

Fig. 1

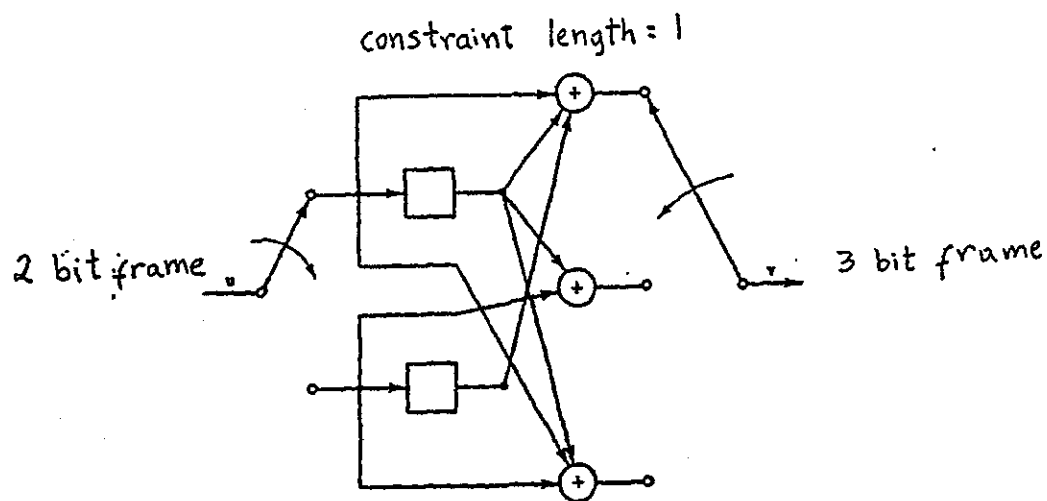


Fig. 2

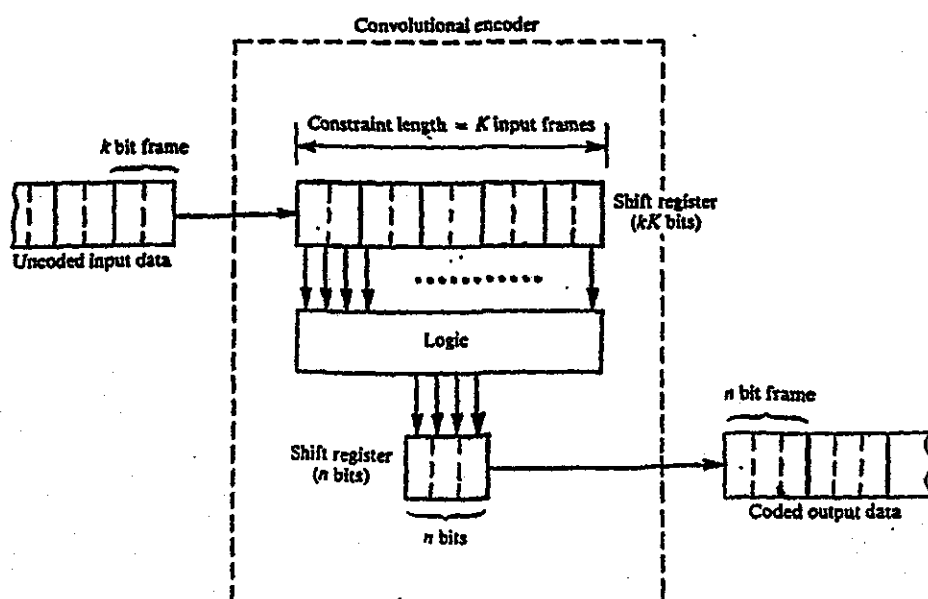


Fig. 3

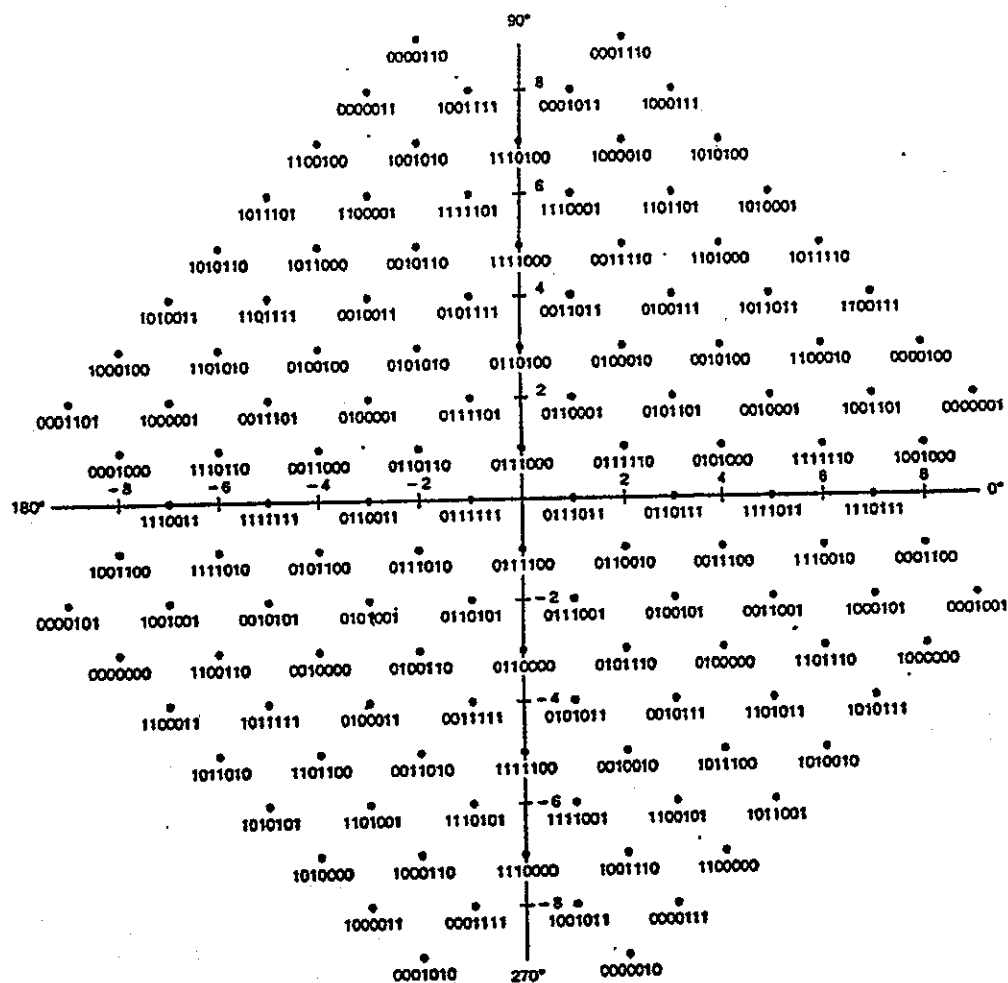


Fig. 4

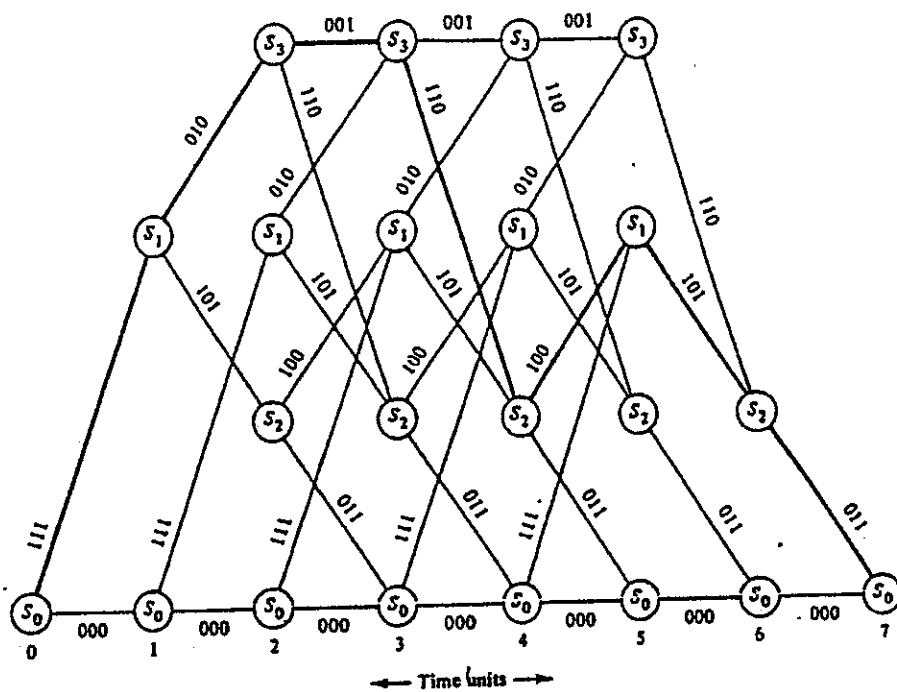
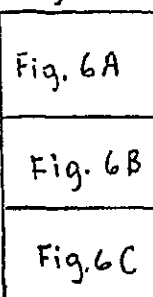
Figure 11.1 Trellis diagram for a (3, 1, 2) code with $L = 5$.

FIG. 5

Fig. 6



```

1  BranchMetricsCalculation(){
2      int StateIndex;
3      int BranchIndex;
4      int PreviousState;
5      int MinState = 0;
6      int OldState = 0;
7
8      int Number_of_States;
9      int MinStateMetrics = 32767;
10     int n = 0; // array index
11     int m; //temporary variable
12     array PathMetricsIndex[]; // Collection of indexes
13                                     //into the PathMetricStorage.
14     array PathMetrics[]; // Storage array for path
15                                     // metric values/distances.
16     array BranchMetricsIndex[]; // Collection of indexes
17                                     //into the BranchMetricStorage.
18     array BranchMetrics[]; // Storage array for branch
19                                     // metric values.
20     array NewPathMetrics[]; // Storage for new path
