

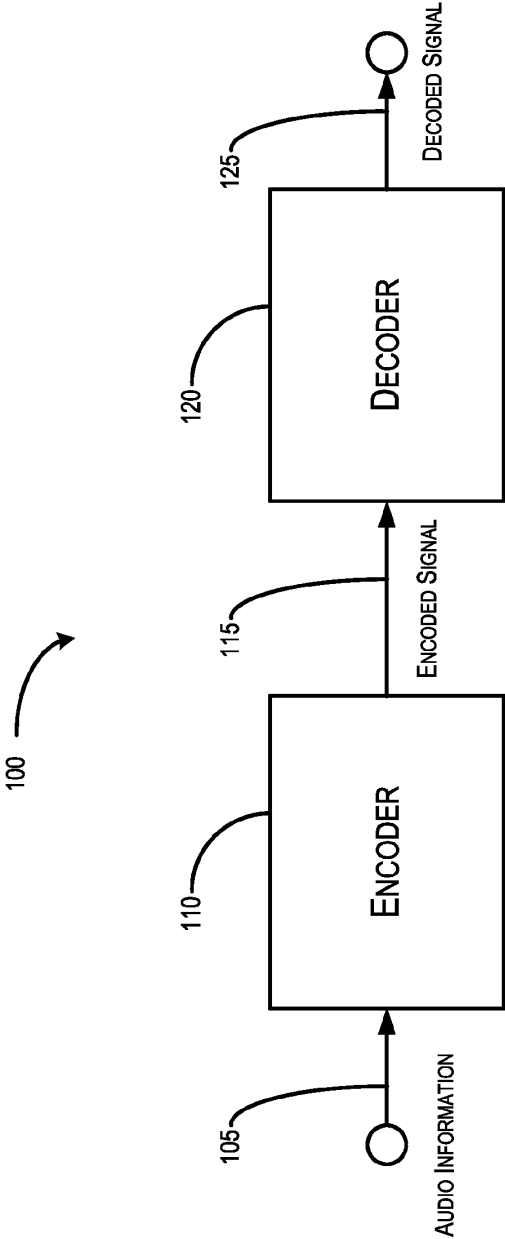


FIG. 1

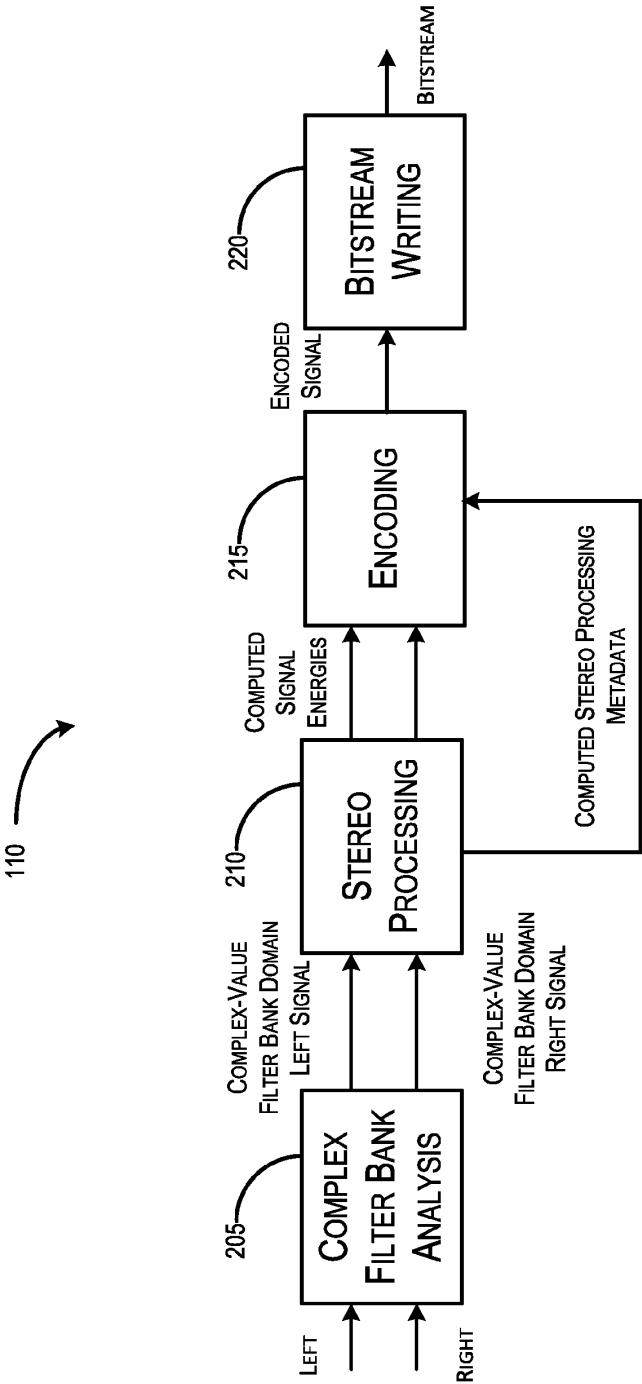


FIG. 2

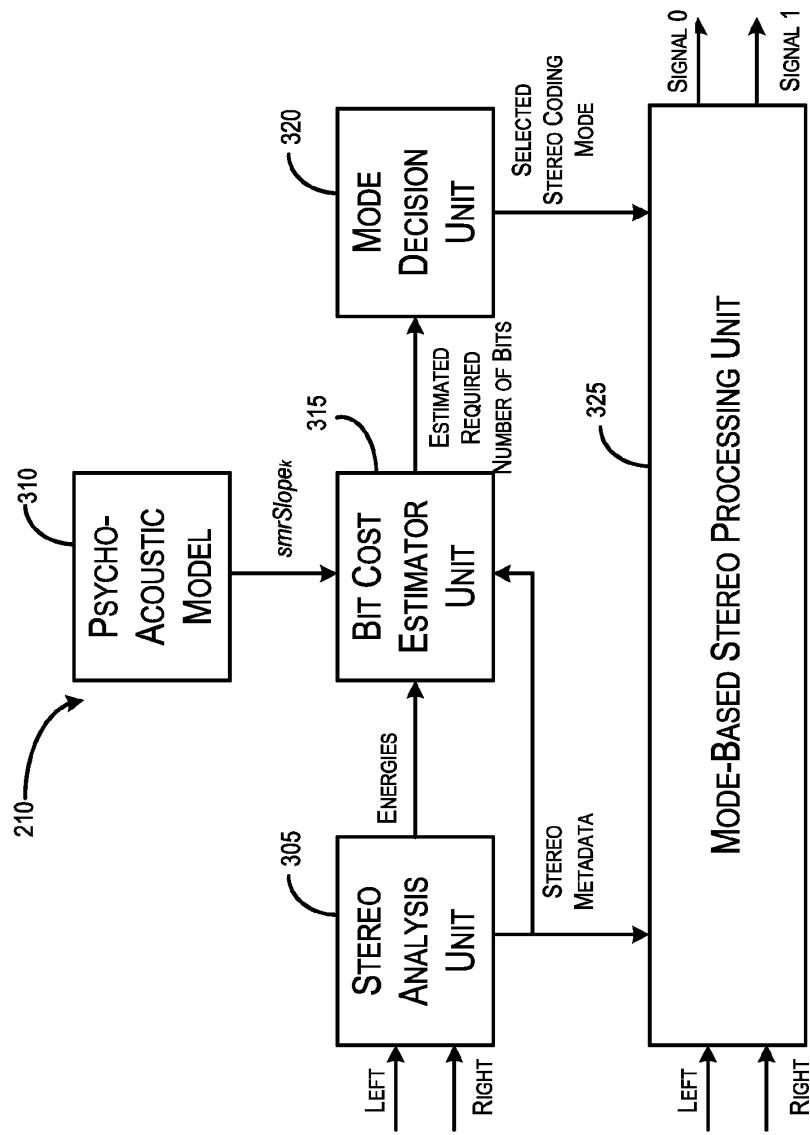


FIG. 3

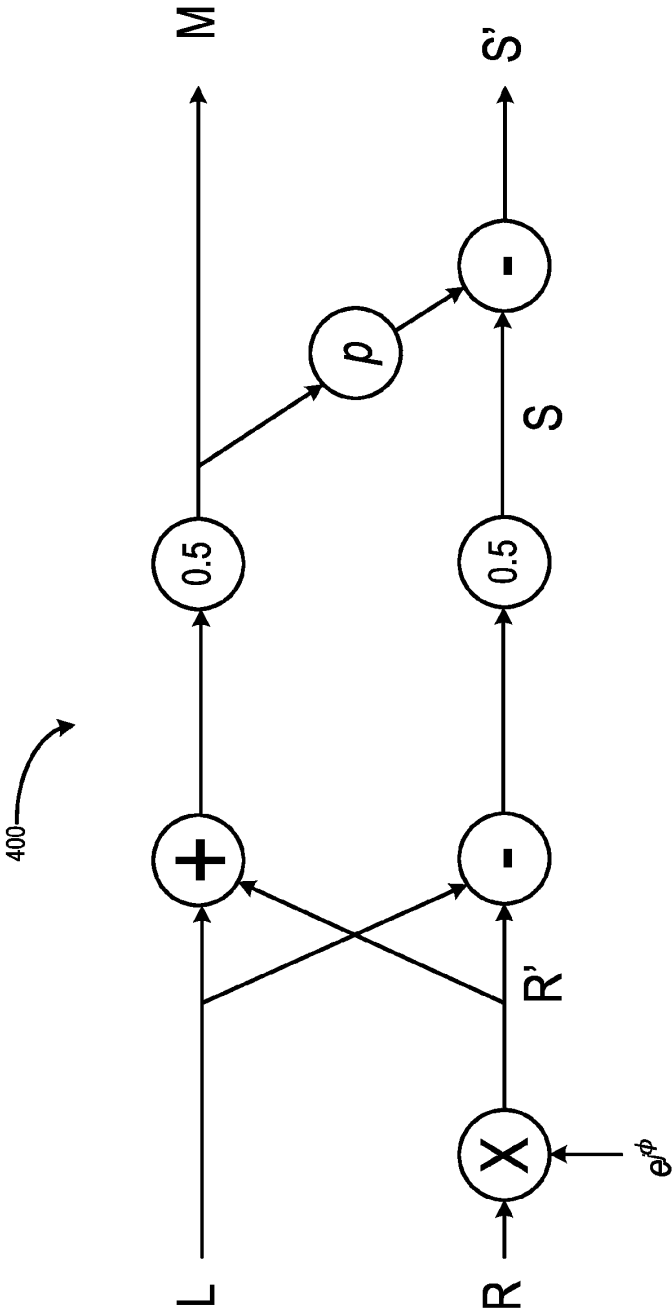


FIG. 4A

Stereo Coding Type	Parameters	Description	Related Bitstream Elements
0	None	No joint coding - Separate Left and Right coding	MSMode = 0
1	$\mathcal{P}_{k,Q} = 0, \varphi_{k,Q} = 0$	Standard Mid/Side coding	MSMode = 1 MSMode = 2
2	$\mathcal{P}_{k,Q} = 0, \varphi_{k,Q}! = 0$	Mid/Side coding with phase alignment only	MSMode = 3 AnyNonZeroPhase = 1 AnyNonZeroPred = 0
3	$\mathcal{P}_{k,Q}! = 0, \varphi_{k,Q} = 0$	Mid/Side coding with side prediction only	MSMode = 3 AnyNonZeroPhase = 0 AnyNonZeroPred = 1
4	$\mathcal{P}_{k,Q}! = 0, \varphi_{k,Q}! = 0$	Mid/Side coding with phase alignment and side prediction	MSMode = 3 AnyNonZeroPhase = 0 AnyNonZeroPred = 1

