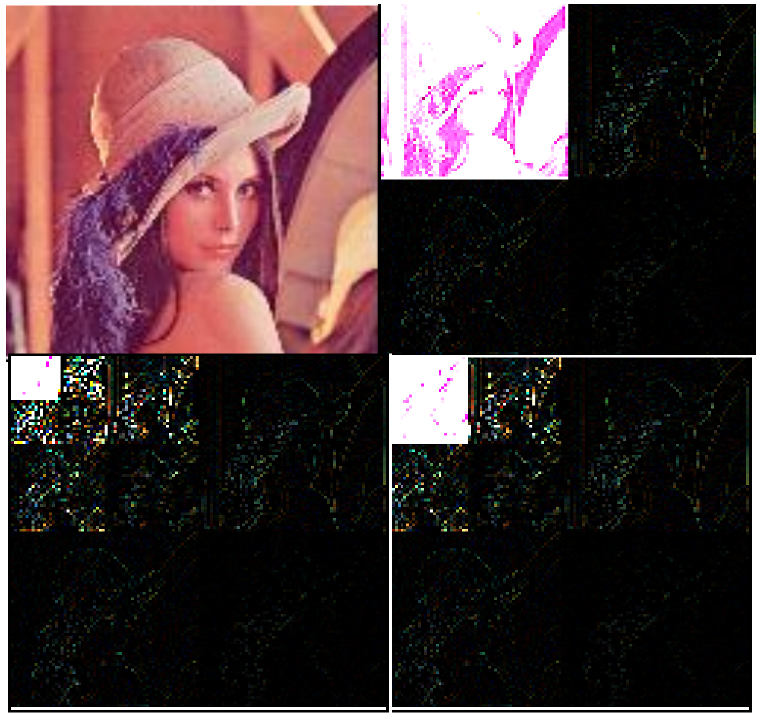
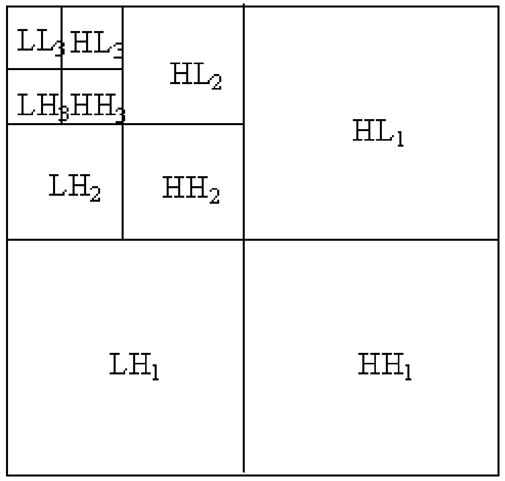
Embedded Zerotree of Wavelet Coefficients (EZW)

- E – The EZW encoder is based on progressive encoding. Progressive encoding is also known as embedded encoding
- Z – A data structure called zero-tree is used in EZW algorithm to encode the data
- W – The EZW encoder is specially designed to use with wavelet transform. It was originally designed to operate on images (2-D signals)

EZW – basic concepts(1)



A Multi-resolution Analysis Example



Lower octave has higher resolution and contains higher frequency information

EZW – basic concepts(2)

The EZW algorithm is based on two observations:

- Natural images in general have a low pass spectrum. When an image is wavelet transformed, the energy in the sub-bands decreases with the scale goes lower (low scale means high resolution), so the wavelet coefficient will, on average, be smaller in the lower levels than in the higher levels.
- Large wavelet coefficients are more important than small wavelet coefficients.

631	544	86	10	-7	29	55	-54
730	655	-13	30	-12	44	41	32
19	23	37	17	-4	–13	-13	39
25	-49	32	-4	9	-23	-17	-35
32	-10	56	-22	-7	-25	40	-10
6	34	-44	4	13	-12	21	24
-12	-2	-8	-24	-42	9	-21	45
13	-3	-16	-15	31	-11	-10	-17

typical wavelet coefficients for a 8*8 block in a real image

EZW – basic concepts(3)

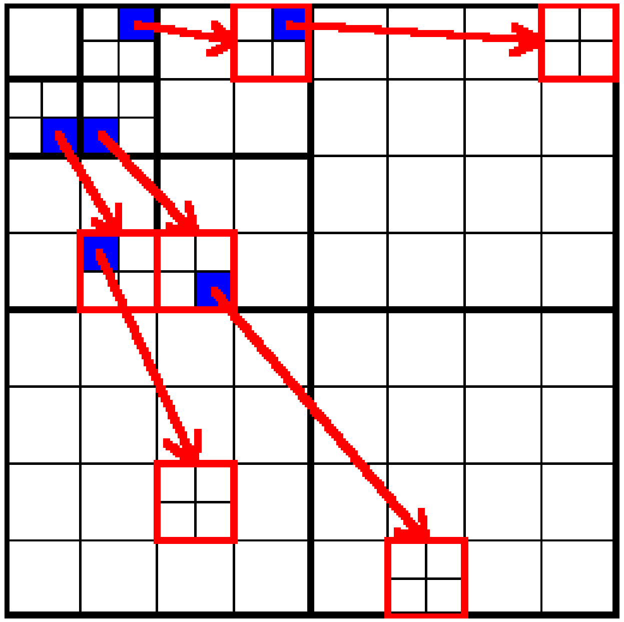
The observations give rise to the basic progressive coding idea:

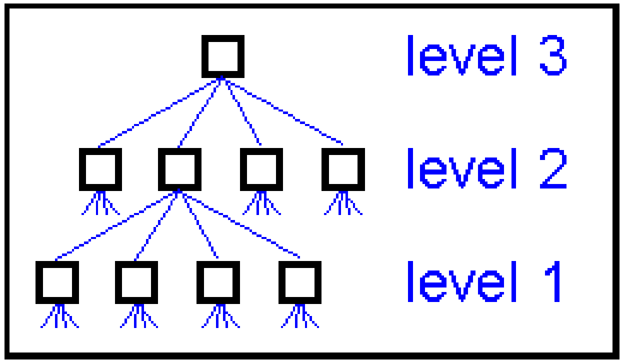
1. We can set a threshold T, if the wavelet coefficient is larger than T, then encode it as 1, otherwise we code it as 0.
2. ‘1’ will be reconstructed as T (or a number larger than T) and ‘0’ will be reconstructed as 0.
3. We then decrease T to a lower value, repeat 1 and 2. So we get finer and finer reconstructed data.

