

5.3.6.4 系数反量化

5.3.6.4.1 量化参数

量化参数 QPIndex 的取值应为 0~255。帧解码开始时, QPIndex 初始设置为 base_qindex。每个解码块的 QPIndex 计算如下:

$$\text{QPIndex} = \text{QPIndex} + \text{dqp_sign} ? \text{dqp_abs} : -\text{dqp_abs}$$

亮度 DC 系数量化参数 QPY_dc 以 QPIndex、y_dc_delta_q 与 BitDepthY 为索引查表 50~表 52 得到, QPY_dc 为查表得到的 QP_dc, 其中 index=Clip3(0, 255, QPIndex+y_dc_delta_q)。亮度 AC 系数量化参数 QPY_ac 以 QPIndex、y_ac_delta_q 与 BitDepthY 为索引查表 53~表 55 得到, QPY_ac 为查表得到的 QP_ac, 其中 index=Clip3(0, 255, QPIndex)。色度 DC 系数量化参数 QPC_dc 以 QPIndex、uv_dc_delta_q 与 BitDepthY 为索引查表 50~表 52 得到, QPC_dc 为查表得到的 QP_dc, 其中 index=Clip3(0, 255, QPIndex+uv_dc_delta_q)。色度 AC 系数量化参数 QPC_ac 以 QPIndex、uv_ac_delta_q 与 BitDepthY 为索引查表 53~表 55 得到, QPC_ac 为查表得到的 QP_ac, 其中 index=Clip3(0, 255, QPIndex+uv_ac_delta_q)。

表 50 BitDepth=8 时, QP_dc 与 index=QPIndex+dc_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_dc	4	8	8	9	10	11	12	12
index	8	9	10	11	12	13	14	15
QP_dc	13	14	15	16	17	18	19	19
index	16	17	18	19	20	21	22	23
QP_dc	20	21	22	23	24	25	26	26
index	24	25	26	27	28	29	30	31
QP_dc	27	28	29	30	31	32	32	33
index	32	33	34	35	36	37	38	39
QP_dc	34	35	36	37	38	38	39	40
index	40	41	42	43	44	45	46	47
QP_dc	41	42	43	43	44	45	46	47
index	48	49	50	51	52	53	54	55
QP_dc	48	48	49	50	51	52	53	53
index	56	57	58	59	60	61	62	63
QP_dc	54	55	56	57	57	58	59	60
index	64	65	66	67	68	69	70	71
QP_dc	61	62	62	63	64	65	66	66
index	72	73	74	75	76	77	78	79
QP_dc	67	68	69	70	70	71	72	73

表 50 (续)

index	80	81	82	83	84	85	86	87
QP_dc	74	74	75	76	77	78	78	79
index	88	89	90	91	92	93	94	95
QP_dc	80	81	81	82	83	84	85	85
index	96	97	98	99	100	101	102	103
QP_dc	87	88	90	92	93	95	96	98
index	104	105	106	107	108	109	110	111
QP_dc	99	101	102	104	105	107	108	110
index	112	113	114	115	116	117	118	119
QP_dc	111	113	114	116	117	118	120	121
index	120	121	122	123	124	125	126	127
QP_dc	123	125	127	129	131	134	136	138
index	128	129	130	131	132	133	134	135
QP_dc	140	142	144	146	148	150	152	154
index	136	137	138	139	140	141	142	143
QP_dc	156	158	161	164	166	169	172	174
index	144	145	146	147	148	149	150	151
QP_dc	177	180	182	185	187	190	192	195
index	152	153	154	155	156	157	158	159
QP_dc	199	202	205	208	211	214	217	220
index	160	161	162	163	164	165	166	167
QP_dc	223	226	230	233	237	240	243	247
index	168	169	170	171	172	173	174	175
QP_dc	250	253	257	261	265	269	272	276
index	176	177	178	179	180	181	182	183
QP_dc	280	284	288	292	296	300	304	309
index	184	185	186	187	188	189	190	191
QP_dc	313	317	322	326	330	335	340	344
index	192	193	194	195	196	197	198	199
QP_dc	349	354	359	364	369	374	379	384
index	200	201	202	203	204	205	206	207
QP_dc	389	395	400	406	411	417	423	429