Table 1

FIG. 4B

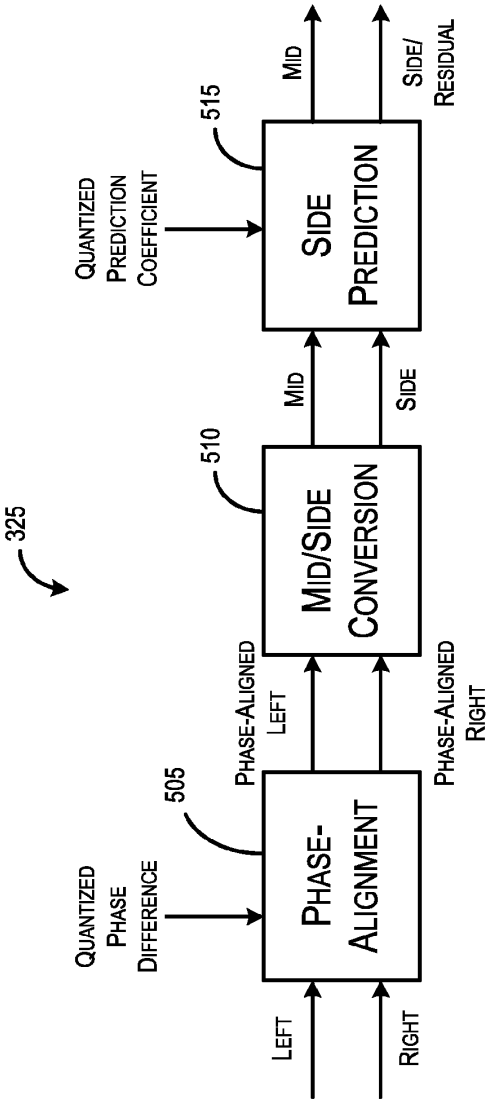


FIG. 5A

Name	Number of Bits	Semantic
MSMode	2	0=L/R, 1=M/S, 2=L/R or M/S, 3= M/Res
MSFlags[numBands]	If MSMode==2 NumBands else 0	If MSMode==0 MSFlags[numBands] = {0} elseif MSMode==1 MSFlags[numBands] = {1} elseif MSMode==2 for b=0:numBands-1 MSFlags[b] = ReadBits(1) if MSMode!=3 return
MsPredAll	1	0 = L/R or M/Res, 1 = M/Res
MSFlags[numBands]	If bMsPredAll==1 0 else NumBands	If MsPredAll ==1 MSFlags[numBands] = {1} NumMSBands = NumBands else NumMSBands = 0 for b=0:numBands-1 MSFlags[b] = ReadBits(1) NumMSBands += MSFlags[b]
AnyNonZeroPhase	1	
PhaseDiffs[numBands]	If AnyNonZeroPhase==0 0 else 5+Variable	If AnyNonZeroPhase == 0 PhaseDiffs[numBands]= {0} else PhaseDiffs[0] = ReadBits(5); for b=1: NumMSBands -1 PhaseDiffs[b] = HuffmanTable[ReadHuffmanWord];
AnyNonZeroPred	1	
PredCoeffs[numBands]	If AnyNonZeroPred ==0 0 else 5+Variable	If AnyNonZeroPred ==0 PredCoeffs[numBands]= {0} else PredCoeffs[0] = ReadBits(5); for b=1: NumMSBands-1 PredCoeffs [b] = HuffmanTable[ReadHuffmanWord];

Table 2

FIG. 5B

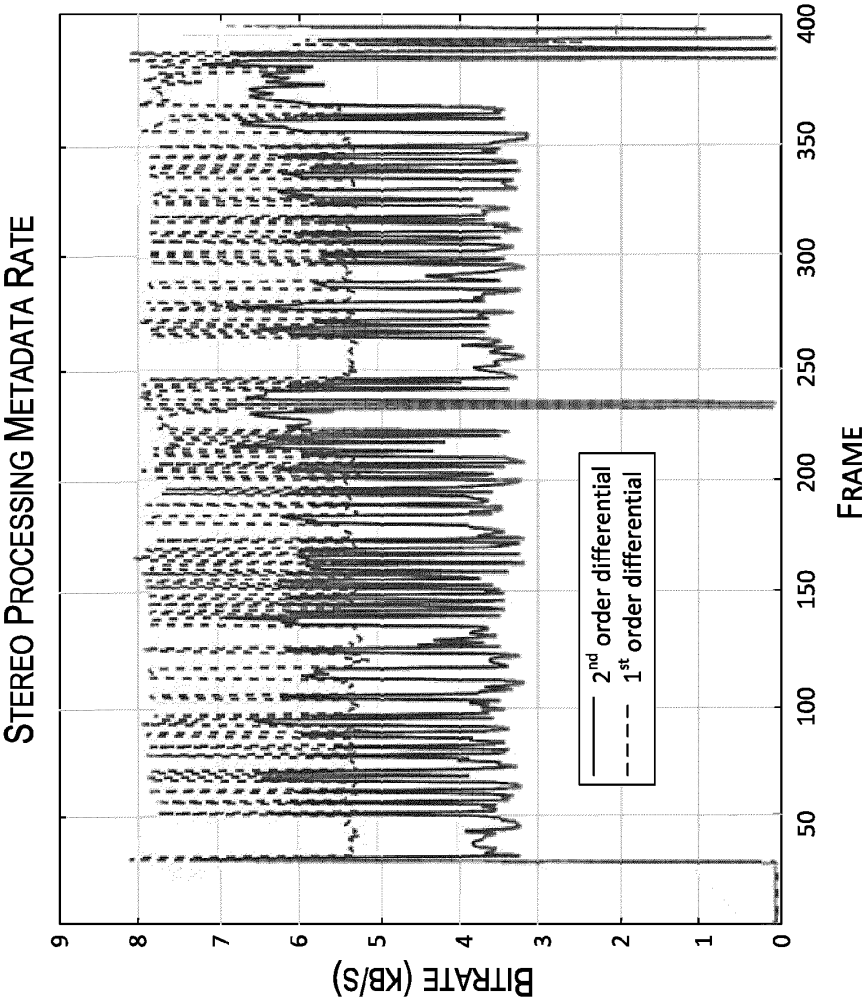


FIG. 6A

```
int PrepEncode(int *piQuant, const int *piMSFlags, const int numBands)
{
    int b, numMSBands = 0;
    for (b = 0; b < numBands; b++)
    {
        if (piMSFlags[b])
        { // store quantized values adjacently
            piQuant[numMSBands++] = piQuant[b];
        }
    }
    return numMSBands; // number of encoded data
}
```