The actual implementation of EZA algorithm should consider :

1. What should we do to the sign of the coefficients. (positive or negative) ? – answer: use P and N
2. Can we code the ‘0’s more efficiently? -- answer: zero-tree
3. How to decide the threshold T and how to reconstruct? –answer: see the algorithm

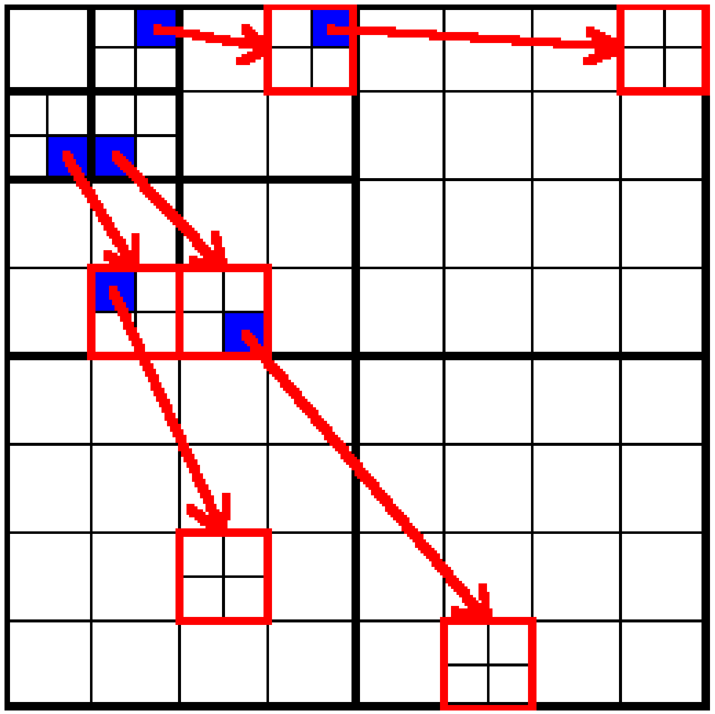
EZW – basic concepts(4)

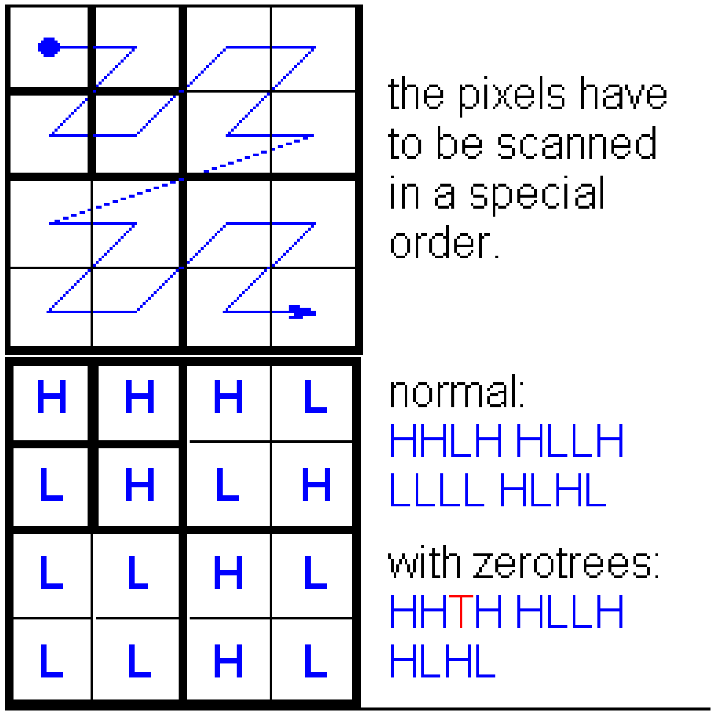




coefficients that are in the same spatial location consist of a ***quad-tree***.

Parent-Child Relationship in a Zerotree





*The relations between wavelet coefficients in different subbands (left), how to scan them (upper right) and the result of using zerotree (lower right) symbols (T) in the coding process. An H means that the coefficient is higher than the threshold and an L means that it is below the threshold. The zerotree symbol (T) replaces the four L's in the lower left part **and** the L in the upper left part.*

EZW – basic concepts(5)

- **The definition of the zero-tree:**

  There are coefficients in different subbands that represent the same spatial location in the image and this spatial relation can be depicted by a quad tree except for the root node at top left corner representing the DC coeeficient which only has three children nodes.

- **Zero-tree Hypothesis**

  ***If a wavelet coefficient c at a coarse scale is insignificant with respect to a given threshold T, i.e. |c|<T then all wavelet coefficients of the same orientation at finer scales are also likely to be insignificant with respect to T*.**

EZW – the algorithm(1)

- First step: The DWT of the entire 2-D image will be computed by FWT
- Second step: Progressively EZW encodes the coefficients by decreasing the threshold
- Third step: Arithmetic coding is used to entropy code the symbols

EZW – the algorithm(2)

In the *dominant_pass*

- All the coefficients are scanned in a special order
- If the coefficient is a zero tree root, it will be encoded as t. All its descendants don't need to be encoded – they will be reconstructed as zero at this threshold level
- If the coefficient itself is insignificant but one of its descendants is significant, it is encoded as z (isolated zero).
- If the coefficient is significant then it is encoded as p (positive) or n (negative) depends on its sign.

This encoding of the zero tree produces significant compression because gray level images resulting from natural sources typically result in DWTs with many ZTR symbols. Each ZTR indicates that no more bits are needed for encoding the descendants of the corresponding coefficient

EZW – the algorithm(3)

At the end of *dominant_pass*

- all the coefficients that are in absolute value larger than the current threshold are extracted and placed without their sign on the subordinate list and their positions in the image are filled with zeroes. This will prevent them from being coded again.

