterfly structure, X0, X3, X1 and X2 are the inputs, so, we will be getting. Actually, input is going to be small x0, x3, x1 and then x2 output is going to be capital X0, X1, X2, X3 in line with our FFT what it is shown by combining these two butterflies. So, intermediate you will be seeing that P 0, M 0, P 1, M 1.

And these are the output you will be seeing there in order x0, x1, x2, x3 and input has got reversed in this case.

(Refer Slide Time: 21:59)

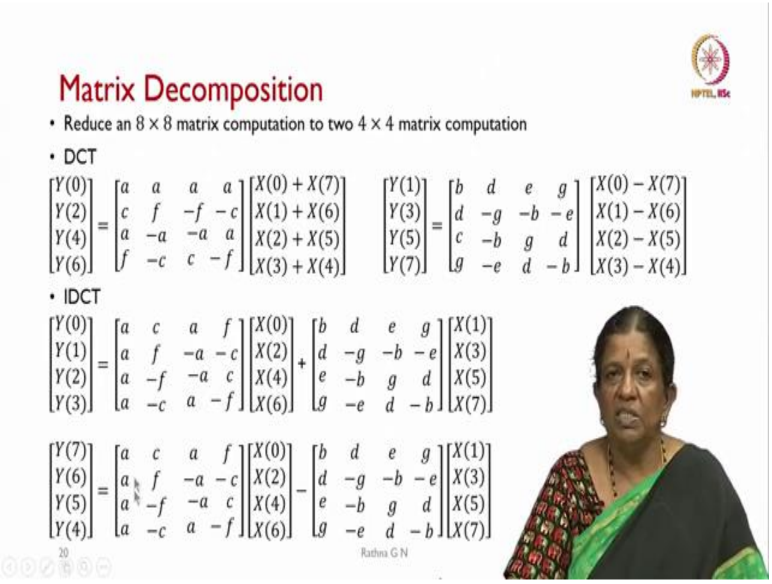


So now, extending the same thing from 4-point to 8-point. How we did with respect to DFT? So, we will do it for 8 point DCT, so, you will be seeing that input that is output is capital X0 to X7 input is small x0 to x7. What we have it, in terms of coefficients you will be seeing that from c 2 because it is n by 2 what we are going to have it that is 8 by 2 is 4. So, the first row is going to be c 4 coefficients you will be having it.

After that you will have c 1, c 3, c 5, c 7, c 9, c 11, c 13 and then c 15. So, you will be writing this way. So, all the coefficients. You can write it in this fashion and then later on as we see the symmetry thing what you will be looking into the thing. So, will be reorganizing my will be seeing that my c 10 is going to be –c 6. So, how many coefficients I need it here, can you guess it? We said in 4-point 3 3 coefficients in 8 point we need only 7 coefficients.

So that is c 1 to c 7 what we are supposed to have it. So, we have to organize in that manner, when we do that you can see the simplified matrix what it is given in this case. And then you will be seeing that these are the outputs and multiplied with x0 to x7 in order.

(Refer Slide Time: 23:42)



So, how we are going to represent this in matrix form. So, one of the famous thing is what they call it as a c f and then b d e g coefficients form to put it in f p g a. So, you can use the butterfly structure last I will be showing it, otherwise most of the f p g a implementation use this format. That is we call it as instead of X0 output will be Y0, Y2, Y4, Y6. So, this is our DCT or even coefficients what we are going to combine with the coefficients a c f.