21                                     // distances.
22
23
24     for (StateIndex=0; StateIndex < Number_of_States;
25         StateIndex++)
26     {
27         // Search for the best branch into the X state
28         for ( BranchIndex = 0; BranchIndex < 4; BranchIndex++)
29         {
30             PreviousState = PathMetricsIndex[n];
31
32             // Add Path metrics
33             m = PathMetrics[PreviousState] +
34             BranchMetrics[(BranchMetricsIndex[n])];
35
36             // Update if a shorter minimum path is found
37             if (m < NewPathMetrics[StateIndex])
38             {
39                 NewPathMetrics[StateIndex] = m;
40                 OldState = PathMetricsIndex[n];
41             }
42             n++;
43         }
44
45         // Save old state

```

FIGURE 6A

```

46     StateMem[8*TraceBackIndex + StateIndex] = 8*
47     (TraceBackIndex + 1) & Trace_Back_Length_Mask) + OldState;
48
49     // Search for Best State
50     if ( NewPathMetrics[StateIndex] < MinStateMetrics)
51     {
52         MinStateMetrics=NewPathMetrics[StateIndex];
53         MinState=StateIndex;
54     }
55
56     /* Normalization */
57     for (i=0; i<Number_of_States; i+)
58         PathMetrics[i] = NewPathMetrics[i] -
59         MinStateMetrics;
60
61     // Trace Back from the MinState
62     TraceBackState = Number_of_States*TraceBackIndex +
63     MinState;
64
65     for (i=0; i<Trace_Back_Length - 2; i+=2)
66     {
67         TraceBackState = StateMem[TraceBackState];
68         TraceBackState = StateMem[TraceBackState];
69     }
70
71     NewState = TraceBackState & Number_of_States_Mask;
72     OldState = StateMem[TraceBackState] &
73     Number_of_States_Mask;
74     Group = GetGroup [8*OldState + NewState];
75
76     // Re-slice for the selected group
77     TraceBackIQ[2* TraceBackIndex] = Zx;
78     TraceBackIQ[2* TraceBackIndex + 1] = Zy;
79
80     zindex = (TraceBackIndex + Trace_Back_Length - 2) &
81     Trace_Back_Length_Mask;
82
83     // Decide on the Symbol
84
85     TCMDecision(TraceBackIQ[2* zindex], TraceBackIQ[2*
86     zindex + 1], Group, Symbol, &Ax, &Ay, ModulationType, p);
87
88     // Re-slice for the Equalizer and
89     // Carrier Recovery Adaptation
90     TraceBackState = Number_of_States * TraceBackIndex +
91     MinState;
92

