

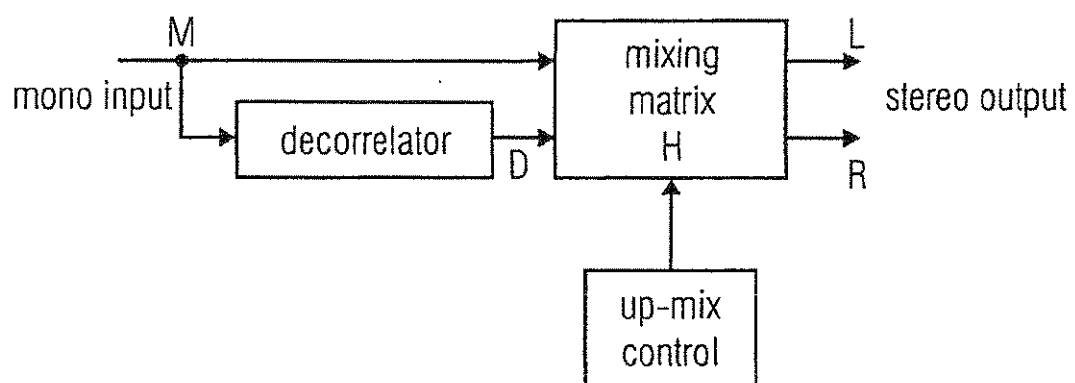


FIG 1

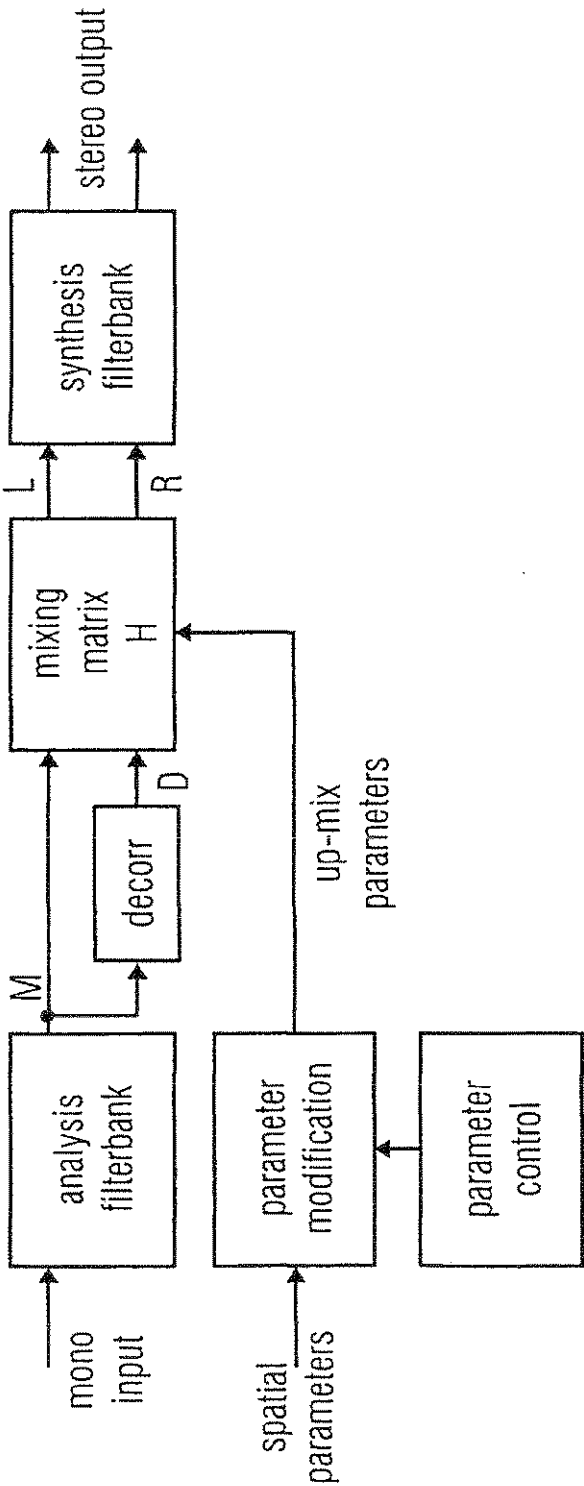


FIG 2

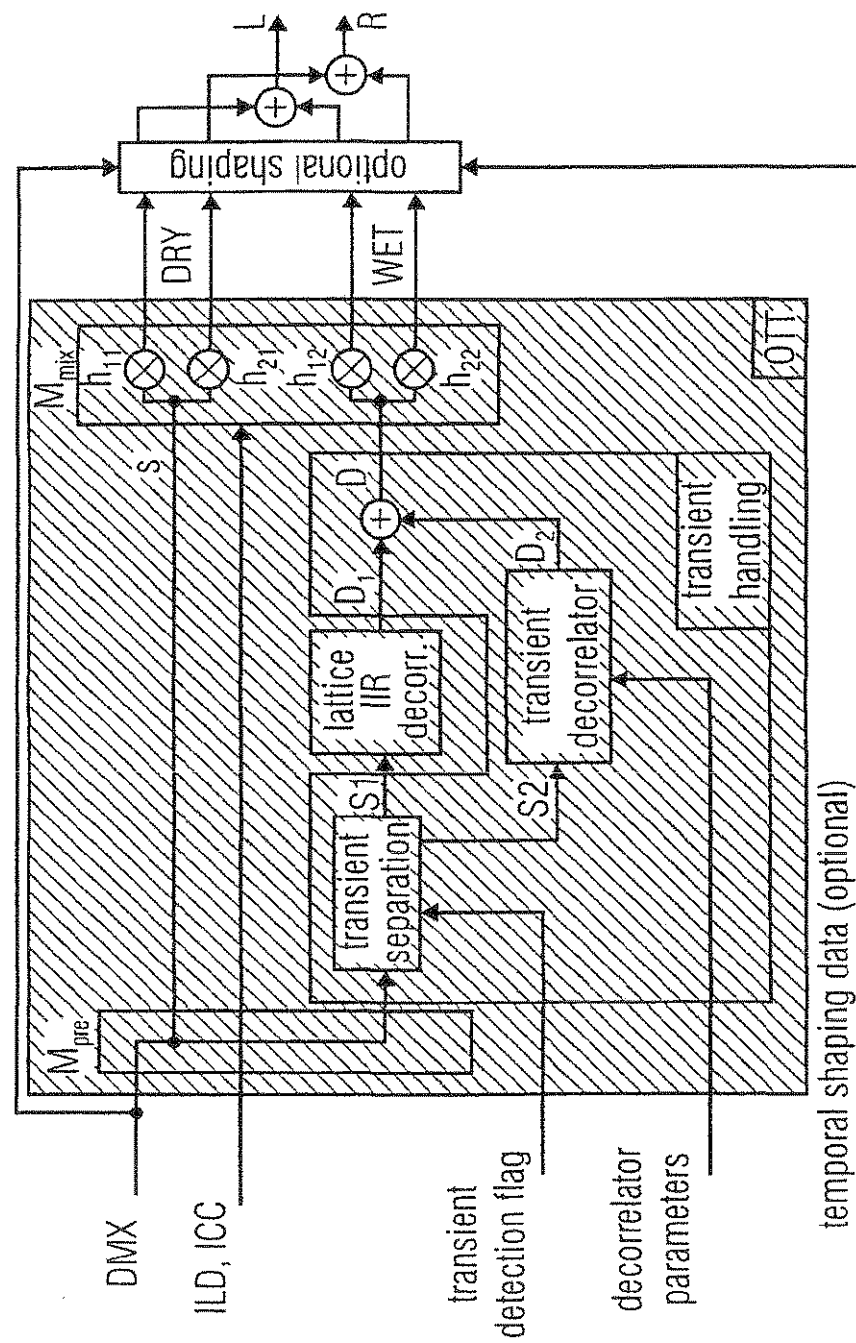


FIG 3

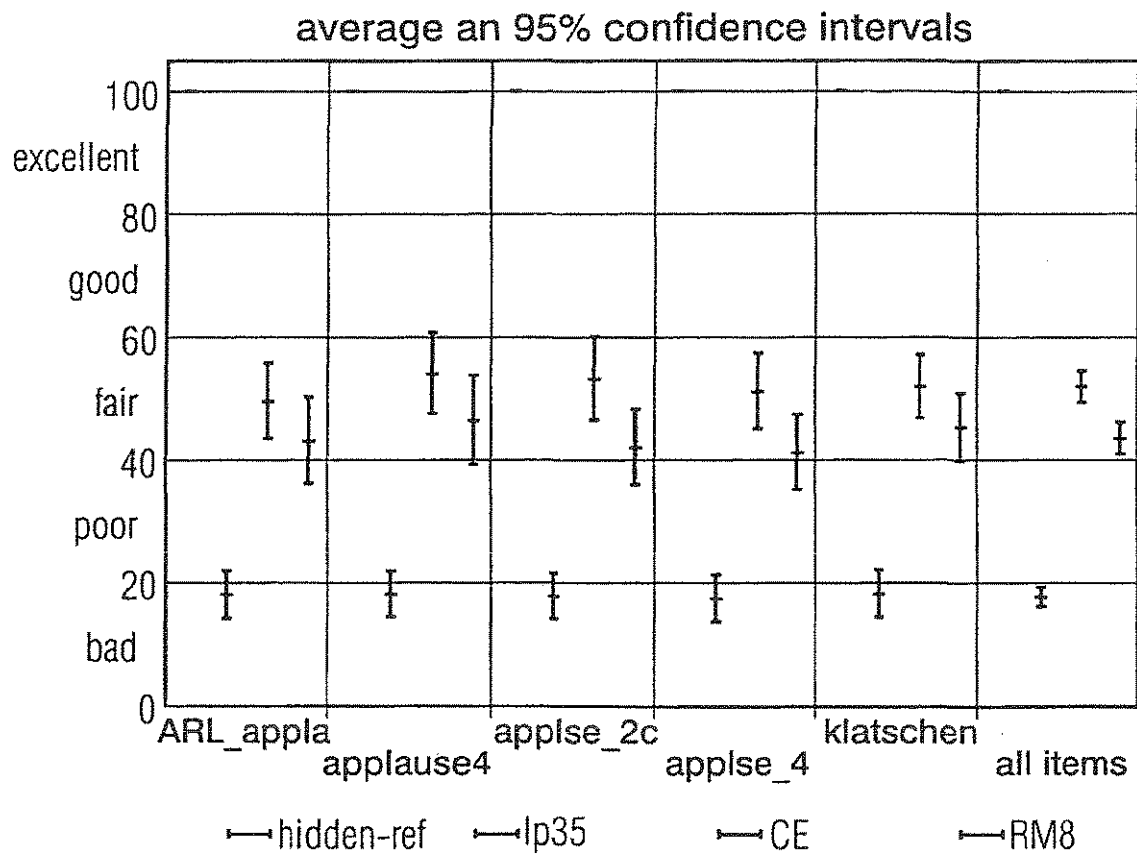


FIG 4

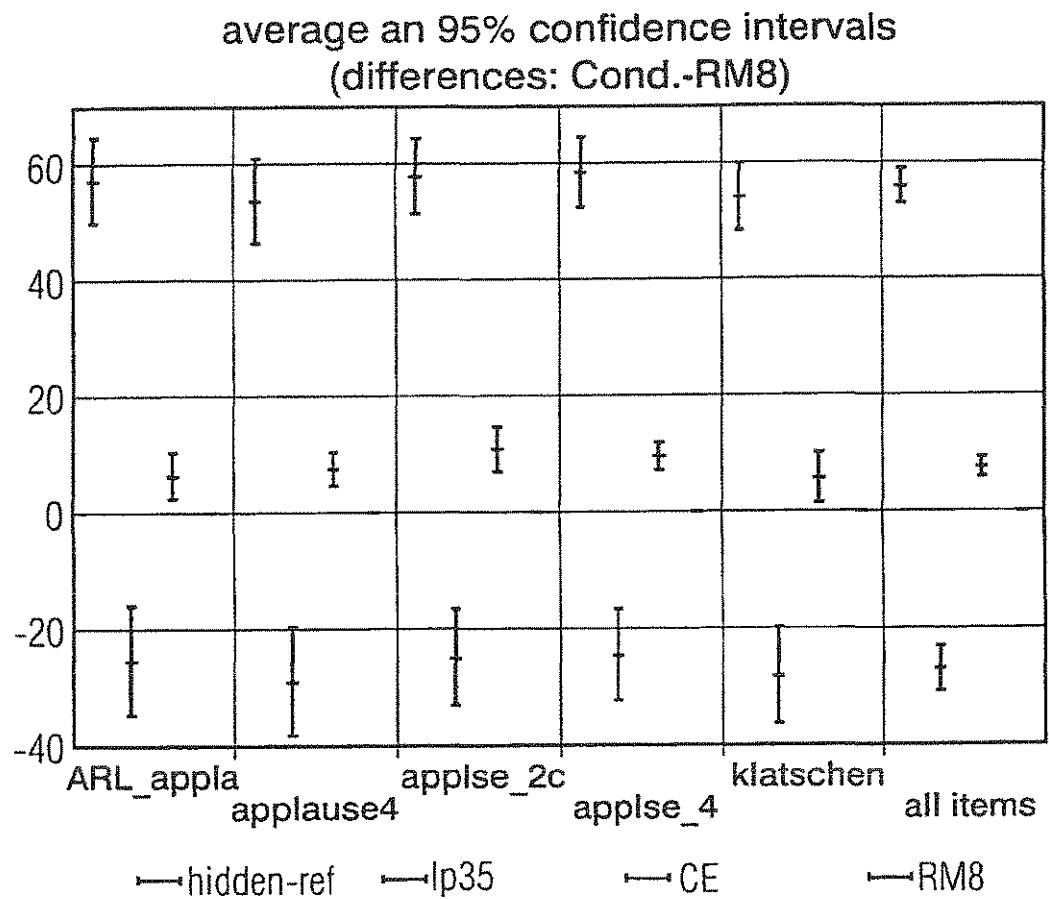