Pseudocode 1

FIG. 6B

```
void EncodePhase(int *piPhaseQuant, int numMSBands, int diffDim)
{
    int b;
    if (diffDim > 0)
    { // first order differential
        int tmp1, tmp2;
        tmp1 = piPhaseQuant[0];
        for (b = 1; b < numMSBands; b++)
        {
            tmp2 = piPhaseQuant[b];
            piPhaseQuant[b] -= tmp1;
            tmp1 = tmp2;
        }
    }
    if (diffDim > 1)
    { // second order differential
        int tmp1, tmp2;
        tmp1 = piPhaseQuant[1];
        for (b = 2; b < numMSBands; b++)
        {
            tmp2 = piPhaseQuant[b];
            piPhaseQuant[b] -= tmp1;
            tmp1 = tmp2;
        }
    }
    for (b = 1; b < numMSBands; b++)
    { // wrap into quantized 2pi range
        piPhaseQuant[b] = RequantPhase(piPhaseQuant[b]);
    }
}
```

Pseudocode 2

FIG. 6C

```
int RequantPhase(int iPhaseQ)
{
    if (iPhaseQ >= PHASE_MAX_VAL)
    {
        iPhaseQ += PHASE_MAX_VAL;
        iPhaseQ %= (2 * PHASE_MAX_VAL);
        iPhaseQ -= PHASE_MAX_VAL;
    }
    else if (iPhaseQ < PHASE_MIN_VAL)
    {
        iPhaseQ -= PHASE_MAX_VAL;
        iPhaseQ %= (2 * PHASE_MAX_VAL);
        iPhaseQ += PHASE_MAX_VAL;
        if (iPhaseQ == PHASE_MAX_VAL)
        {
            iPhaseQ = PHASE_MIN_VAL;
        }
    }

    return iPhaseQ;
}
```