```

FIGURE 6B

```
93     NewState = TraceBackState & Number_of_States_Mask;
94     OldState = StateMem[TraceBackState] &
95     Number_of_States_Mask;
96     Group = GetGroup[8*OldState + NewState];
97
98     // Decide on the (ad) for the equalizer and the pll
99     TCMDecision(Zx, Zy, Group, & TempSymbol, Ar, Ai,
100     ModulationType, p);
101
102     // Update the Trace Back Index
103     //(instead of doing --,
104     // we can do += 15 and modulus 16)
105     TraceBackIndex += Trace_Back_Length - 1;
106
107     TraceBackIndex &= Trace_Back_Length_Mask;
108 } // END OF BranchMetricsCalculation
```

FIGURE 6C

```

1  int i;
2  int TempSymbol;
3  int Index;
4  int BranchIndex;
5  int TraceBackState;
6  int NewState;
7  int Group;
8  int zindex
9
10 int MinStateMetrics = 32767;
11
12 int MinState = 0;
13
14 int n = 0;
15 array PM = {0,4,6,2,2,6,4,0,4,0,2,6,6,2,0,4,
16             1,3,7,5,7,5,1,3,5,7,3,1,3,1,5,7};
17 array Control_Array0[Number_of_States], Control_Array0
18 [Number_of_States];
19
20 array temp0 [Number_of_States], temp1 [Number_of_States],
21 temp2 [Number_of_States];
22 int32 reg;
23
24 int m[4];
25 int Min32[8] = (2147483647.0), (2147483647.0), .
26 (2147483647.0), (2147483647.0),
27 (2147483647.0), (2147483647.0), (2147483647.0),
28 (2147483647.0);
29 /*for (i=0; i<Number_of_States; i++)
30 Control_Array0[i]=0;
31
32 BranchMetricsCalculation()
33 {    // branch metrics for 'even' transitions
34     for (Index=0; Index<4; Index++)
35     {
36         // search for the best branch into the x state
37
38         // Add
39         for (i = 0; i<4; i++)
40         {
41
42             m[i] = (PathMetrics [n+i]<<16) +
43 (BranchMetrics [i]<<16);
44             if (m[i] > 2147483647) m[i] = 2147483647;
45
46         }
47

```

Fig 7

Fig. 7A

Fig. 7B

Fig 7C

Fig 7D

FIGURE 7A

```

48      //Find Minimum
49      for (i = 0; i<4; i++)
50      {
51          if (m[i]<Min32[Index])
52          {
53              Min32[Index] = m[i];
54              reg = i;
55          }
56      }
57
58      NewPathMetrics[Index] = (m[reg]/65536.0);
59      Control_Array0[Index] = PM[n+reg];
60      Control_Array1[Index] = (reg<<1);
61
62      n+=4;
63
64      /* search for the best state */
65      if (NewPathMetrics[Index] < MinStateMetrics)
66      {
67          MinStateMetrics = p->NewPathMetrics[Index];
68          MinState = Index;
69      }
70
71      // branch metrics for 'odd' transitions
72      for (Index=4; Index<8; Index++)
73      {
74          for (i = 0; i<4; i++)
75          {
76              m[i] = (PathMetrics[n+i]<<16) +
77              BranchMetrics [i+4]<<16);
78              if (m[i]>2147483647) m[i] = 2147483647;
79          }
80      }
81
82      //find minimum
83      for (i = 0; i<4; i++)
84      {
85          if (m[i]<=Index]. = Index])
86          {
87              Min32[Index] = m[i];
88              reg = i;
89          }
90      }
91      NewPathMetrics[Index] = (m[reg]/65536.0);
92      Control_Array0[Index] = PM[n+reg];
93
94