In the *subordinate_pass*

- All the values in the subordinate list are refined. this gives rise to some juggling with uncertainty intervals and it outputs next most significant bit of all the coefficients in the subordinate list.

Example

57	-37	39	-20	3	7	9	10
-29	30	17	33	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	33	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

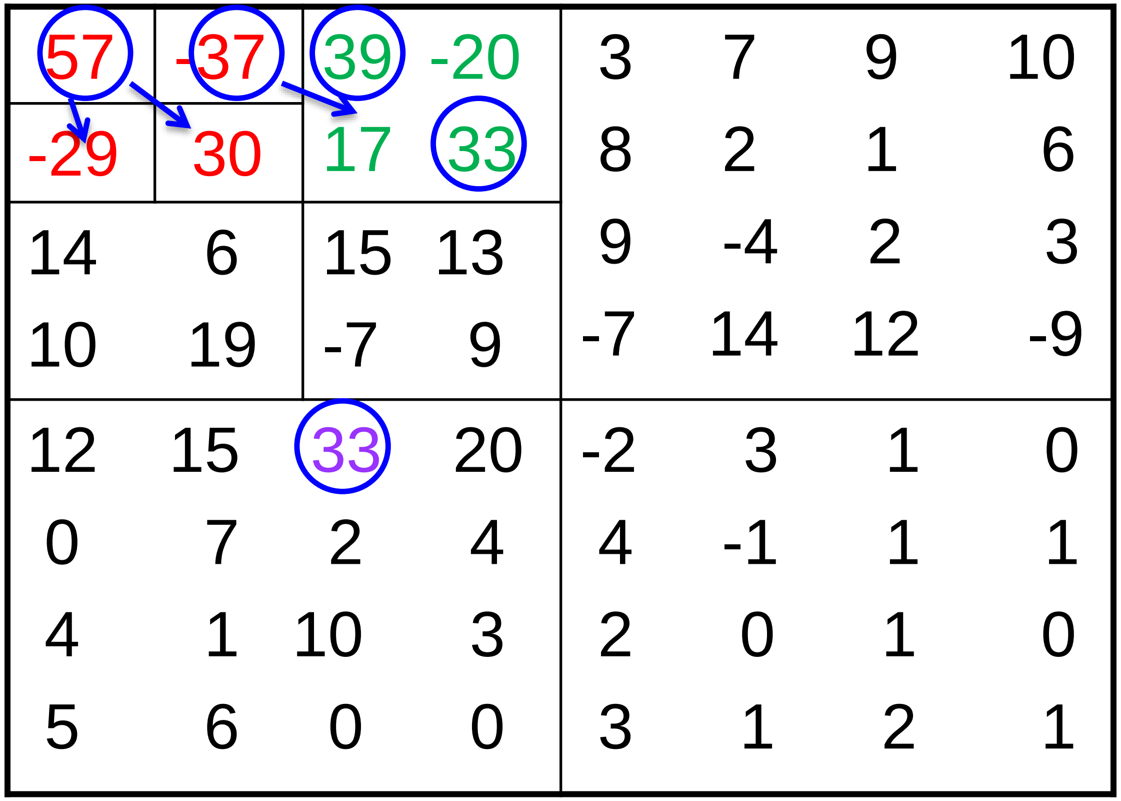
T0 = 32

Example

57	-37	39	-20	3	7	9	10
-29	30	17	33	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	33	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

$T_0 = 32$

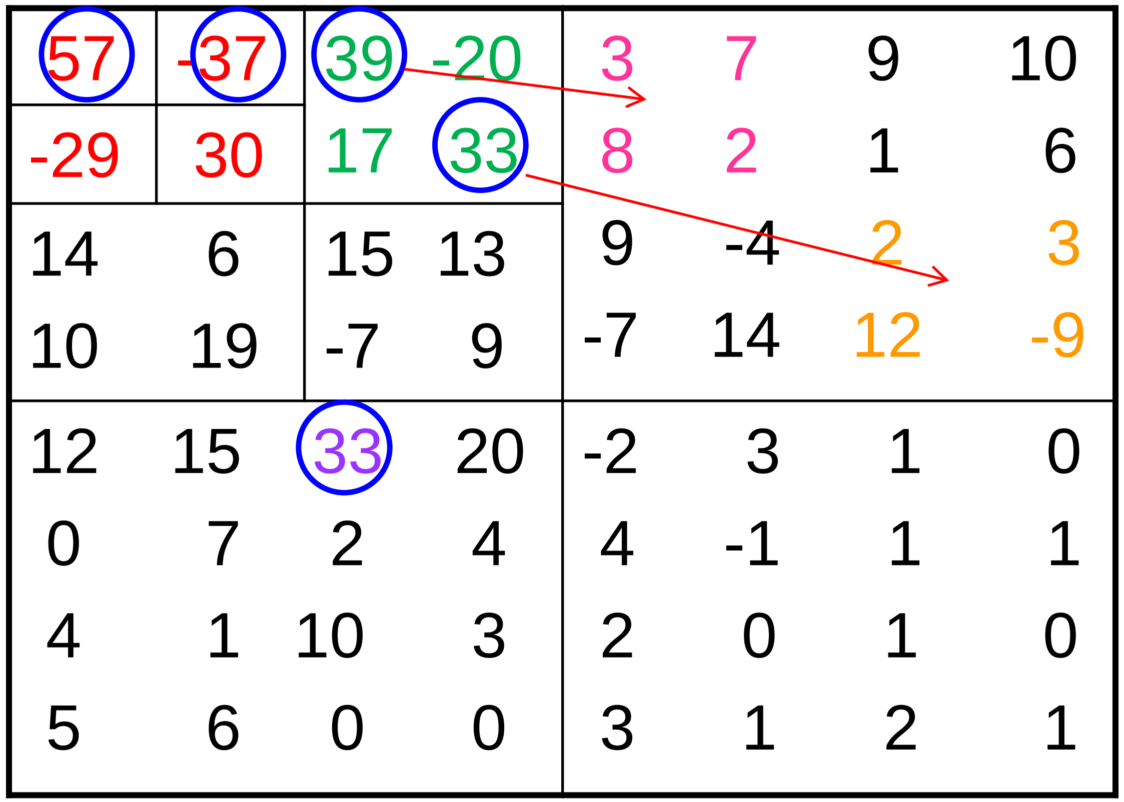
Example



57	-37	39	-20	3	7	9	10
-29	30	17	33	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	33	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