FIG 5

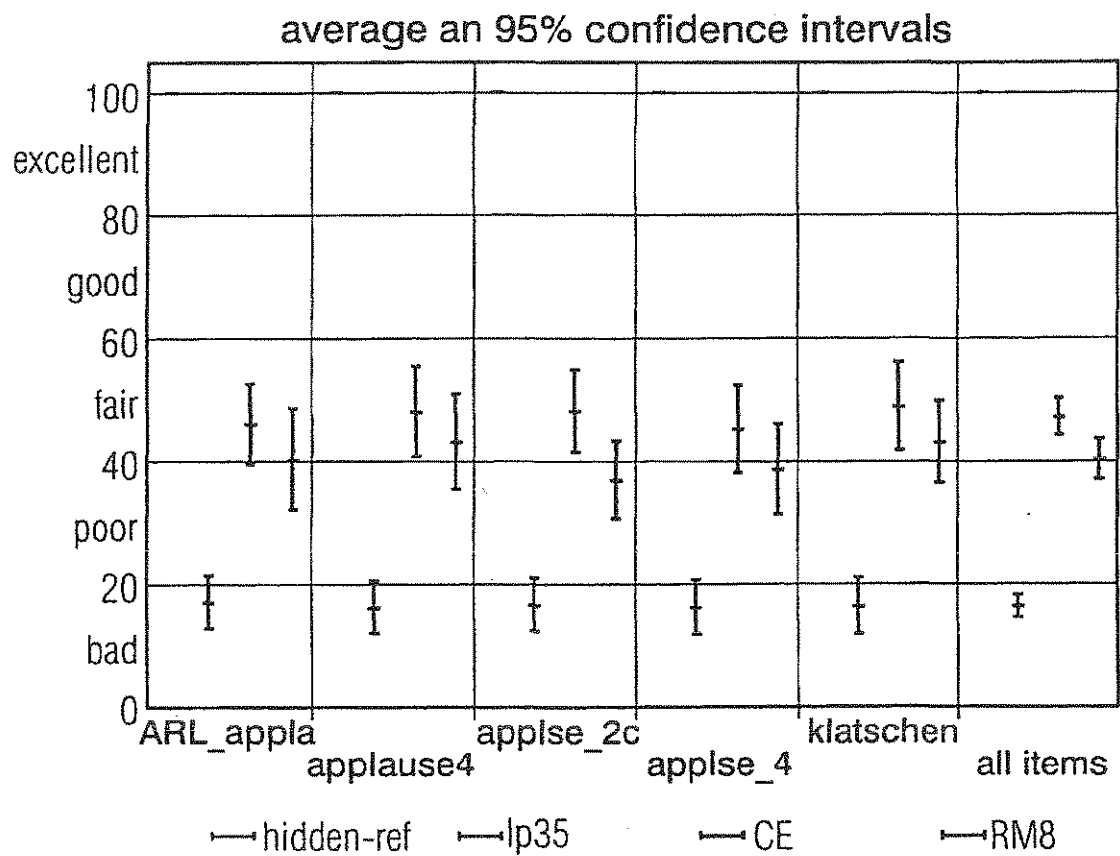


FIG 6

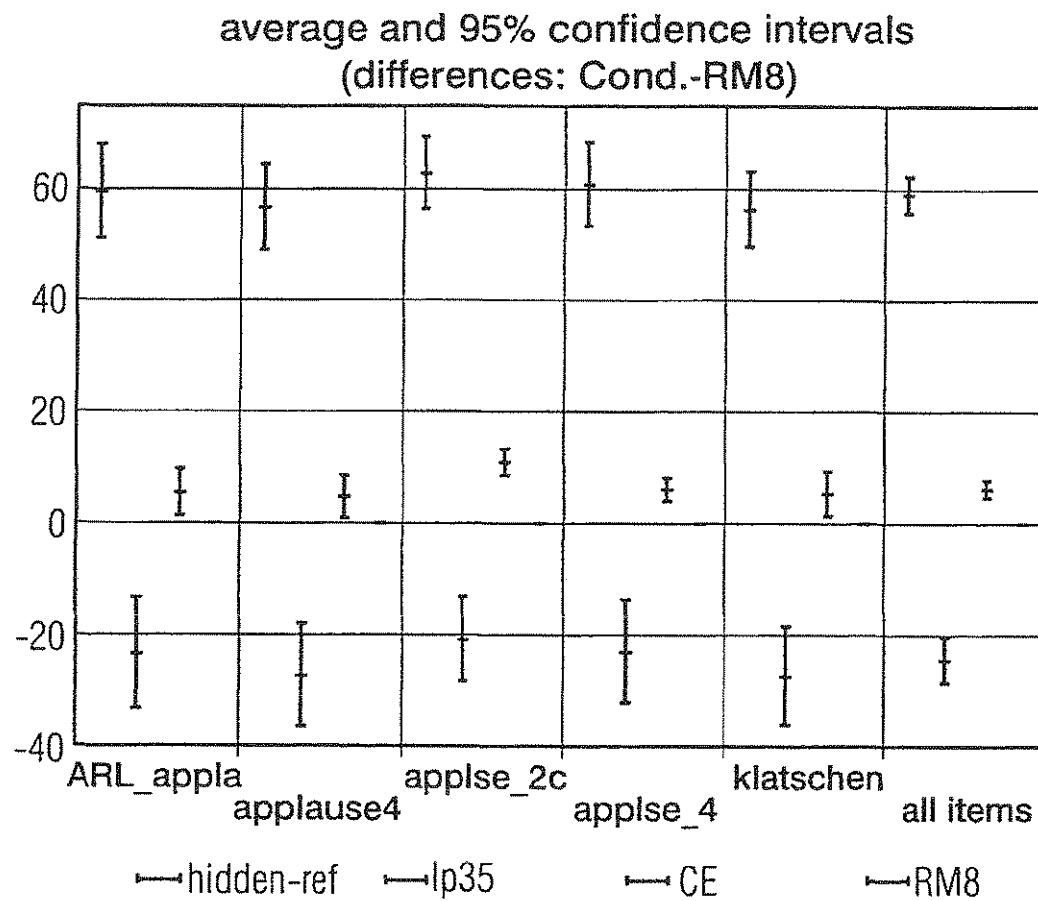


FIG 7

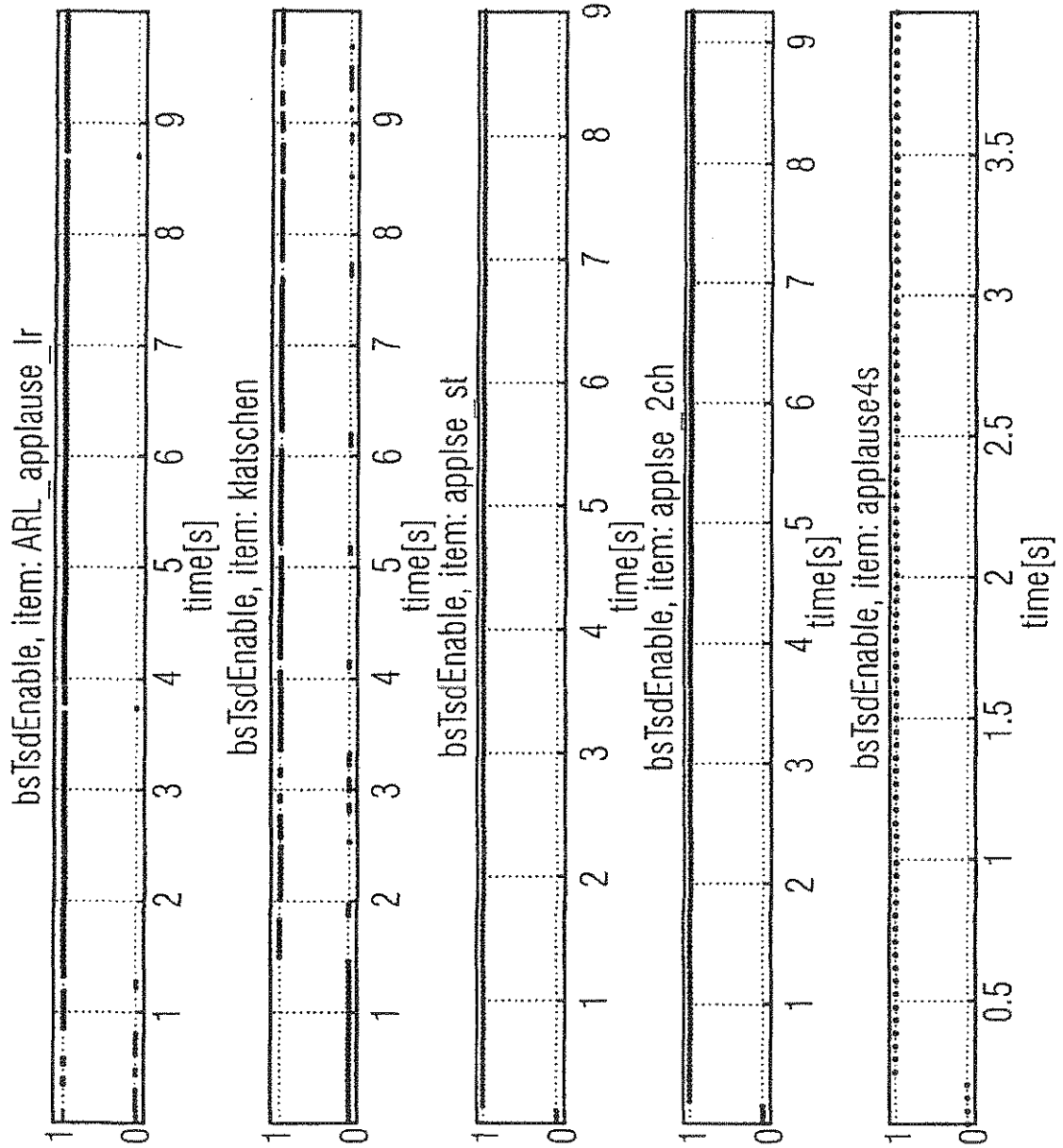


FIG 8

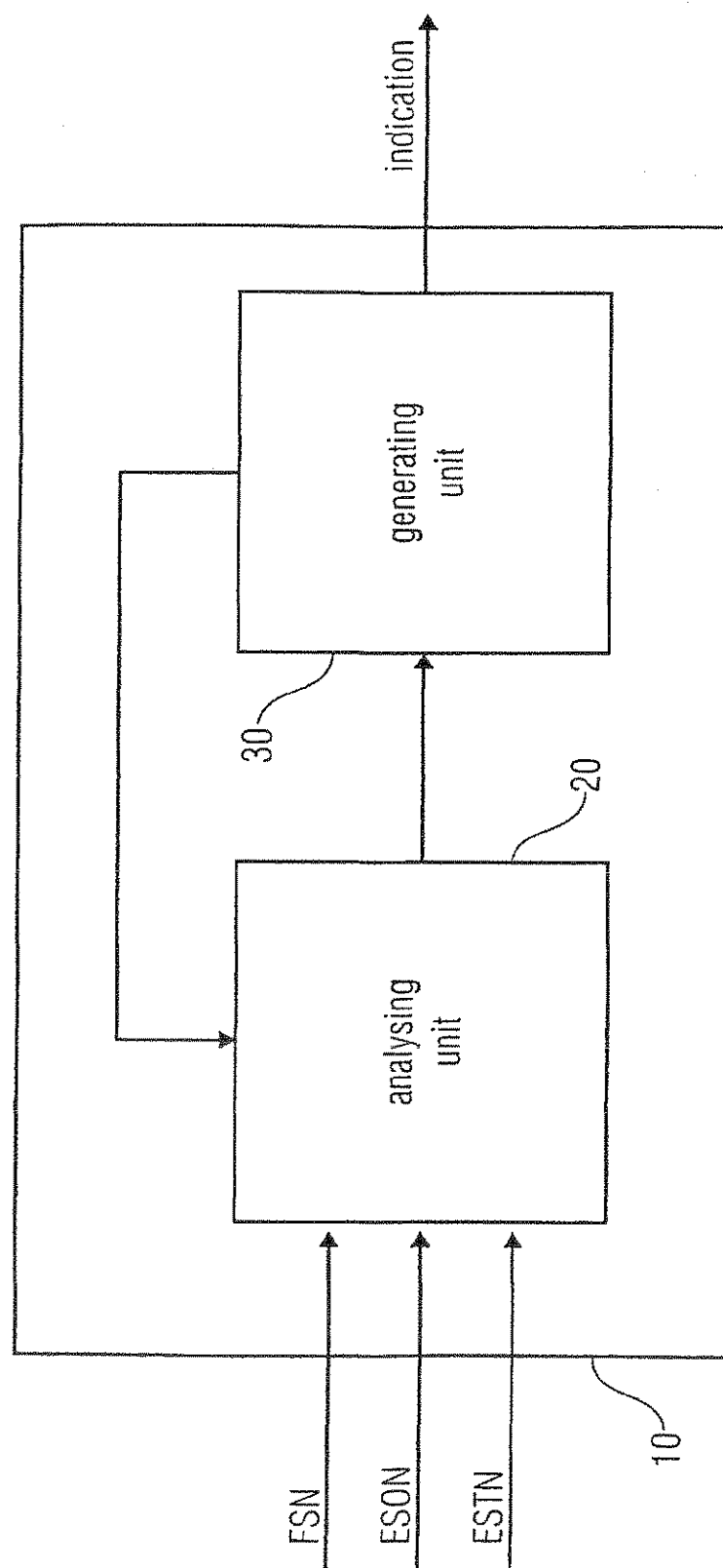


FIG 9A

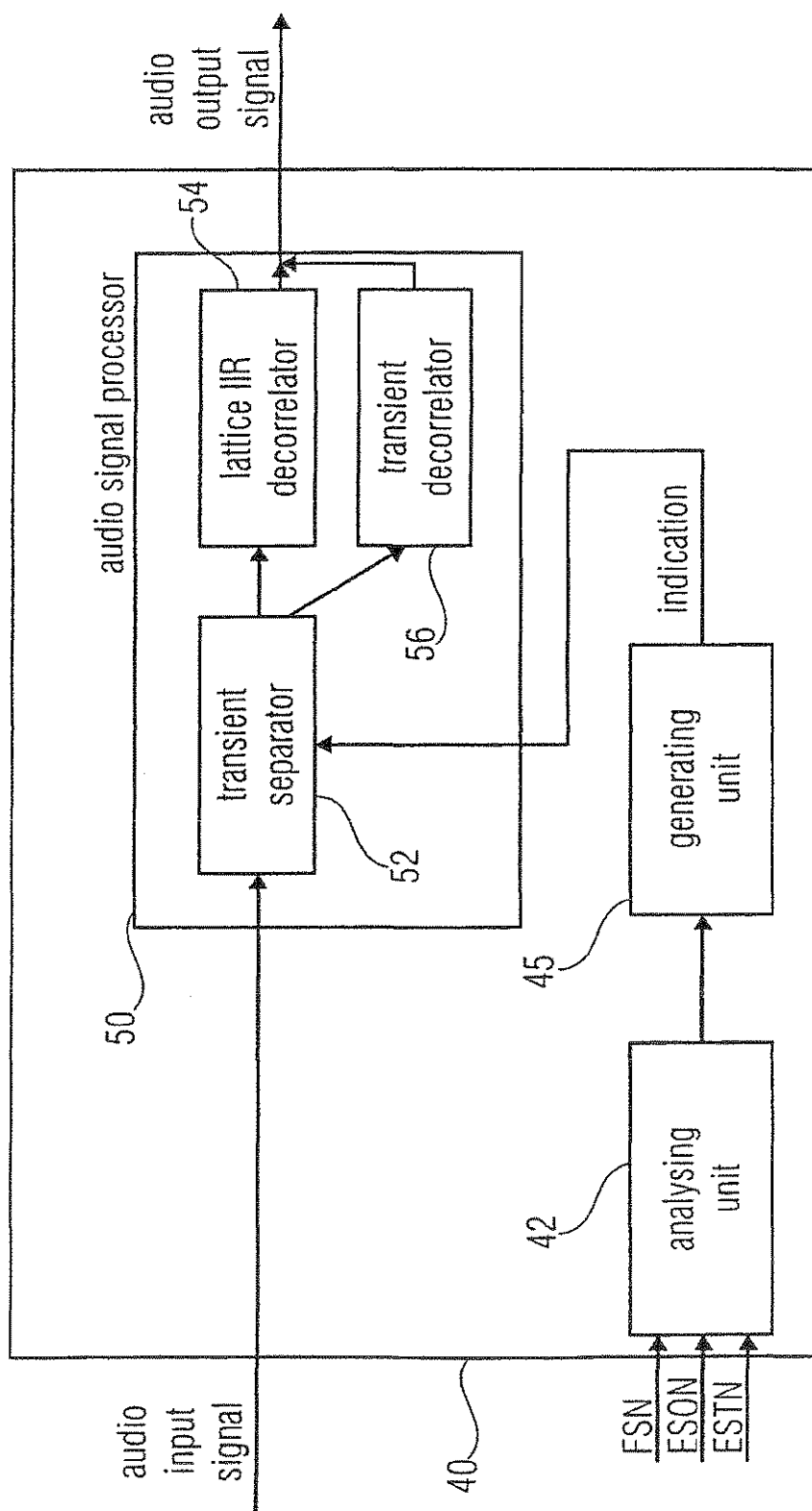