表 50 (续)

index	208	209	210	211	212	213	214	215
QP_dc	435	441	447	454	461	467	475	482
index	216	217	218	219	220	221	222	223
QP_dc	489	497	505	513	522	530	539	549
index	224	225	226	227	228	229	230	231
QP_dc	559	569	579	590	602	614	626	640
index	232	233	234	235	236	237	238	239
QP_dc	654	668	684	700	717	736	755	775
index	240	241	242	243	244	245	246	247
QP_dc	796	819	843	869	896	925	955	988
index	248	249	250	251	252	253	254	255
QP_dc	1 022	1 058	1 098	1 139	1 184	1 232	1 282	1 336

表 51 BitDepth = 10 时, QP_dc 与 index=QPIndex + dc_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_dc	4	9	10	13	15	17	20	22
index	8	9	10	11	12	13	14	15
QP_dc	25	28	31	34	37	40	43	47
index	16	17	18	19	20	21	22	23
QP_dc	50	53	57	60	64	68	71	75
index	24	25	26	27	28	29	30	31
QP_dc	78	82	86	90	93	97	101	105
index	32	33	34	35	36	37	38	39
QP_dc	109	113	116	120	124	128	132	136
index	40	41	42	43	44	45	46	47
QP_dc	140	143	147	151	155	159	163	166
index	48	49	50	51	52	53	54	55
QP_dc	170	174	178	182	185	189	193	197
index	56	57	58	59	60	61	62	63
QP_dc	200	204	208	212	215	219	223	226
index	64	65	66	67	68	69	70	71
QP_dc	230	233	237	241	244	248	251	255

表 51 (续)

index	72	73	74	75	76	77	78	79
QP_dc	259	262	266	269	273	276	280	283
index	80	81	82	83	84	85	86	87
QP_dc	287	290	293	297	300	304	307	310
index	88	89	90	91	92	93	94	95
QP_dc	314	317	321	324	327	331	334	337
index	96	97	98	99	100	101	102	103
QP_dc	343	350	356	362	369	375	381	387
index	104	105	106	107	108	109	110	111
QP_dc	394	400	406	412	418	424	430	436
index	112	113	114	115	116	117	118	119
QP_dc	442	448	454	460	466	472	478	484
index	120	121	122	123	124	125	126	127
QP_dc	490	499	507	516	525	533	542	550
index	128	129	130	131	132	133	134	135
QP_dc	559	567	576	584	592	601	609	617
index	136	137	138	139	140	141	142	143
QP_dc	625	634	644	655	666	676	687	698
index	144	145	146	147	148	149	150	151
QP_dc	708	718	729	739	749	759	770	782
index	152	153	154	155	156	157	158	159
QP_dc	795	807	819	831	844	856	868	880
index	160	161	162	163	164	165	166	167
QP_dc	891	906	920	933	947	961	975	988
index	168	169	170	171	172	173	174	175
QP_dc	1 001	1 015	1 030	1 045	1 061	1 076	1 090	1 105
index	176	177	178	179	180	181	182	183
QP_dc	1 120	1 137	1 153	1 170	1 186	1 202	1 218	1 236
index	184	185	186	187	188	189	190	191
QP_dc	1 253	1 271	1 288	1 306	1 323	1 342	1 361	1 379
index	192	193	194	195	196	197	198	199
QP_dc	1 398	1 416	1 436	1 456	1 476	1 496	1 516	1 537