Pseudocode 3

FIG. 6D

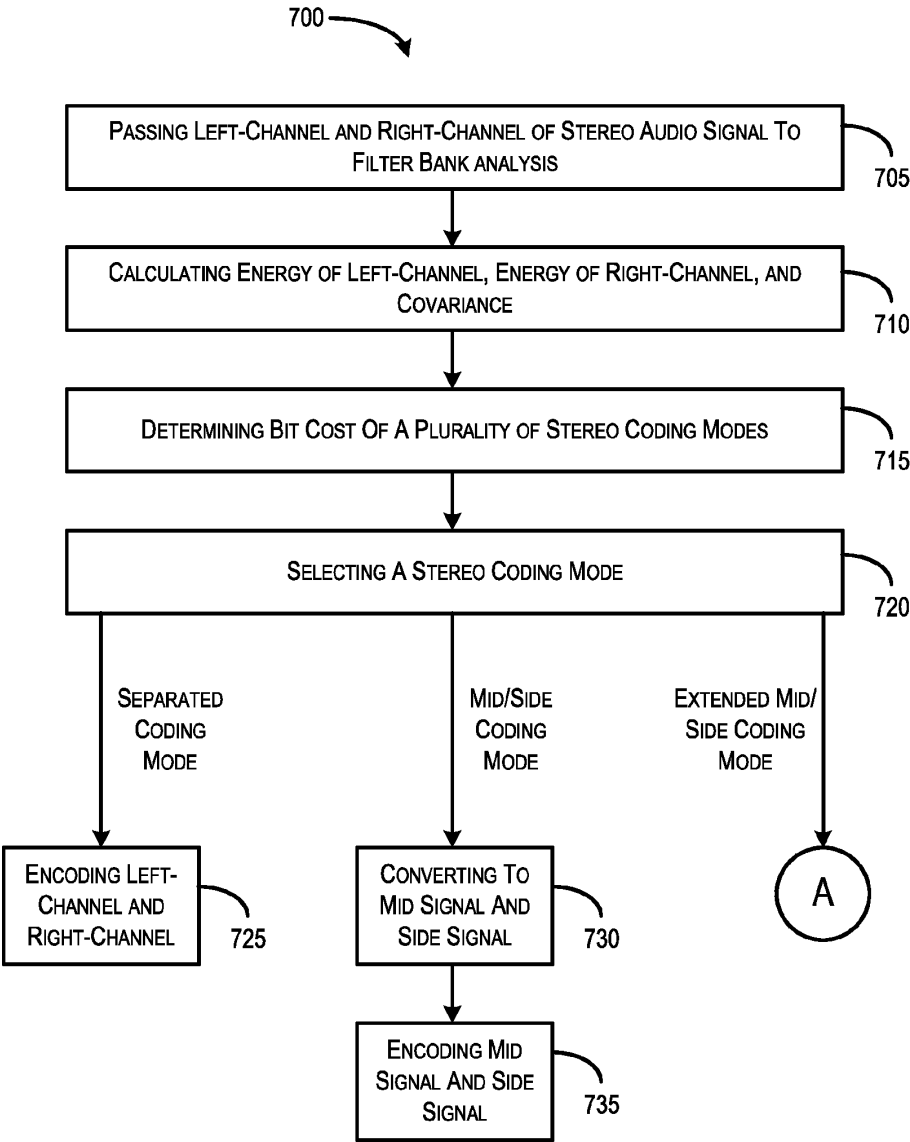


FIG. 7A

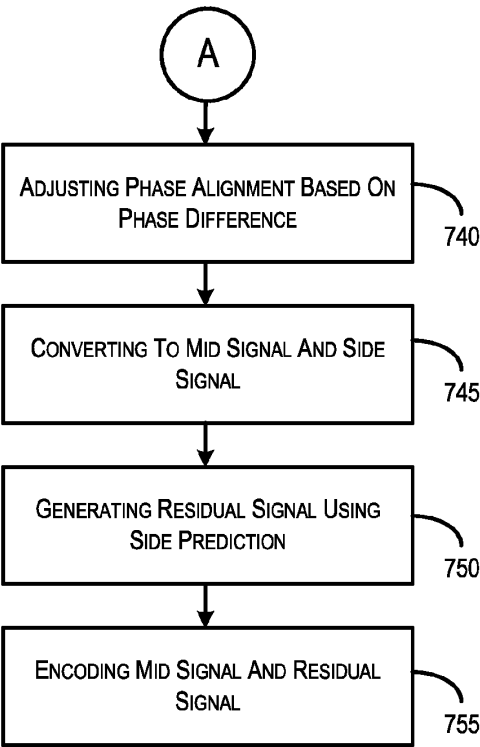


FIG. 7B

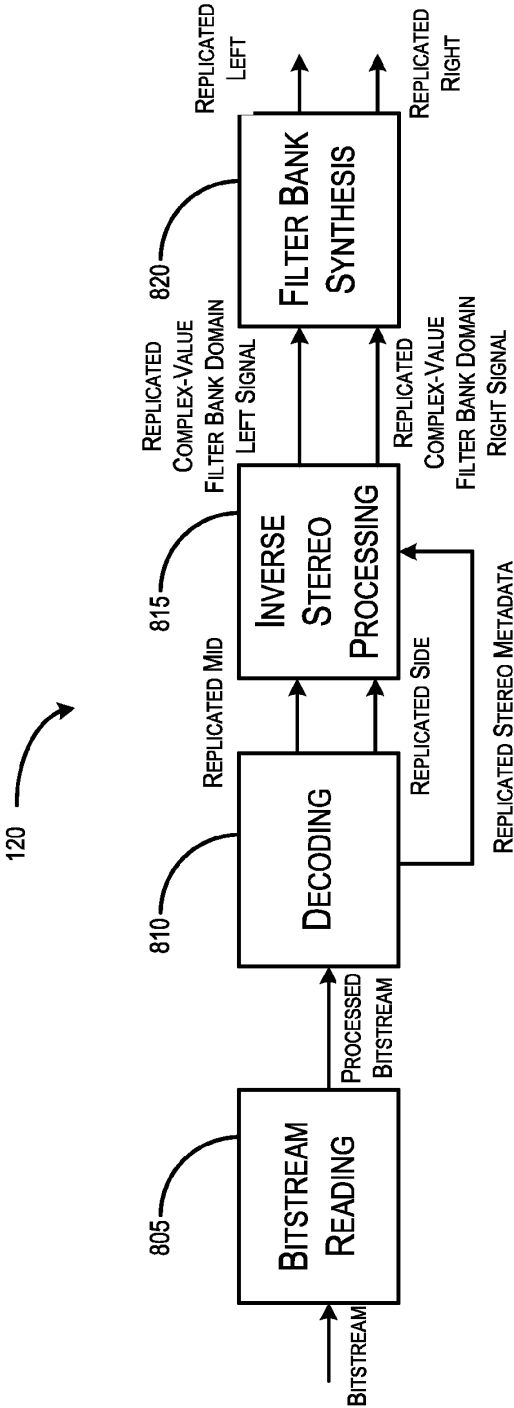


FIG. 8A

```
void DecodePhase(int *piPhaseQuant, int numMSBands, int diffDim)
{
    int b;
    if (diffDim > 1)
    {
        for (b = 2; b < numMSBands; b++)
        {
            piPhaseQuant[b] += piPhaseQuant[b - 1];
        }
    }
    if (diffDim > 0)
    {
        for (b = 1; b < numMSBands; b++)
        {
            piPhaseQuant[b] += piPhaseQuant[b - 1];
        }
    }
    for (b = 1; b < numMSBands; b++)
    {
        piPhaseQuant[b] = requantPhase(piPhaseQuant[b]);
    }
}
```