FIG 9B

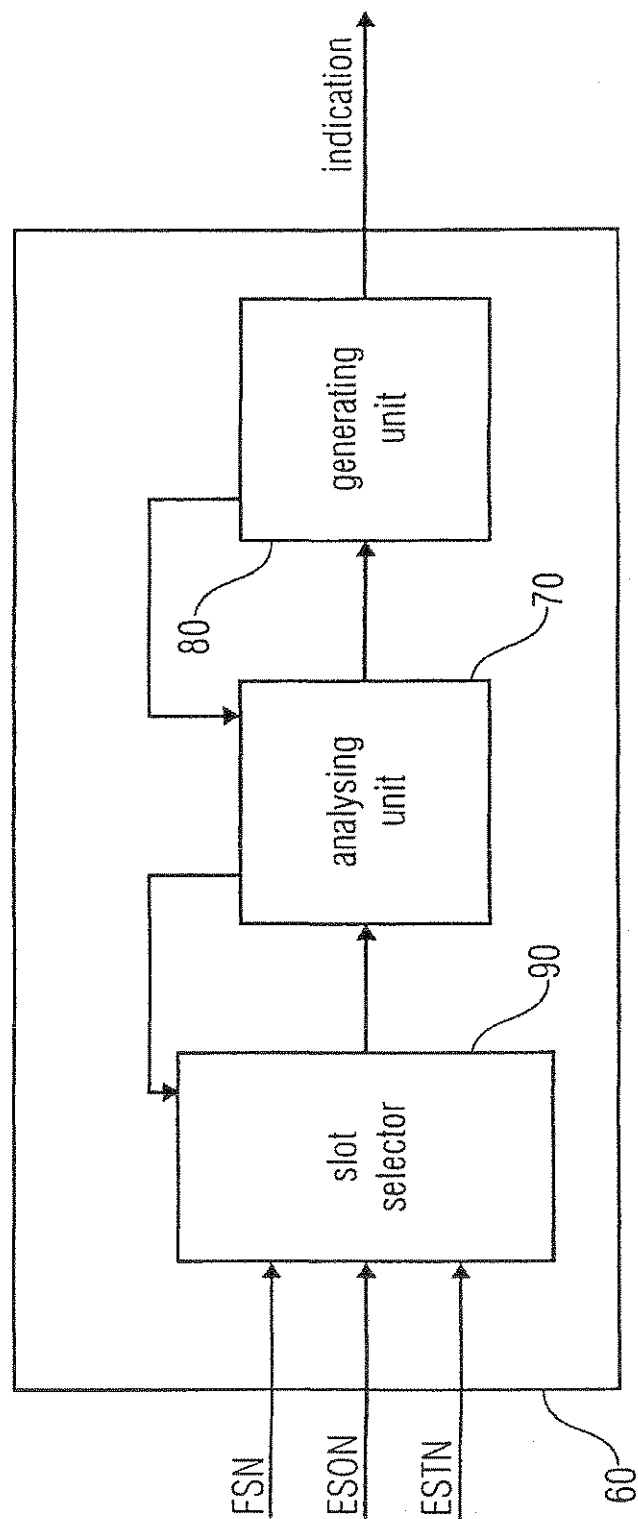


FIG 9C

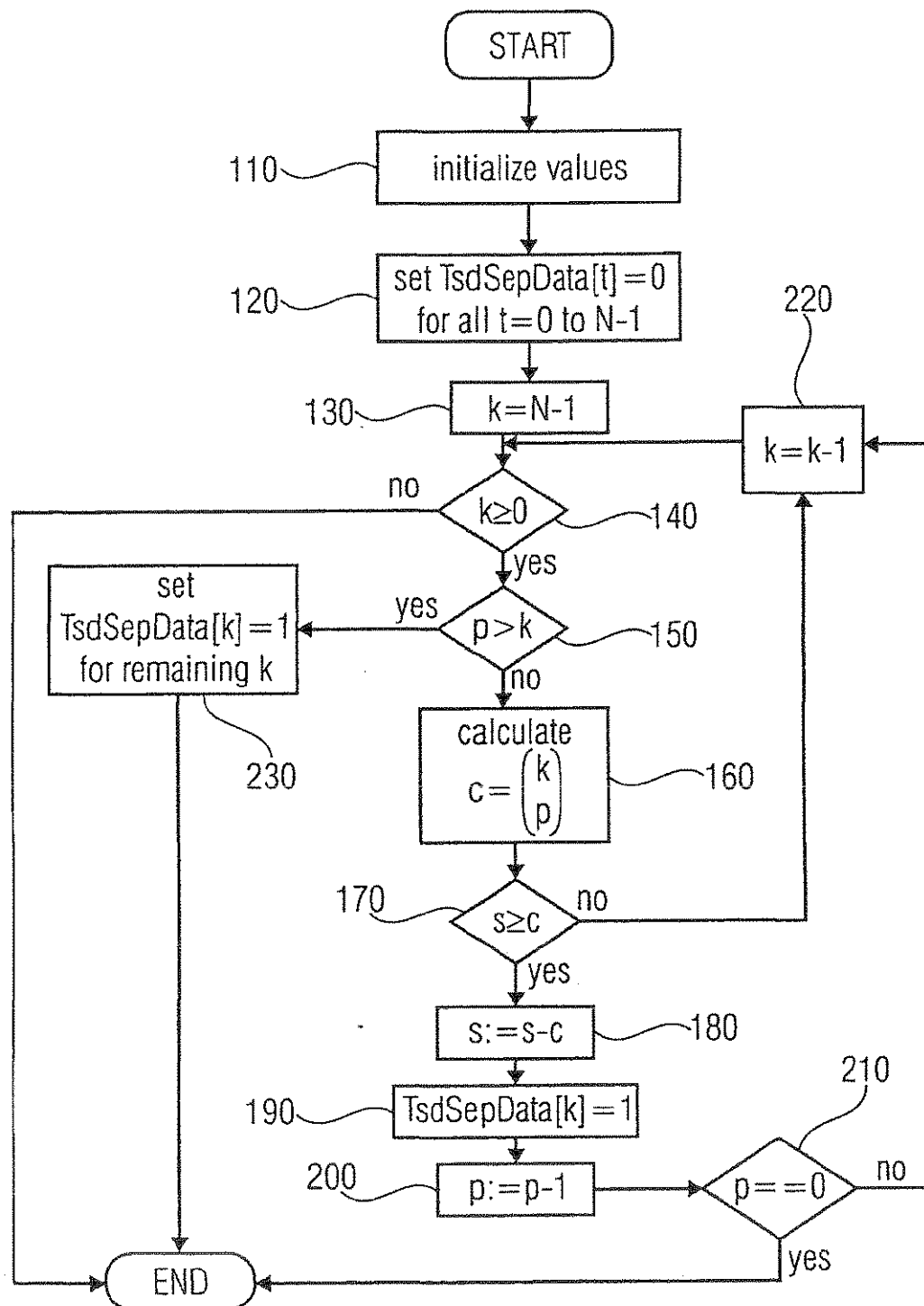


FIG 10

```
s = bsTsdCodedPos;
p = bsTsdNumTrSlots + 1;
N = numSlots

for (k=0; k<N; k++)
{
    TsdSepData[k]=0;
}