表 51 (续)

index	200	201	202	203	204	205	206	207
QP_dc	1 559	1 580	1 601	1 624	1 647	1 670	1 692	1 717
index	208	209	210	211	212	213	214	215
QP_dc	1 741	1 766	1 791	1 817	1 844	1 871	1 900	1 929
index	216	217	218	219	220	221	222	223
QP_dc	1 958	1 990	2 021	2 054	2 088	2 123	2 159	2 197
index	224	225	226	227	228	229	230	231
QP_dc	2 236	2 276	2 319	2 363	2 410	2 458	2 508	2 561
index	232	233	234	235	236	237	238	239
QP_dc	2 616	2 675	2 737	2 802	2 871	2 944	3 020	3 102
index	240	241	242	243	244	245	246	247
QP_dc	3 188	3 280	3 375	3 478	3 586	3 702	3 823	3 953
index	248	249	250	251	252	253	254	255
QP_dc	4 089	4 236	4 394	4 559	4 737	4 929	5 130	5 347

表 52 BitDepth = 12 时, QP_dc 与 index=QPIndex+dc_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_dc	4	12	18	25	33	41	50	60
index	8	9	10	11	12	13	14	15
QP_dc	70	80	91	103	115	127	140	153
index	16	17	18	19	20	21	22	23
QP_dc	166	180	194	208	222	237	251	266
index	24	25	26	27	28	29	30	31
QP_dc	281	296	312	327	343	358	374	390
index	32	33	34	35	36	37	38	39
QP_dc	405	421	437	453	469	484	500	516
index	40	41	42	43	44	45	46	47
QP_dc	532	548	564	580	596	611	627	643
index	48	49	50	51	52	53	54	55
QP_dc	659	674	690	706	721	737	752	768
index	56	57	58	59	60	61	62	63
QP_dc	783	798	814	829	844	859	874	889

表 52 (续)

index	64	65	66	67	68	69	70	71
QP_dc	904	919	934	949	964	978	993	1 008
index	72	73	74	75	76	77	78	79
QP_dc	1 022	1 037	1 051	1 065	1 080	1 094	1 108	1 122
index	80	81	82	83	84	85	86	87
QP_dc	1 136	1 151	1 165	1 179	1 192	1 206	1 220	1 234
index	88	89	90	91	92	93	94	95
QP_dc	1 248	1 261	1 275	1 288	1 302	1 315	1 329	1 342
index	96	97	98	99	100	101	102	103
QP_dc	1 368	1 393	1 419	1 444	1 469	1 494	1 519	1 544
index	104	105	106	107	108	109	110	111
QP_dc	1 569	1 594	1 618	1 643	1 668	1 692	1 717	1 741
index	112	113	114	115	116	117	118	119
QP_dc	1 765	1 789	1 814	1 838	1 862	1 885	1 909	1 933
index	120	121	122	123	124	125	126	127
QP_dc	1 957	1 992	2 027	2 061	2 096	2 130	2 165	2 199
index	128	129	130	131	132	133	134	135
QP_dc	2 233	2 267	2 300	2 334	2 367	2 400	2 434	2 467
index	136	137	138	139	140	141	142	143
QP_dc	2 499	2 532 _{SZ16}	2 575	2 618	2 661	2 704	2 746	2 788
index	144	145	146	147	148	149	150	151
QP_dc	2 830	2 872	2 913	2 954	2 995	3 036	3 076	3 127
index	152	153	154	155	156	157	158	159
QP_dc	3 177	3 226	3 275	3 324	3 373	3 421	3 469	3 517
index	160	161	162	163	164	165	166	167
QP_dc	3 565	3 621	3 677	3 733	3 788	3 843	3 897	3 951
index	168	169	170	171	172	173	174	175
QP_dc	4 005	4 058	4 119	4 181	4 241	4 301	4 361	4 420
index	176	177	178	179	180	181	182	183
QP_dc	4 479	4 546	4 612	4 677	4 742	4 807	4 871	4 942
index	184	185	186	187	188	189	190	191
QP_dc	5 013	5 083	5 153	5 222	5 291	5 367	5 442	5 517