T0 = 32

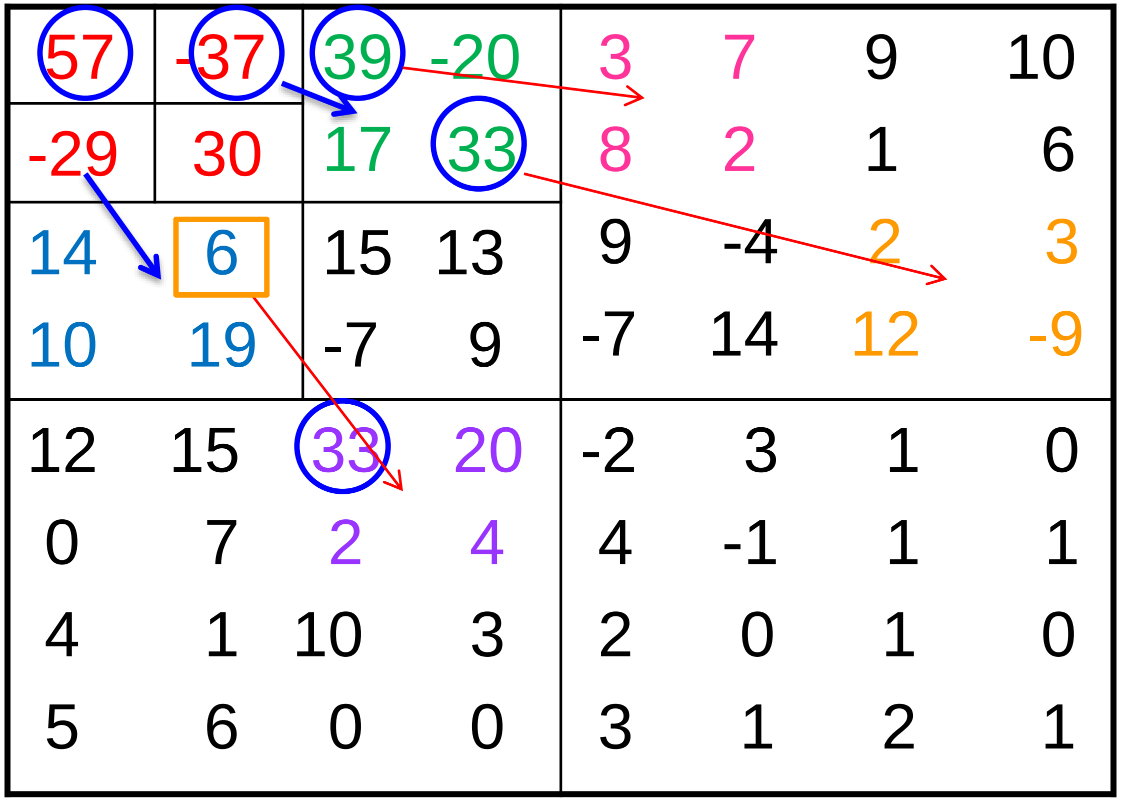
Example



57	-37	39	-20	3	7	9	10
-29	30	17	33	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	33	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

T0 = 32

Example



57	-37	39	-20	3	7	9	10
-29	30	17	33	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	33	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

T0 = 32

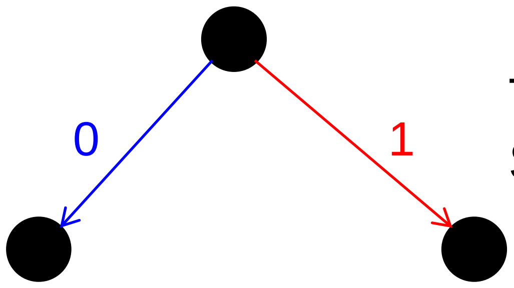
{57, -37, -29, 30, 39, -20, 17, 33, 14, 6, 10, 19, 3, 7, 8, 2, 2, 3, 12, -9, 33, 20, 2,4}

D0: *p n z t p t t p t z t t t t t t t t t t p t t t*

The coefficient −*29 is insignificant, but contains a significant* descendant 33 in LH1. Therefore, it is coded as *z.*

Encoding(T0=32, S0)

T0=32
S0: 0: [32, 48)



T0=32
S0: 1: [48, 64)

{57, -37, -29, 30, 39, -20, 17, 33, 14, 6, 10, 19, 3, 7, 8, 2, 2, 3, 12, -9, 33, 20, 2,4}