for (k=N-1; k>=0; k--)
{
    if (p > k) {
        for (;k>=0; k--)
            TsdSepData[k]=1;
        break;
    }
    c = k-p+1;
    for (h=2; h<=p; h++) {
        c *= k - p + h;
        c /= h;
    }
    if (s >= (int)c) { /* c is long long for up to 32 slots */
        s -= c;
        TsdSepData[k]=1;
        p--;
        if (p == 0)
            break;
    }
}
```

FIG 11

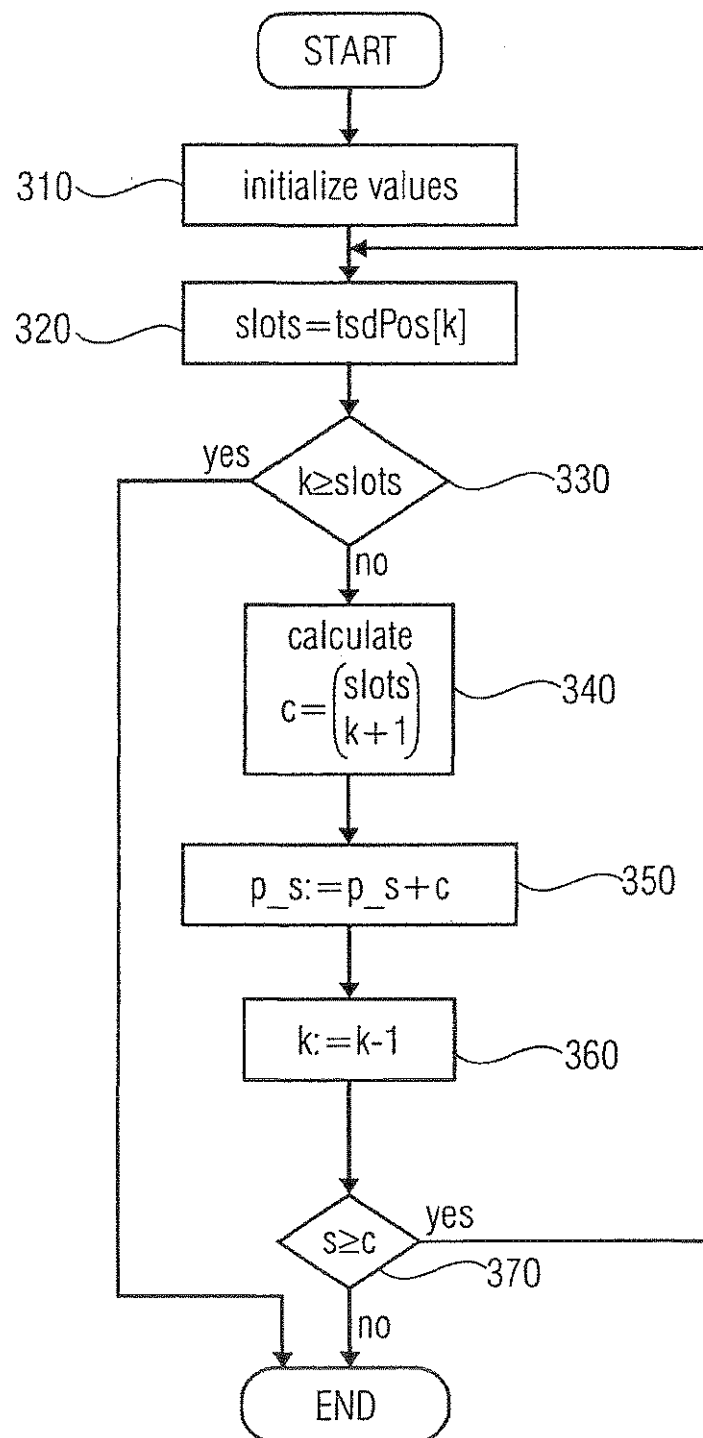


FIG 12

```

p_s = 0;
for (k=tsdPosLen-1; k>=0; k--){
    slots = tsdPos[k];
    if (k >= slots) {
        break;
    }
    c = 1; /* c is long long for up to 32 slots */
    for (h=1; h<=(k+1); h++) {
        c *= slots-k-1 + h;
        c /= h;
    }
    p_s += c;
}
bsTsdCodedPos = p_s;