表 52 (续)

index	192	193	194	195	196	197	198	199
QP_dc	5 591	5 665	5 745	5 825	5 905	5 984	6 063	6 149
index	200	201	202	203	204	205	206	207
QP_dc	6 234	6 319	6 404	6 495	6 587	6 678	6 769	6 867
index	208	209	210	211	212	213	214	215
QP_dc	6 966	7 064	7 163	7 269	7 376	7 483	7 599	7 715
index	216	217	218	219	220	221	222	223
QP_dc	7 832	7 958	8 085	8 214	8 352	8 492	8 635	8 788
index	224	225	226	227	228	229	230	231
QP_dc	8 945	9 104	9 275	9 450	9 639	9 832	10 031	10 245
index	232	233	234	235	236	237	238	239
QP_dc	10 465	10 702	10 946	11 210	11 482	11 776	12 081	12 409
index	240	241	242	243	244	245	246	247
QP_dc	12 750	13 118	13 501	13 913	14 343	14 807	15 290	15 812
index	248	249	250	251	252	253	254	255
QP_dc	16 356	16 943	17 575	18 237	18 949	19 718	20 521	21 387

表 53 BitDepth=8 时, QP_ac 与 index=QPIndex + ac_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_ac	4	8	9	10	11	12	13	14
index	8	9	10	11	12	13	14	15
QP_ac	15	16	17	18	19	20	21	22
index	16	17	18	19	20	21	22	23
QP_ac	23	24	25	26	27	28	29	30
index	24	25	26	27	28	29	30	31
QP_ac	31	32	33	34	35	36	37	38
index	32	33	34	35	36	37	38	39
QP_ac	39	40	41	42	43	44	45	46
index	40	41	42	43	44	45	46	47
QP_ac	47	48	49	50	51	52	53	54
index	48	49	50	51	52	53	54	55
QP_ac	55	56	57	58	59	60	61	62

表 53 (续)

index	56	57	58	59	60	61	62	63
QP_ac	63	64	65	66	67	68	69	70
index	64	65	66	67	68	69	70	71
QP_ac	71	72	73	74	75	76	77	78
index	72	73	74	75	76	77	78	79
QP_ac	79	80	81	82	83	84	85	86
index	80	81	82	83	84	85	86	87
QP_ac	87	88	89	90	91	92	93	94
index	88	89	90	91	92	93	94	95
QP_ac	95	96	97	98	99	100	101	102
index	96	97	98	99	100	101	102	103
QP_ac	104	106	108	110	112	114	116	118
index	104	105	106	107	108	109	110	111
QP_ac	120	122	124	126	128	130	132	134
index	112	113	114	115	116	117	118	119
QP_ac	136	138	140	142	144	146	148	150
index	120	121	122	123	124	125	126	127
QP_ac	152	155	158	161	164	167	170	173
index	128	129	130	131	132	133	134	135
QP_ac	176	179	182	185	188	191	194	197
index	136	137	138	139	140	141	142	143
QP_ac	200	203	207	211	215	219	223	227
index	144	145	146	147	148	149	150	151
QP_ac	231	235	239	243	247	251	255	260
index	152	153	154	155	156	157	158	159
QP_ac	265	270	275	280	285	290	295	300
index	160	161	162	163	164	165	166	167
QP_ac	305	311	317	323	329	335	341	347
index	168	169	170	171	172	173	174	175
QP_ac	353	359	366	373	380	387	394	401
index	176	177	178	179	180	181	182	183
QP_ac	408	416	424	432	440	448	456	465

表 53 (续)

index	184	185	186	187	188	189	190	191
QP_ac	474	483	492	501	510	520	530	540
index	192	193	194	195	196	197	198	199
QP_ac	550	560	571	582	593	604	615	627
index	200	201	202	203	204	205	206	207
QP_ac	639	651	663	676	689	702	715	729
index	208	209	210	211	212	213	214	215
QP_ac	743	757	771	786	801	816	832	848
index	216	217	218	219	220	221	222	223
QP_ac	864	881	898	915	933	951	969	988
index	224	225	226	227	228	229	230	231
QP_ac	1 007	1 026	1 046	1 066	1 087	1 108	1 129	1 151
index	232	233	234	235	236	237	238	239
QP_ac	1 173	1 196	1 219	1 243	1 267	1 292	1 317	1 343
index	240	241	242	243	244	245	246	247
QP_ac	1 369	1 396	1 423	1 451	1 479	1 508	1 537	1 567
index	248	249	250	251	252	253	254	255
QP_ac	1 597	1 628	1 660	1 692	1 725	1 759	1 793	1 828