```

FIGURE 7B

```

95         Control_Array1[Index] = ((reg<<1))
96
97     n+=4;
98
99     if (NewPathMetrics[Index] , MinStateMetrics)
100     {
101         MinStateMetrics = NewPathMetrics[Index];
102         MinState = Index;
103     }
104 }
105 Index = Control_Array0[MinState];
106
107
108     Group = ((SurvivorY2[Index]&01) <<2)+
109     ((SurvivorY1[Index]&01) << 1)+(SurvivorY0[Index]&01);
110
111     // Normalization
112     for (i=0; i<32; i++) PathMetrics[i] =
113     NewPathMetrics[PM[i]] - MinStateMetrics;
114
115     for (Index=0 Index<N_of_States; Index++)
116     {
117         temp0[Index] = SurvivorY0[Index]>>1; //ps shift
118         right
119         temp1[Index] = SurvivorY1[Index]>>1;
120         temp2[Index] = SurvivorY2[Index]>>1;
121     }
122
123     for(Index=0; Index < 4; Index++)
124     {
125         i = Control_Array0[Index];
126         SurvivorY0[Index] = (temp0[i]);
127         SurvivorY1[Index] = (temp1[i]) |
128         ((Control_Array1[Index]&0x02) << (TRACEBACKY1));
129         SurvivorY2[Index] = (temp2[i]) |
130         ((Control_Array1[Index]&0x04) << (TRACEBACKY2));
131     }
132     for(Index=4; Index < 8; Index++)
133     {
134         i = Control_Array0[Index];
135         p->SurvivorY0[Index] = (temp0[i]) | ((0x01) <<
136         (TRACEBACKY0));
137         p->SurvivorY1[Index] = (temp1[i]) |
138         ((Control_Array1[Index]&0x02) << (TRACEBACKY1));
139         p->SurvivorY2[Index] = (temp2[i]) |
140         ((Control_Array1[Index]&0x04) << (TRACEBACKY1));
141     }

```

FIGURE 7C

```
142
143     //Re-slice for the selected group
144
145     zindex = (p->TraceBackIndex = TRACEBACKM2) &
146     Trace_Back_Length_Mask;
147
148     // Decide on the Symbol
149     Symbol[0] = Symbol[(zindex<<3) | Group];
150
151     //Reslice for the Equalizer and Carrier Recovery
152     // Adaptation
153     Group = Control_Array1[MinState];
154     TempSymbol = Symbol[TraceBackIndex<<3) | Group];
155
156     // Decide on the (ad) for the equalizer and the pll
157     *Ar = Iv17[TempSymbol];
158     *Ai = Qv17[TempSymbol];
159
160     // Update the Trace Back Index
161     // (instead of doing --, we can do += 15 and modulus
162 16) */
163     traceBackIndex += TRACEBACKM1;
164     TraceBackIndex &= Trace_Back_Length_Mask;
165
166 } //END OF BranchMetricsCalculation()
```