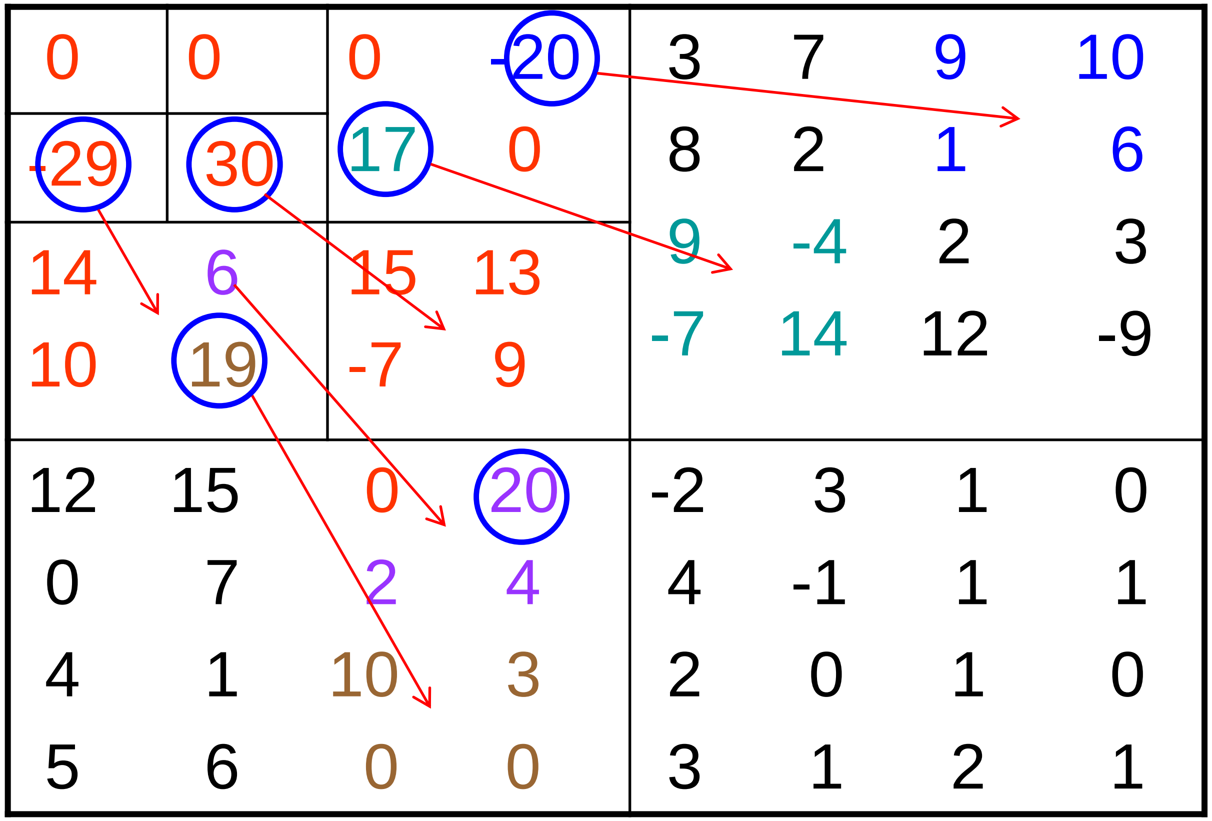
D0: *p n z t p t t p t z t t t t t t t t t t p t t t*

Subordinate List: {57, -37, 39, 33, 33}

➔S0: 10000

0	0	0	-20	3	7	9	10
-29	30	17	0	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	0	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

T1 = 16



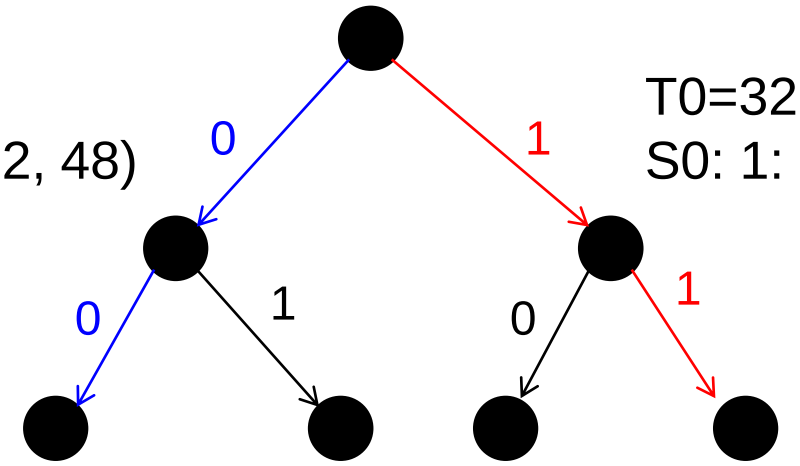
0	0	0	-20	3	7	9	10
-29	30	17	0	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	0	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

T1 = 16

{0, 0, -29, 30, 0, -20, 17, 0, 14, 6, 10, 19, 15, 13. -7, 9
9, 10, 1, 6, 9, -4, -7, 14, 0, 20, 2, 4,10, 3, 0, 0}

D1: z z n p t n p t t z t p t t t t
t t t t t t t t t p t t t t t t

Encoding(T1=16, S1)



T0=32
S0: 0: [32, 48)

T0=32
S0: 1: [48, 64)

T1=16
S1: 0: [32, 40)

T1=16
S1: 0: [48, 56)
1: [56, 64)

{0, 0, -29, 30, 0, -20, 17, 0, 14, 6, 10, 19, 15, 13. -7, 9
9, 10, 1, 6, 9, -4, -7, 14, 0, 20, 2, 4,10, 3, 0, 0}

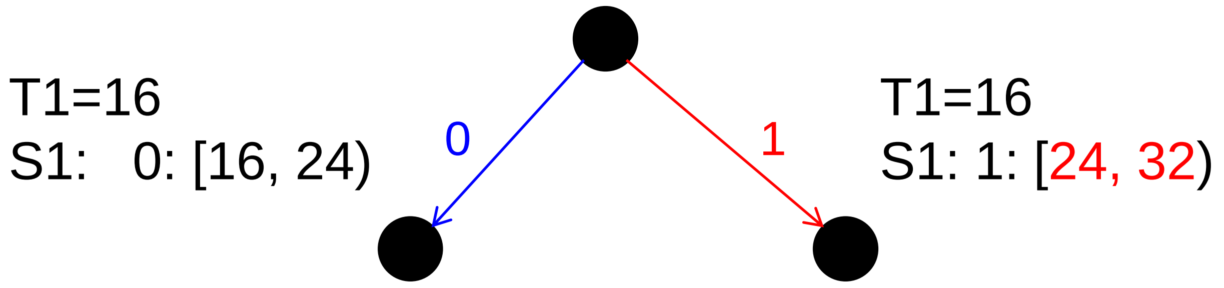
D1: z z n p t n p t t z t p t t t t
t t t t t t t t t p t t t t t t