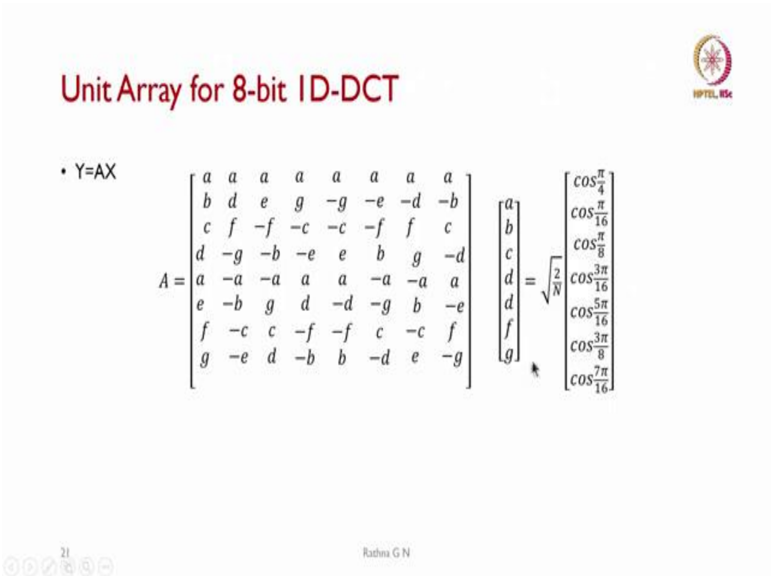
So that is what, for the even and you will be adding X0 + X7, X1 + X6 and X2 + X5 and then X3 + X4 and these are the odd coefficients bottom. So, you can put them in bottom Y1, Y3 Y5 and then Y7. So, they need b d e g coefficients to compute and the input of this will be X0 –

X7 and X1 – X6 so on and this one. So now, to compute with these coefficients back IDCT. So, how you will be using in the, that is Y0, Y1, Y2, Y3 what you will be calculating it.

So, you will be using the a c f coefficient in the top portion but you will be seeing that input is going to be that is bit reversed that is X0, X2, X4, X6 not completely bit reversed in this case. Even anyway what you will be considering it here, same as our FFT. So, here it will be X1 X3 X5 and then X7. What will be using the thing and then, when you multiply with this b d e g coefficients and add them up, you will be getting the Y0 to Y3.

Then what you are going to do is do the same thing but you have to subtract with this b d g coefficients which are getting multiplied with of odd numbers. So, you will be getting the higher that is Y4 to Y7. So, you will be seeing that it has come in the reverse order. So, Y7 will be first and then Y4 will be coming later on. So, this is how you would be reconstructing with the same coefficients both in DCT, what you will be using it and then in the inverse DCT also.

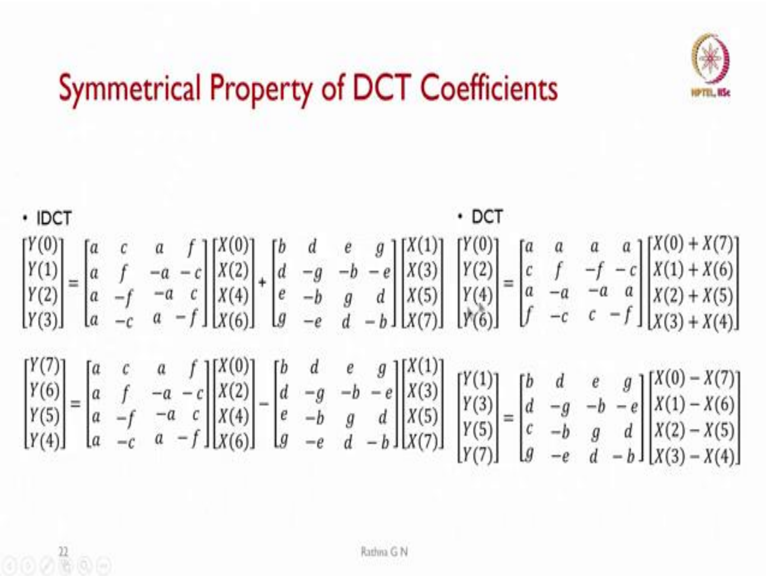
(Refer Slide Time: 26:29)



How we are going to do that so, you will be putting that is, if you are putting 8 bit 1D-DCT when you are putting in this format. So that is Y is equal to our matrix A into X. The matrix with coefficients whatever we have discussed in the previous one that is a c f and then b d e g. So, you will be seeing that all these are a coefficients and these are b d e g –g –e –d and –b and then the rest of the coefficients are getting filled like this.