FIGURE 7D

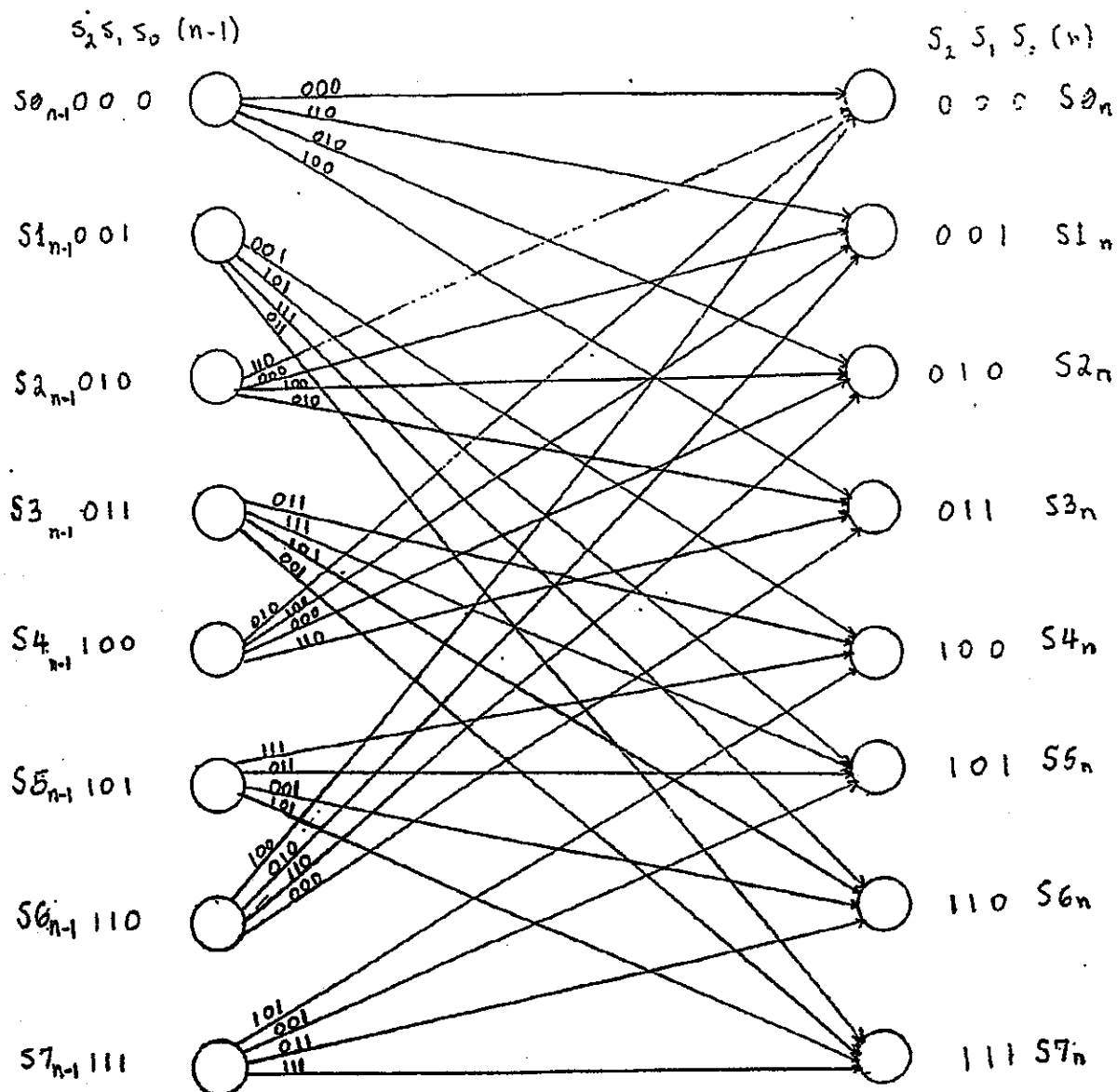


Fig. 8

V.32 bis trellis

$$\text{PM}[32] = \left(\begin{array}{c} \underbrace{\begin{array}{c} S0_n \\ 0, 4, 6, 2, \\ 1, 3, 7, 5, \end{array}}_{S4_n} \underbrace{\begin{array}{c} S1_n \\ 2, 6, 4, 0, \\ 7, 5, 1, 3, \end{array}}_{S5_n} \underbrace{\begin{array}{c} S2_n \\ 4, 0, 2, 6, \\ 5, 7, 3, 1, \end{array}}_{S6_n} \underbrace{\begin{array}{c} S3_n \\ 6, 2, 0, 4, \\ 3, 1, 5, 7, \end{array}}_{S7_n} \end{array} \right);$$