Subordinate List: {57, -37, 39, 33, 33} ➔S0: 10000

Subordinate List: {57; 37; 39; 33; 33; 29; 30; 20; 17; 19; 20}

➔S1: 10000110000

Encoding(T1=16,S1)



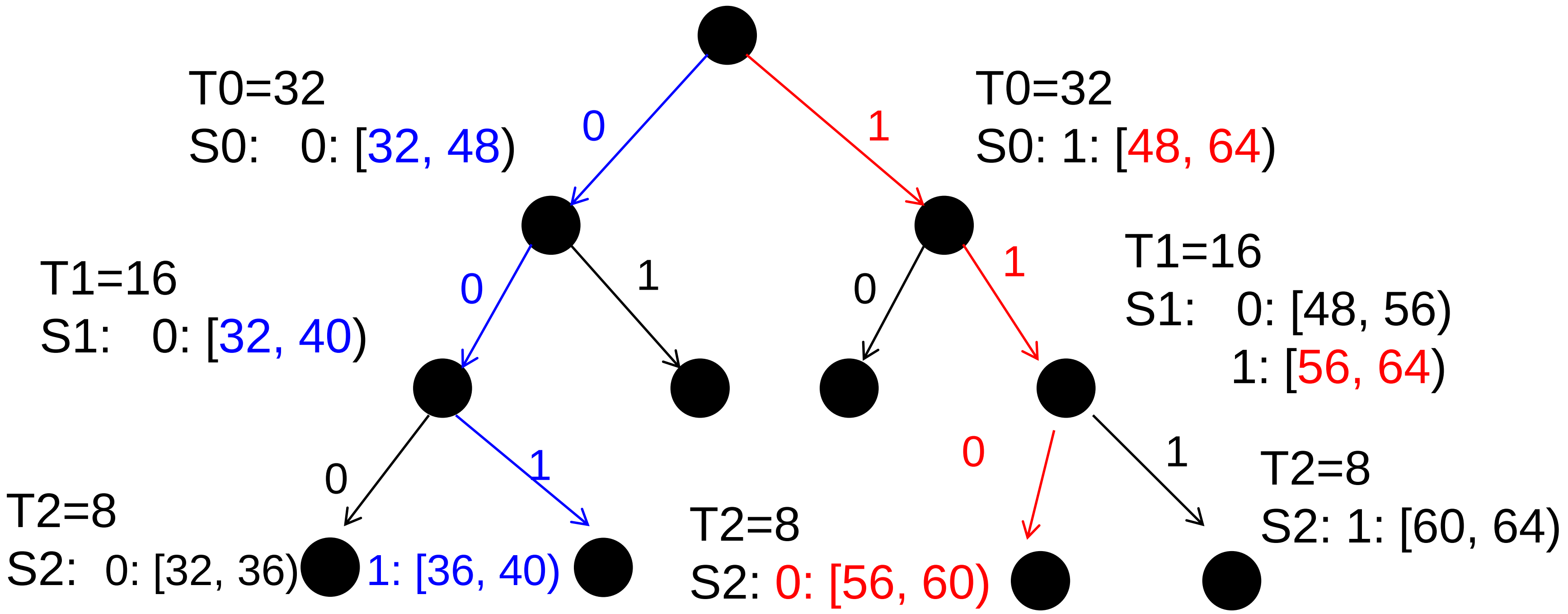
{0, 0, -29, 30, 0, -20, 17, 0, 14, 6, 10, 19, 15, 13. -7, 9
9, 10, 1, 6, 9, -4, -7, 14, 0, 20, 2, 4,10, 3, 0, 0}

D1: z z n p t n p t t z t p t t t t
t t t t t t t t t p t t t t t t

Subordinate List: {57; 37; 39; 33; 33; 29; 30; 20; 17; 19; 20}

➔S1: 10000110000

Encoding(T2=8,S2)