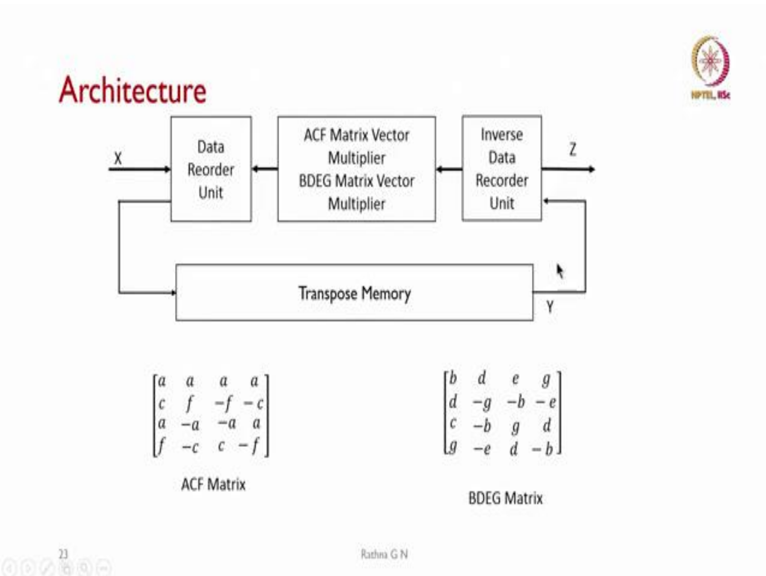
You can go back and then cross verify it and then what are the our a b c d f g coefficients? So, it is nothing but root 2 by n, so, a is cos pi by 4, b is going to be cos pi by 16 and c is this one cos pi by 8 and then g is going to be cos 7 pi by 16. So, this is for the 8 by 8 matrix. So, you can substitute it will be much faster and then computation.

(Refer Slide Time: 27:40)



So, the symmetrical property of that is what the DCT coefficients what you have seen the thing IDCT after adding it DCT part what we said by multiplying it what will be getting it. So, the same thing again it is shown that how from IDCT, DCT or DCT to IDCT what you will be traversing with the same type of coefficients.

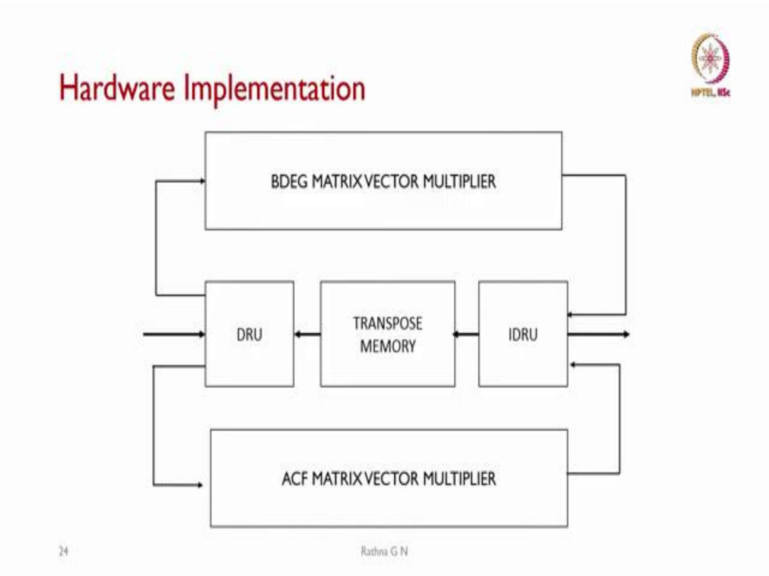
(Refer Slide Time: 28:02)



So, how the architecture inside f p g a is going to look like. So, this will be X is the input that is we call it as data recorder unit and then we will be having the ACF matrix vector multiplier and the other one is BDEG, matrix vector multiplier and then we are going to have the inverse data recorder unit basically that is output is going to be Z coming out of it and then what is Z?