Fig. 9

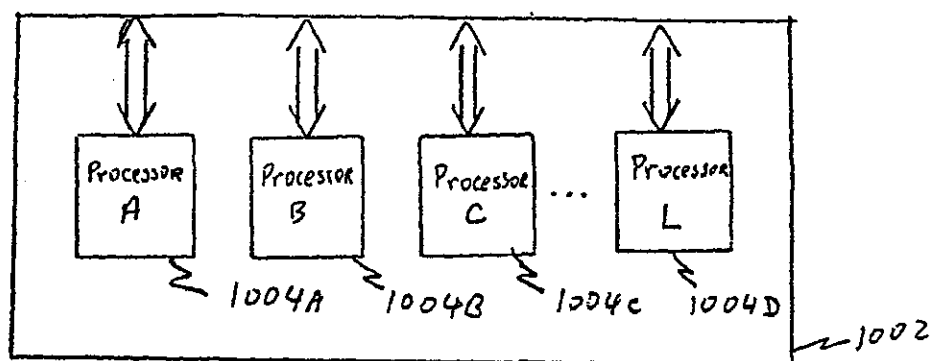


Fig. 10

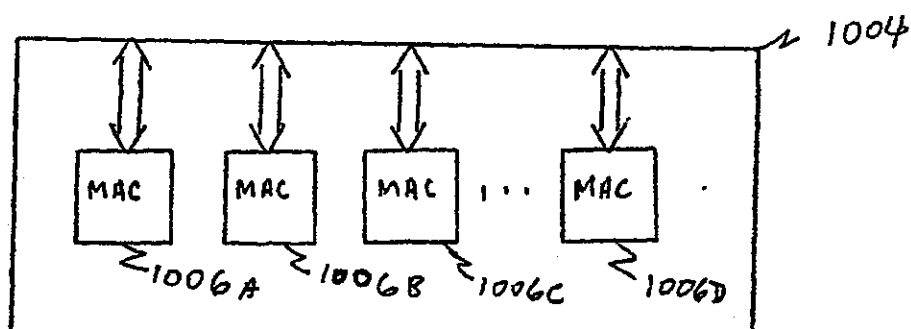


Fig 11

Y0	$X0_n^*$	$X0_{n-1}^*$		$X0_{n-15}^*$
Y1	$X1_n^*$	$X1_{n-1}^*$		$X1_{n-15}^*$
Y2	$X2_n^*$	$X2_{n-1}^*$		$X2_{n-15}^*$
Y3	$X3_n^*$	$X3_{n-1}^*$		$X3_{n-15}^*$
Y4	$X4_n^*$	$X4_{n-1}^*$		$X4_{n-15}^*$
Y5	$X5_n^*$	$X5_{n-1}^*$		$X5_{n-15}^*$
Y6	$X6_n^*$	$X6_{n-1}^*$		$X6_{n-15}^*$
Y7	$X7_n^*$	$X7_{n-1}^*$		$X7_{n-15}^*$
	n	n-1		n-15