Subordinate List: {57, -37, 39, 33, 33} ➔S0: 10000

Subordinate List: {57; 37; 39; 33; 33; 29; 30; 20; 17; 19; 20}

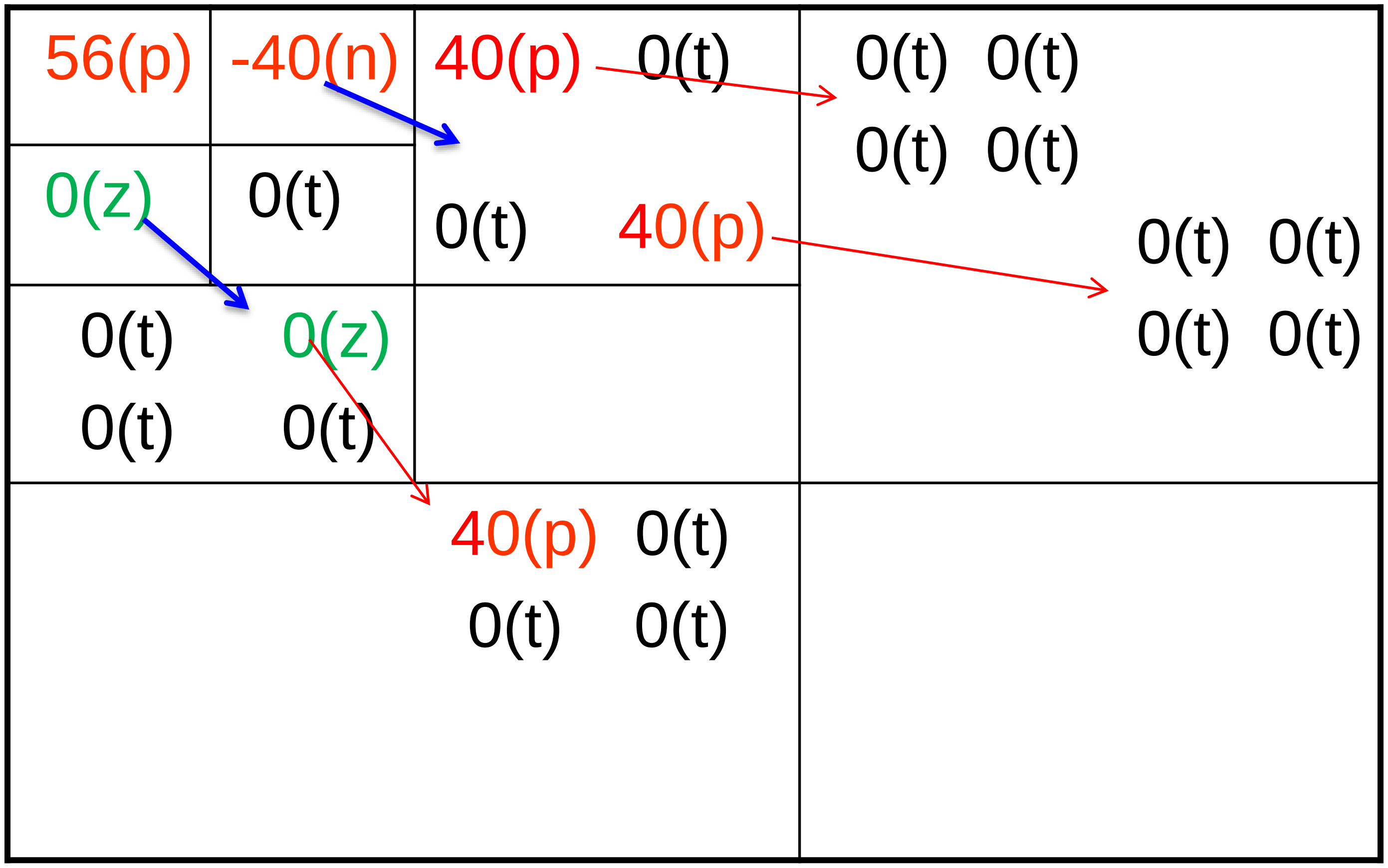
➔S1: 10000110000

Subordinate List: {57; 37; 39; 33; 33; 29; 30; 20; 17; 19; 20;…}

➔S2: 011001110011011000000110110

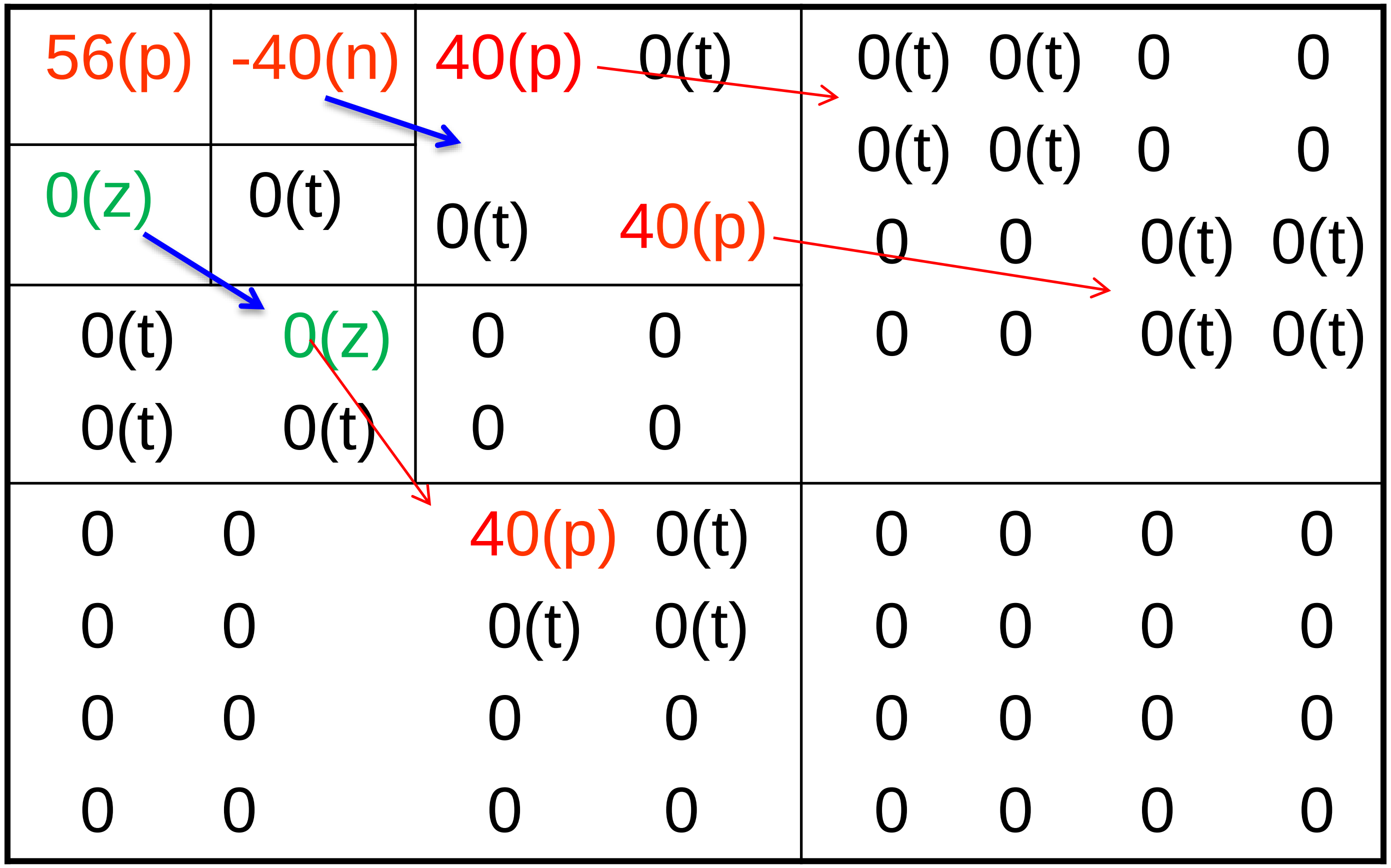
Decoding

- D0, S0 ➔ 0: [32, 48) ➔ 40
- 1: [48, 64) ➔ 56
- D0, S0, D1, S1 ➔ 11: [56, 64) ➔ 60
- 00: [32, 40) ➔ 36
- ➔ 1: [24, 32) ➔ 28
- 0: [16, 24) ➔ 20
- D0,S0,D1,S1,D2,S2
- ➔ 110: [56, 60) ➔ 58
- 001: [36, 40) ➔ 38