表 54 BitDepth=10 时, QP_ac 与 index=QPIndex + ac_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_ac	4	9	11	13	16	18	21	24
index	8	9	10	11	12	13	14	15
QP_ac	27	30	33	37	40	44	48	51
index	16	17	18	19	20	21	22	23
QP_ac	55	59	63	67	71	75	79	83
index	24	25	26	27	28	29	30	31
QP_ac	88	92	96	100	105	109	114	118
index	32	33	34	35	36	37	38	39
QP_ac	122	127	131	136	140	145	149	154
index	40	41	42	43	44	45	46	47
QP_ac	158	163	168	172	177	181	186	190

表 54 (续)

index	48	49	50	51	52	53	54	55
QP_ac	195	199	204	208	213	217	222	226
index	56	57	58	59	60	61	62	63
QP_ac	231	235	240	244	249	253	258	262
index	64	65	66	67	68	69	70	71
QP_ac	267	271	275	280	284	289	293	297
index	72	73	74	75	76	77	78	79
QP_ac	302	306	311	315	319	324	328	332
index	80	81	82	83	84	85	86	87
QP_ac	337	341	345	349	354	358	362	367
index	88	89	90	91	92	93	94	95
QP_ac	371	375	379	384	388	392	396	401
index	96	97	98	99	100	101	102	103
QP_ac	409	417	425	433	441	449	458	466
index	104	105	106	107	108	109	110	111
QP_ac	474	482	490	498	506	514	523	531
index	112	113	114	115	116	117	118	119
QP_ac	539	547	555	563	571	579	588	596
index	120	121	122	123	124	125	126	127
QP_ac	604	616	628	640	652	664	676	688
index	128	129	130	131	132	133	134	135
QP_ac	700	713	725	737	749	761	773	785
index	136	137	138	139	140	141	142	143
QP_ac	797	809	825	841	857	873	889	905
index	144	145	146	147	148	149	150	151
QP_ac	922	938	954	970	986	1 002	1 018	1 038
index	152	153	154	155	156	157	158	159
QP_ac	1 058	1 078	1 098	1 118	1 138	1 158	1 178	1 198
index	160	161	162	163	164	165	166	167
QP_ac	1 218	1 242	1 266	1 290	1 314	1 338	1 362	1 386
index	168	169	170	171	172	173	174	175
QP_ac	1 411	1 435	1 463	1 491	1 519	1 547	1 575	1 603

表 54 (续)

index	176	177	178	179	180	181	182	183
QP_ac	1 631	1 663	1 695	1 727	1 759	1 791	1 823	1 859
index	184	185	186	187	188	189	190	191
QP_ac	1 895	1 931	1 967	2 003	2 039	2 079	2 119	2 159
index	192	193	194	195	196	197	198	199
QP_ac	2 199	2 239	2 283	2 327	2 371	2 415	2 459	2 507
index	200	201	202	203	204	205	206	207
QP_ac	2 555	2 603	2 651	2 703	2 755	2 807	2 859	2 915
index	208	209	210	211	212	213	214	215
QP_ac	2 971	3 027	3 083	3 143	3 203	3 263	3 327	3 391
index	216	217	218	219	220	221	222	223
QP_ac	3 455	3 523	3 591	3 659	3 731	3 803	3 876	3 952
index	224	225	226	227	228	229	230	231
QP_ac	4 028	4 104	4 184	4 264	4 348	4 432	4 516	4 604
index	232	233	234	235	236	237	238	239
QP_ac	4 692	4 784	4 876	4 972	5 068	5 168	5 268	5 372
index	240	241	242	243	244	245	246	247
QP_ac	5 476	5 584	5 692	5 804	5 916	6 032	6 148	6 268
index	248	249	250	251	252	253	254	255
QP_ac	6 388	6 512	6 640	6 768	6 900	7 036	7 172	7 312