Pseudocode 4

FIG. 8B

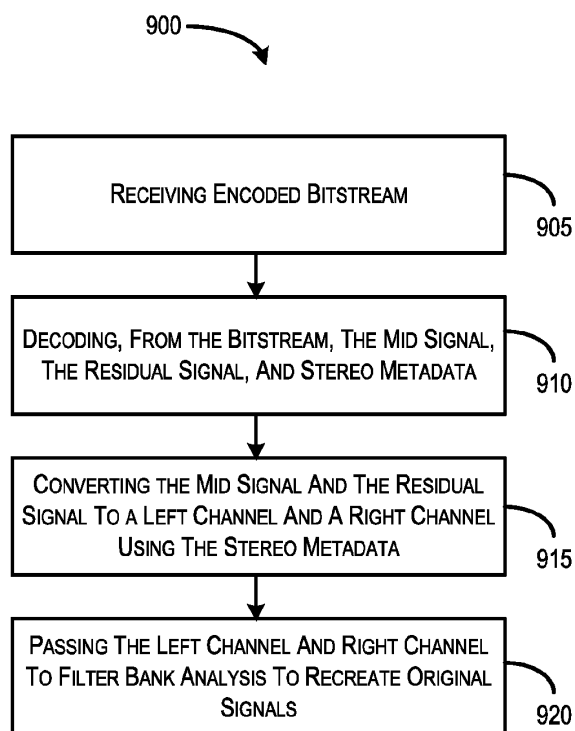


FIG. 9

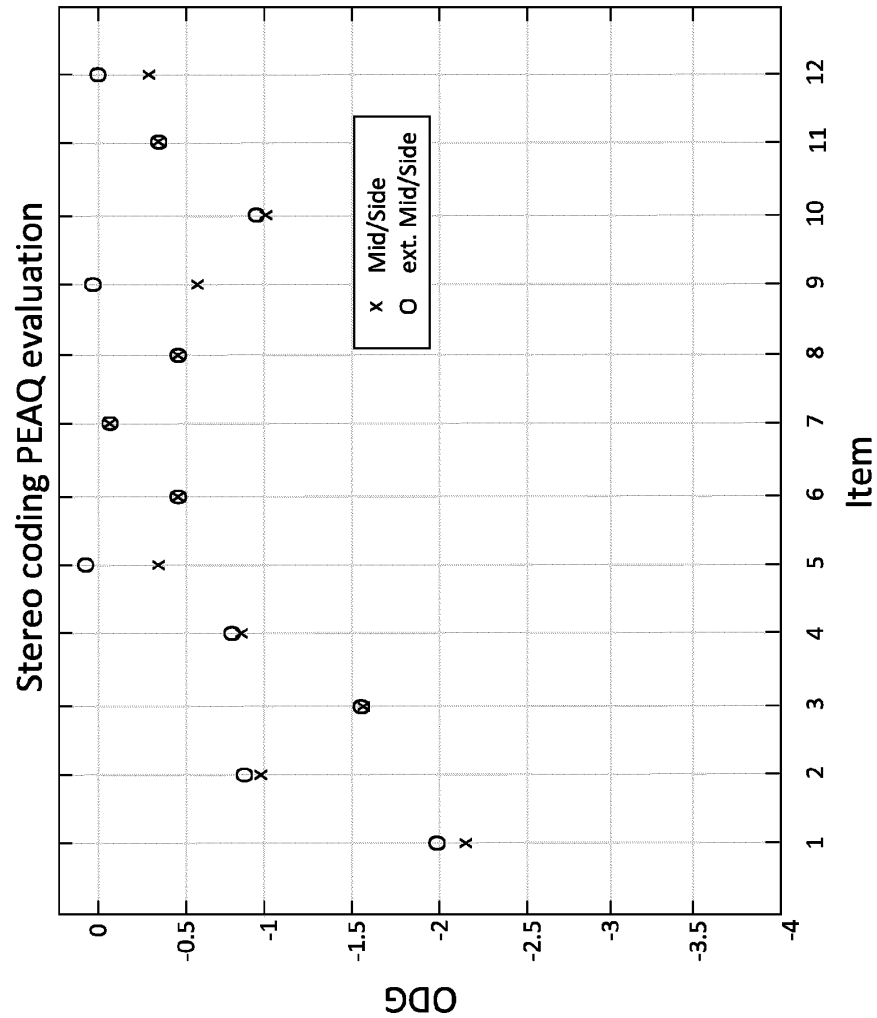