D0: *p n z t p t t p t z t t t t t t t t t t pt t t*

S0: 10000



56(p)	-40(n)	40(p)	0(t)	0(t)	0(t)	0	0
0(z)	0(t)	0(t)	40(p)	0(t)	0(t)	0	0
0(t)	0(z)	0	0	0	0	0(t)	0(t)
0(t)	0(t)	0	0	0	0	0(t)	0(t)
0	0	40(p)	0(t)	0	0	0	0
0	0	0(t)	0(t)	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

D0: *p n z t p t t p t z t t t t t t t t t t pt t t*

S0: 10000

60(z)	-36(z)	36(t)	0(n)	0	0	0(t)	0(t)
0(n)	0(p)	0(p)	36(t)	0	0	0(t)	0(t)
0(t)	0(z)	0(t)	0(t)	0(t)	0(t)	0	0
0(t)	0(p)	0(t)	0(t)	0(t)	0(t)	0	0
0	0	36(t)	0(p)	0	0	0	0
0	0	0(t)	0(t)	0	0	0	0
0	0	0(t)	0(t)	0	0	0	0
0	0	0(t)	0(t)	0	0	0	0

D0: *p n z t p t t p t z t t t t t t t t t t pt t t*

S0: 10000

D1: *zznp tnpt tztp tttt tttt tttt tptt tttt*

S1: 10000110000

60(z)	-36(z)	36(t)	-20(n)	0	0	0(t)	0(t)
-28(n)	28(p)	20(p)	36(t)	0	0	0(t)	0(t)
0(t)	0(z)	0(t)	0(t)	0(t)	0(t)	0	0
0(t)	20(p)	0(t)	0(t)	0(t)	0(t)	0	0
0	0	36(t)	20(p)	0	0	0	0
0	0	0(t)	0(t)	0	0	0	0
0	0	0(t)	0(t)	0	0	0	0
0	0	0(t)	0(t)	0	0	0	0

D0: *p n z t p t t p t z t t t t t t t t t t pt t t*

S0: 10000

D1: *zznp tnpt tztp tttt tttt tttt tptt tttt*

S1: 10000$\boxed{110000}$

58	-38	38	-22	0	0	12	12
-30	30	18	34	12	0	0	0
12	0	12	12	12	0	0	0
12	20	0	12	0	12	12	-12
12	12	34	22	0	0	0	0
0	0	0	0	0	0	0	0
0	0	12	0	0	0	0	0
0	0	0	0	0	0	0	0

Reconstructed transform coefficients from D0, S0, D1, S1, D2, and the first 10 bits of S2

EZW – An example(1)

63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

Wavelet coefficients for a 8*8 block

EZW – An example(2)

The initial threshold is 32 and the result from the dominant_pass is shown in the figure

63 p	-34 n	49 p	10 t	7 t	13 t	-12	7
-31 z	23 t	14 t	-13 t	3 t	4 t	6	-1
15 t	14 z	3	-12	5	-7	3	9
-9 t	-7 t	-14	8	4	-2	3	2
--5	9	-1 t	47 p	4	6	-2	2
3	0	-3 t	2 t	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

Data without any symbol is a node in the zero-tree.

EZW – An example(3)

The result from the dominant_pass is output as the following:

p, n*, z, t, p, t, t, t, t, z, t, t, t, t, t, t, t, p,* ***t****, t*

p—01, n—11, t—00, z--10

The significant coefficients are put in a subordinate list and are refined. A one-bit symbol is output to the decoder.

Original data	63	34	49	47
Output symbol	1	0	1	0
Reconstructed data	56	40	56	40

For example, the output for 63 is:

sign	32	16	8	4	2	1
0	1	1	?	?	?	?

If T+.5T is less than data item take the average of 2T and 1.5T. So 63 will be reconstructed as the average of 48 and 64 which is **56**. If it is more, put a 0 in the code and encode this as t+.5T+.25T. Thus, 34 is reconstructed as **40**.

EZW – An example(4)

*	*	*	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
--5	9	-1	*	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

After dominant_pass, the significant coefficients will be replaced by * or 0
Then the threshold is divided by 2, so we have **16** as current threshold

EZW – An example(5)

The result from the second dominant_pass is output as the following:

z, t, n, p, t, t, t, t, t, t, t, t

The significant coefficients are put in the subordinate list and all data in this list will be refined as:

Original data	63	34	49	47	31	23
Output symbol	1	0	0	1	1	0
Reconstructed data	60	36	52	44	28	20

For example, the output for 63 is:

sign	32	16	8	4	2	1
0	1	1	1	?	?	?

The computatin is now extended with respect to the next significant bit. So 63 will be reconstructed as the average of 56 and 64 –- **60**!

EZW – An example(6)

The process is going on until threshold =1, the final output as:

D1: pnztptttztttttttptt
S1: 1010
D2: ztnptttttttt
S2: 100110
D3: zzzzzppnppnttnnptpttnttttttttpttptttttttttpttttttttttt
S3: 10011101111011011000
D4: zzzzzzztztznzzzzpttptpptpnptntttttptpnpppptttttptptttpnp
S4: 11011111011001000001110110100010010101100
D5: zzzzztzzzzztpzzzttpttttnptppttptttnppnttttpnnpttpttppttt
S5:
10111100110100010111110101101100100000000110110110011000111
D6: zzzttztttztttttnnttt

For example, the output for 63 is:

sign	32	16	8	4	2	1
0	1	1	1	1	1	1

So 63 will be reconstructed as 32+16+8+4+2+1=63!
Note, how progressive transmission can be done.