```

FIG 13

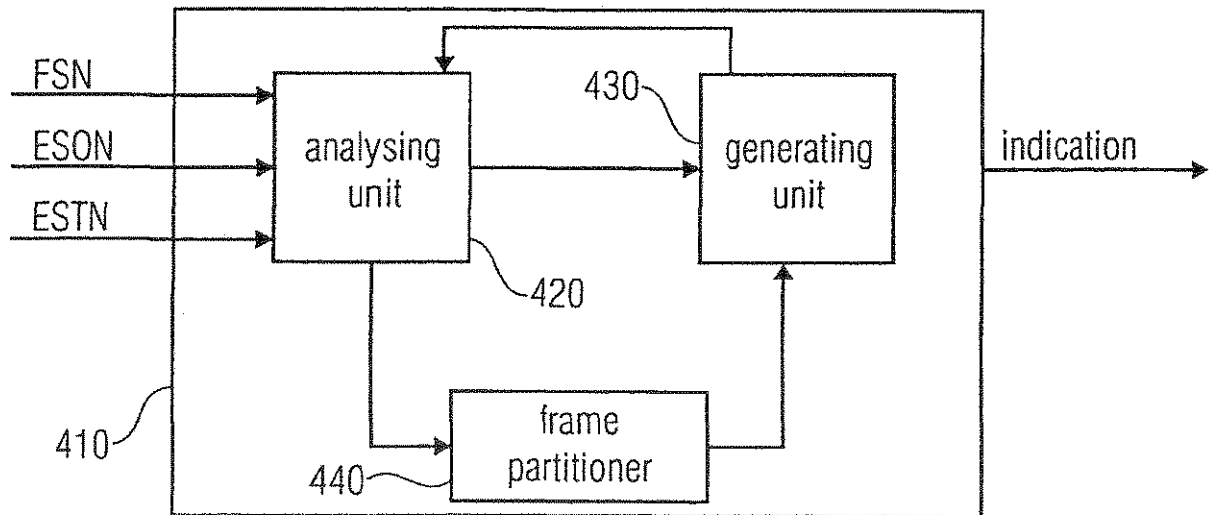


FIG 14

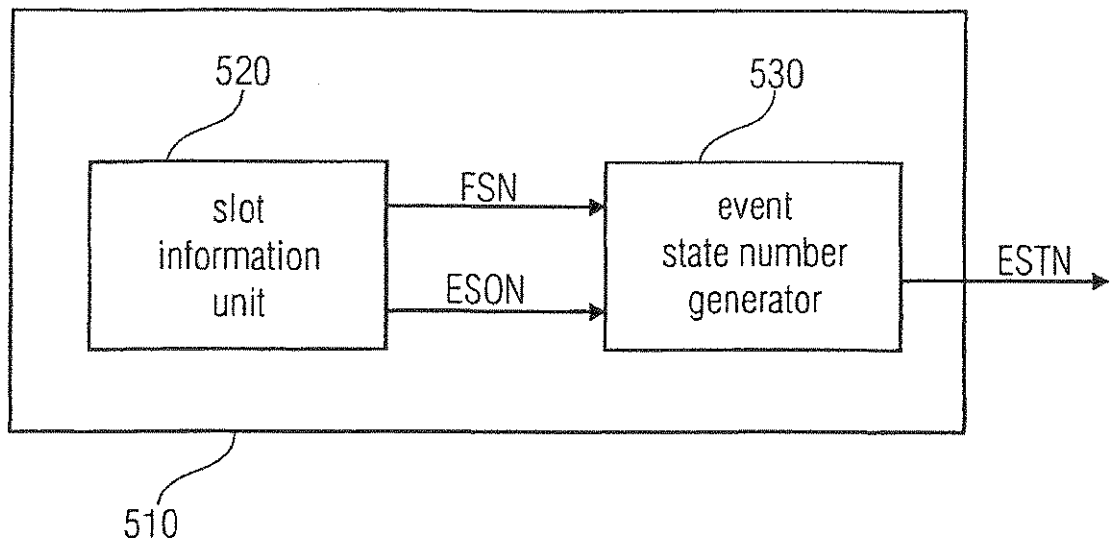


FIG 15

syntax	No. of bits	mnemonic
<pre> Mps212Data(indepFlag) { FrameingInfo(); if (indepFlag) { bsIndependencyFlag=1; } else { bsIndependencyFlag; } OttData(); SmgData(); TempShapeData(); if (bsTsdEnable==1) { TsdData(); } } </pre>	1	uimsbf

FIG 16

syntax	No. of bits	mnemonic
TsdData() { bsTsdNumTrSlots; TsdSepData=TsdTrPos_dec(bsTsdCodedPos); for (ts=0; ts<numSlots; ts++) { if (TsdSepData[ts]==1) { bsTsdTrPhaseData[ts] } else { bsTsdTrPhaseData[ts]=0; } } }		
	nBitsTrSlots	uimbsf note 1
	nBitsTsdCW	vlclbf note 2, 3
	3	uimbsf
note 1: nBitsTrSlots depends on the frame length as defined in Table 98. note 2: nBitTsdCW is calculated according to the rule described in 7.11.2.4. note 3: TsdTrPos_dec() is defined in 7.11.2.4.		

FIG 17

numSlots	nBitsTrSlots
$\text{numSlots} \leq 6$	1
$6 < \text{numSlots} \leq 12$	2
$12 < \text{numSlots} \leq 24$	3
$24 < \text{numSlots} \leq 48$	4
$48 < \text{numSlots}$	5

FIG 18

bsTempShapeConfig	Meaning
0	do not apply temporal shaping
1	apply STP
2	apply GES
3	apply TSD

FIG 19

syntax	No. of bits	mnemonic
TempShapeData() { bsTsdEnable = 0; if (bsTempShapeConfig == 3) { bsTsdEnable; } else if ((bsTempShapeConfig == 1) (bsTempShapeConfig == 2)) { bsTempShapeEnable; if (bsTempShapeEnable) { for (ch=0; ch<numTempShapeChan; ch++) { bsTempShapeEnableChannel[ch]; } if (bsTempShapeConfig == 2) { EnvelopeReshapeHuff(bsTempShapeEnableChannel); } } } }		
	1	uimbsf
	1	uimbsf
		note 1
	1	uimbsf
note 1: numTempShapeChan is 2 as defined 6.2.13.2		