Fig. 12

Y_i	y_2	s_{2n}^*	s_{2n-1}^*		s_{2n-14}^*	s_{2n-15}^*
	y_1	s_{1n}^*	s_{1n-1}^*		s_{1n-14}^*	s_{1n-15}^*
	y_0	s_{0n}^*	s_{0n-1}^*		s_{0n-14}^*	s_{0n-15}^*
		(n)	(n-1)		(n-14)	(n-15)

$$Y_i = \{y_{2i}, y_{1i}, y_{0i}\}$$

$$X_{kn}^* = (s_{2n}^* \ s_{1n}^* \ s_{0n}^*) = S_{kn}^*$$

Fig. 13

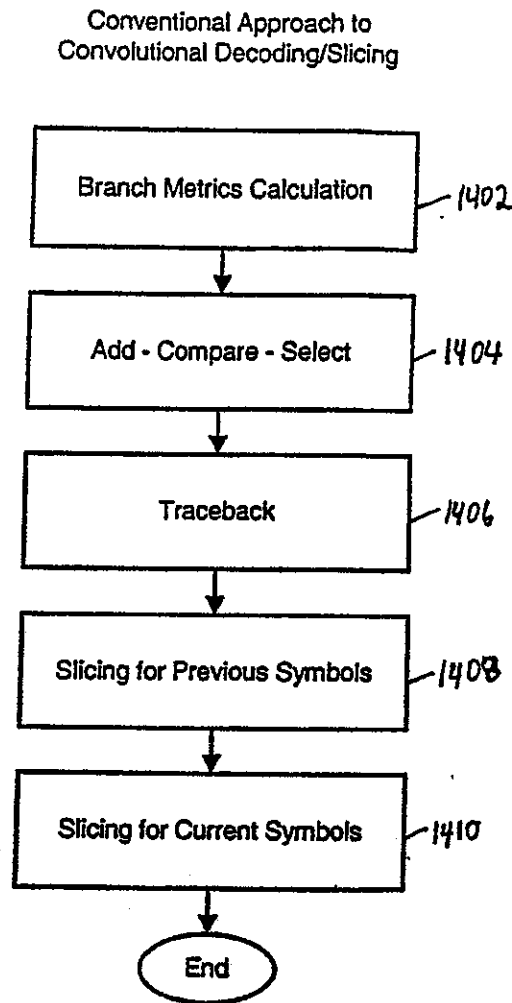


Figure 14

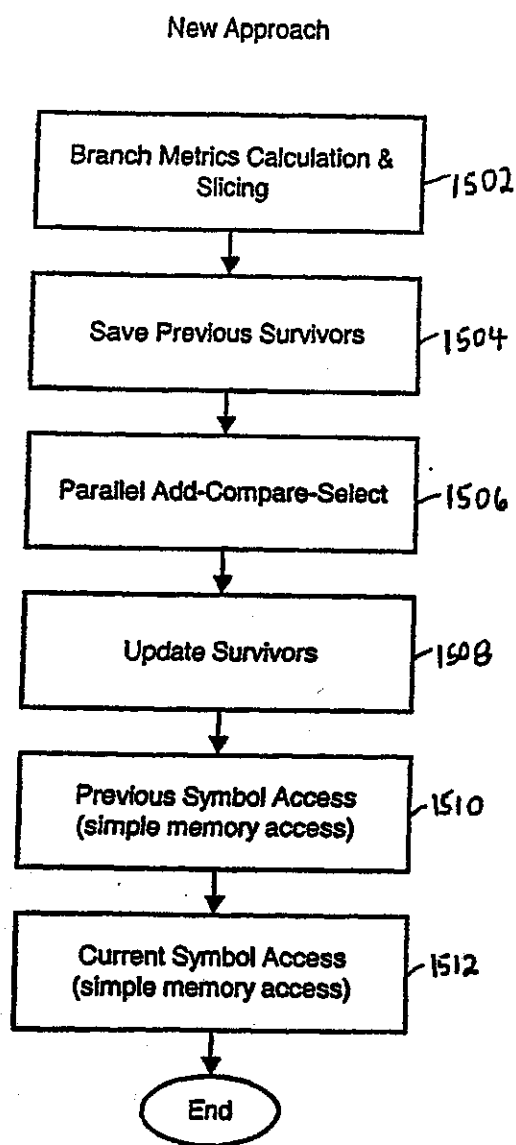


Figure 15