FIG. 10

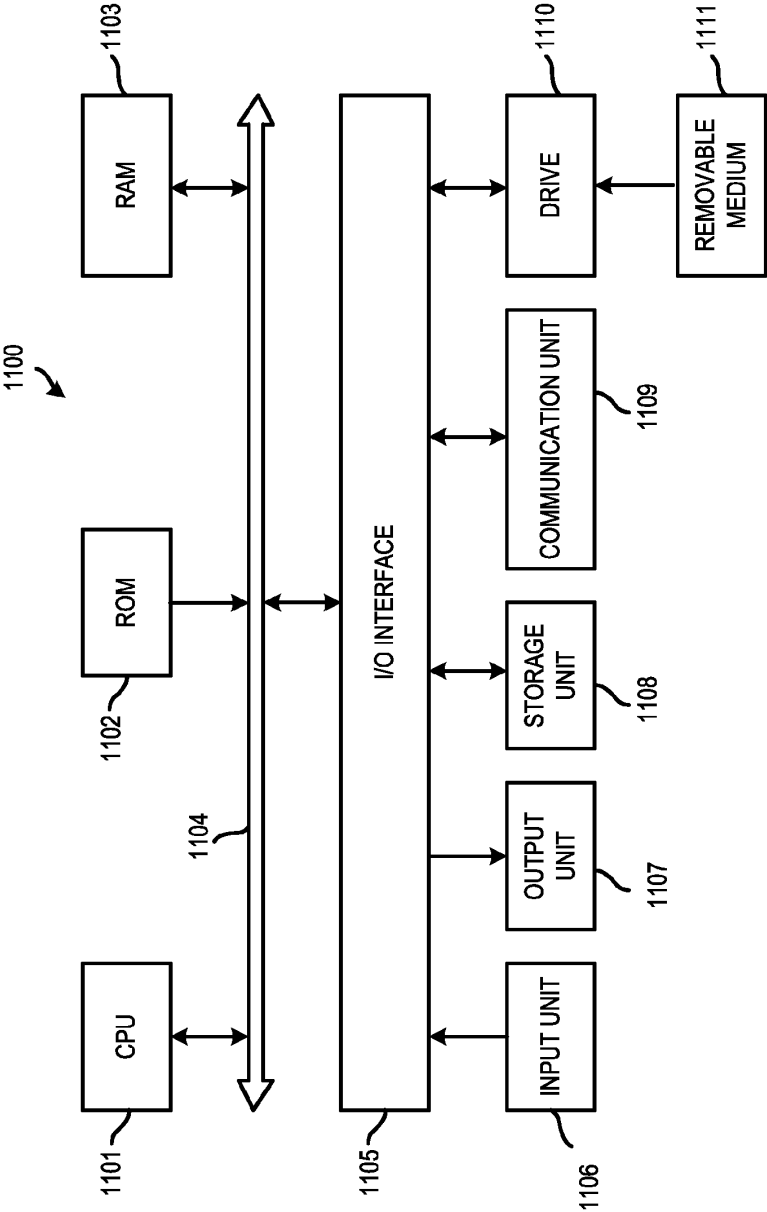


FIG. 11A

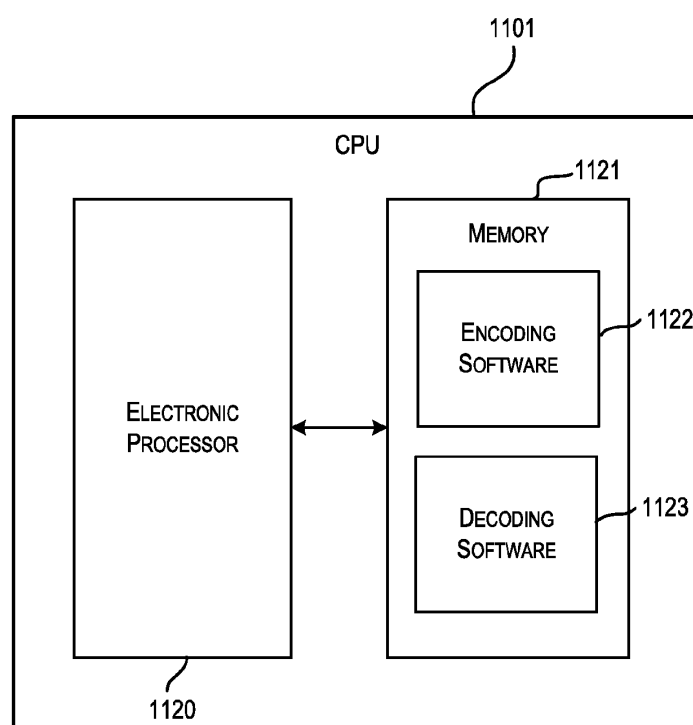


FIG. 11B