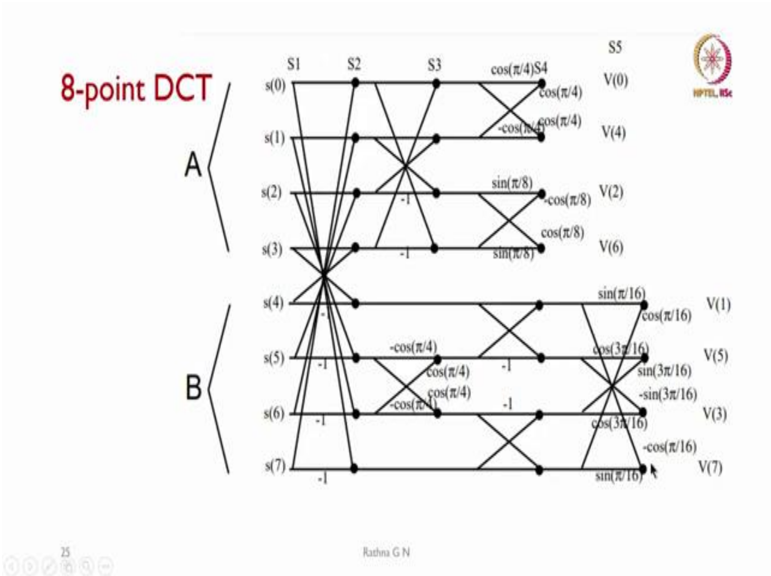
So, the same data you will be doing the transpose the memory and then get it as Y and feed it in the reverse direction. So, this is our ACF matrix and this is our BDEG matrix. So, you will be you can go from a Y to X or X to Z. So, this will be transposing and then putting it as input and then will be working it out.

(Refer Slide Time: 28:28)



So, this is how the hardware implementation is going to happen in FPGA. This is data recorder unit in the thing. So, you will have the transpose of memory and you have the ACF matrix vector multiplier. Input will be coming to here and this output is going to be given to our inverse data recorder unit here and we will be doing the transpose and then will be getting the BDEG matrix vector multiplier and then we can feed it in and then will be taking the output from here. So, if you are using it in the DCT you can use it in one direction and then, if I want to have the IDCT the same hardware is going to be used in the reverse direction. As we pointed out in the thing.

(Refer Slide Time: 29:45)



So, to give you a flavour of how the 8 point DCT, I want you to work it out how much we have reduced in the 1D-DCT with the butterfly structure. So, you will be seeing that these are the stages what we are going to have it S1, S2, S3 and S4 compared to log 2n stages in FFT. So, we will be having log 2n + 1 for 8 point. As you can see, there is sy stage also will be coming in the outside part of it.

So, this is our s0 and then you will be combining with –s7 as input A and B what you will be showing it as a matrix 4-point what you can take it and then put it across. This is the even side this is the art part of it. So, I want you to work it out so that the output is going to be V0, V4, V2, V6. So, input is in order, output is going to be in the bit reversal format what you will be getting it.

The odd side you will be seeing that it is V1, V5, V3 and V7 and these are the sine and cos function. So, if those who are interested in this paper, I will be uploading it in the think, how they have reduced number of multiplications and additions so that it comes to 13 multiplications In this case, one more paper reduces from 13 multiplications to 11 multiplications, with 27 additions along with it.

So, this is how you can implement it in hardware that is butterfly structure with cos and sine functions, what it has been given and then you can see that to the normal DCT output, whether this output for the 8-point what you will be getting it.

(Refer Slide Time: 32:00)

So, this gives the complete summary of the DCT in hardware. So, have a happy learning and then thank you. Hopefully you have enjoyed this course. So, we will do the summary of the complete course in the next class. What all we covered it And then we will discuss how we can implement it in hardware And then how we have done our labs and other sessions based on the theory part of it to verify our outputs. Thank you.