FIG 20

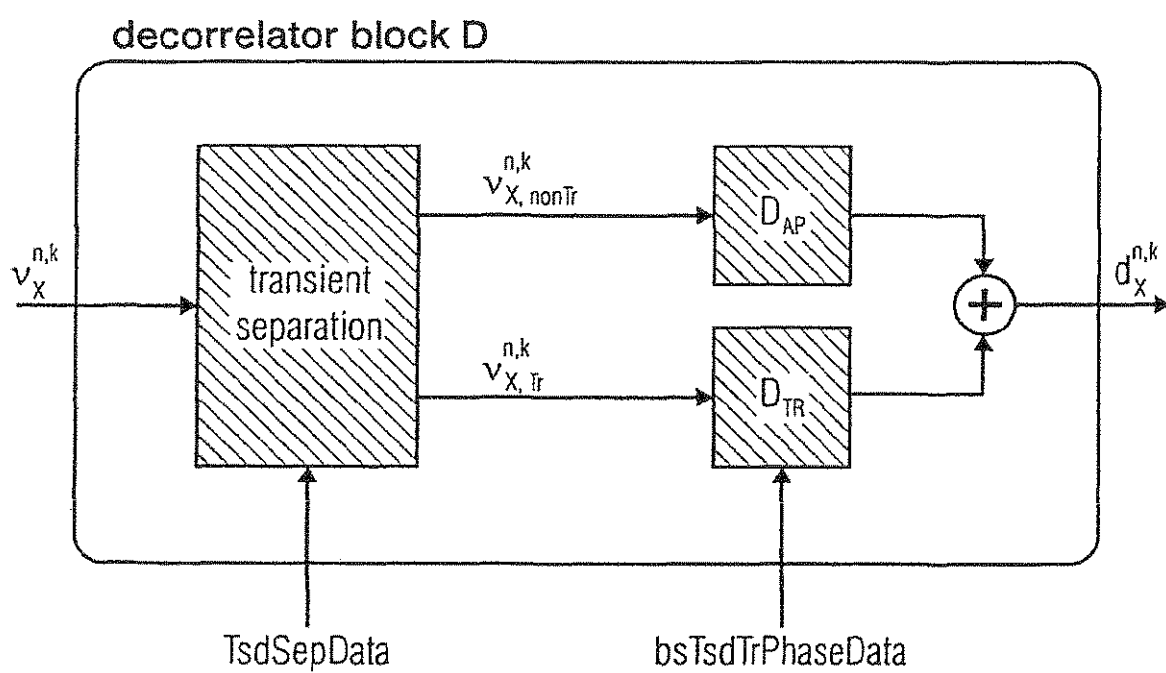


FIG 21

syntax	No. of bits	mnemonic
EcData(dataType, paramIdx, startBand, stopBand)		note 1
{		
dataSets = 0;		
for (ps=0; ps < numParamSets; ps++) {		
bsXXXdataMode[paramIdx][ps];	2	note 2 uimsbf
if (bsXXXdataMode[paramIdx][ps] == 3) {		
dataSets++;		
}		
}		
setIdx = 0;		
while (setIdx < dataSets) {		
bsDataPairXXX[paramIdx][setIdx];	1	uimsbf
bsQuantCoarseXXX[paramIdx][setIdx];	1	uimsbf
bsFreqResStrideXXX[paramIdx][setIdx];	2	uimsbf
dataBands = (stopBand-startBand-1)/pbStride+1;		note 3
EcDataPair(dataType, paramIdx, setIdx, dataBands,		
bsDataPairXXX[paramIdx][setIdx],		
bsQuantCoarseXXX[paramIdx][setIdx]);		
if (bsDataPairXXX[paramIdx][setIdx]) {		
bsQuantCoarseXXX[paramIdx][setIdx+1] = bsQuantCoarseXXX[paramIdx][setIdx];		
bsFreqResStrideXXX[paramIdx][setIdx+1] = bsFreqResStrideXXX[paramIdx][setIdx];		
}		
setIdx += bsDataPairXXX[paramIdx][setIdx]+1;		
}		
startBandXXX[paramIdx] = startBand;		
stopBandXXX[paramIdx] = stopBand;		
}		
note 1: XXX is to be replaced by the value of dataType (CLD, ICC, IPD). note 2: numParamSets is defined by numParamSets = bsNumParamSets + 1. note 3: pbStride is defined in ISO/IEC 23003-1:2007, Table 70 and depends on bsFreqResStride[]. Furthermore the division shall be interpreted as ANSI C integer division.		

FIG 22

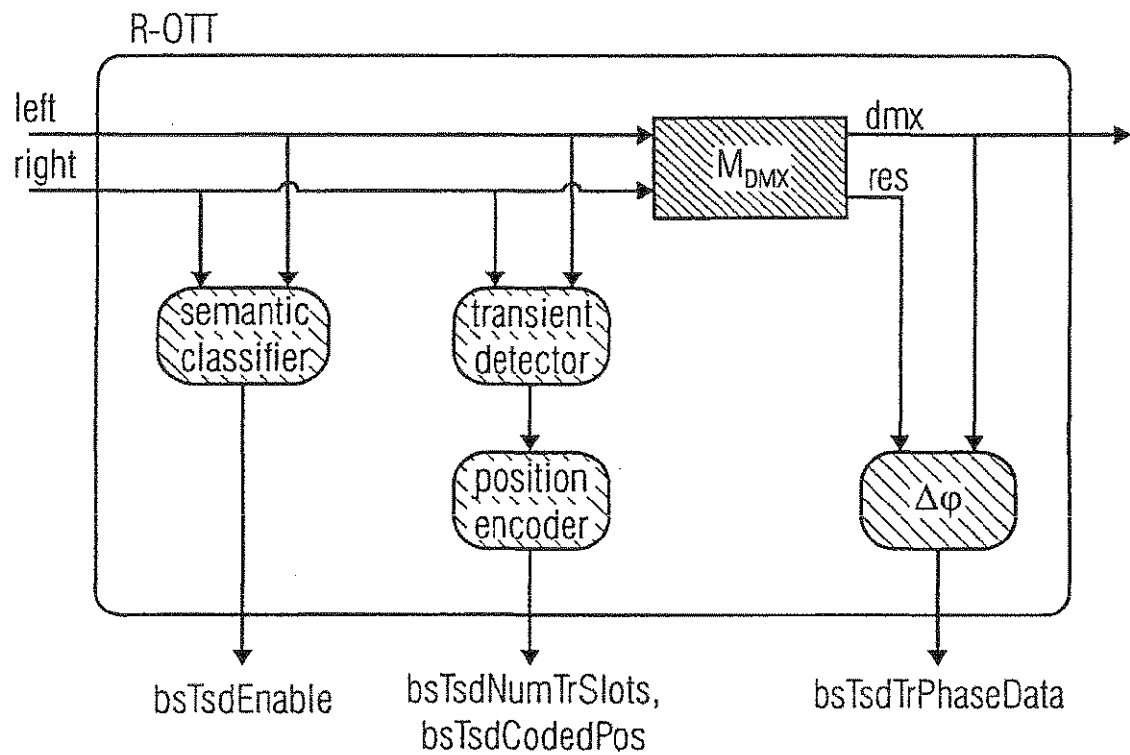


FIG 23