表 55 BitDepth=12 时, QP_ac 与 index=QPIndex + ac_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_ac	4	13	19	27	35	44	54	64
index	8	9	10	11	12	13	14	15
QP_ac	75	87	99	112	126	139	154	168
index	16	17	18	19	20	21	22	23
QP_ac	183	199	214	230	247	263	280	297
index	24	25	26	27	28	29	30	31
QP_ac	314	331	349	366	384	402	420	438
index	32	33	34	35	36	37	38	39
QP_ac	456	475	493	511	530	548	567	586

表 55 (续)

index	40	41	42	43	44	45	46	47
QP_ac	604	623	642	660	679	698	716	735
index	48	49	50	51	52	53	54	55
QP_ac	753	772	791	809	828	846	865	884
index	56	57	58	59	60	61	62	63
QP_ac	902	920	939	957	976	994	1 012	1 030
index	64	65	66	67	68	69	70	71
QP_ac	1 049	1 067	1 085	1 103	1 121	1 139	1 157	1 175
index	72	73	74	75	76	77	78	79
QP_ac	1 193	1 211	1 229	1 246	1 264	1 282	1 299	1 317
index	80	81	82	83	84	85	86	87
QP_ac	1 335	1 352	1 370	1 387	1 405	1 422	1 440	1 457
index	88	89	90	91	92	93	94	95
QP_ac	1 474	1 491	1 509	1 526	1 543	1 560	1 577	1 595
index	96	97	98	99	100	101	102	103
QP_ac	1 627	1 660	1 693	1 725	1 758	1 791	1 824	1 856
index	104	105	106	107	108	109	110	111
QP_ac	1 889	1 922	1 954	1 987	2 020	2 052	2 085	2 118
index	112	113	114	115	116	117	118	119
QP_ac	2 150	2 183	2 216	2 248	2 281	2 313	2 346	2 378
index	120	121	122	123	124	125	126	127
QP_ac	2 411	2 459	2 508	2 556	2 605	2 653	2 701	2 750
index	128	129	130	131	132	133	134	135
QP_ac	2 798	2 847	2 895	2 943	2 992	3 040	3 088	3 137
index	136	137	138	139	140	141	142	143
QP_ac	3 185	3 234 SAG	3 298	3 362	3 426	3 491	3 555	3 619
index	144	145	146	147	148	149	150	151
QP_ac	3 684	3 748	3 812	3 876	3 941	4 005	4 069	4 149
index	152	153	154	155	156	157	158	159
QP_ac	4 230	4 310	4 390	4 470	4 550	4 631	4 711	4 791
index	160	161	162	163	164	165	166	167
QP_ac	4 871	4 967	5 064	5 160	5 256	5 352	5 448	5 544

表 55 (续)

index	168	169	170	171	172	173	174	175
QP_ac	5 641	5 737	5 849	5 961	6 073	6 185	6 297	6 410
index	176	177	178	179	180	181	182	183
QP_ac	6 522	6 650	6 778	6 906	7 034	7 162	7 290	7 435
index	184	185	186	187	188	189	190	191
QP_ac	7 579	7 723	7 867	8 011	8 155	8 315	8 475	8 635
index	192	193	194	195	196	197	198	199
QP_ac	8 795	8 956	9 132	9 308	9 484	9 660	9 836	10 028
index	200	201	202	203	204	205	206	207
QP_ac	10 220	10 412	10 604	10 812	11 020	11 228	11 437	11 661
index	208	209	210	211	212	213	214	215
QP_ac	11 885	12 109	12 333	12 573	12 813	13 053	13 309	13 565
index	216	217	218	219	220	221	222	223
QP_ac	13 821	14 093	14 365	14 637	14 925	15 213	15 502	15 806
index	224	225	226	227	228	229	230	231
QP_ac	16 110	16 414	16 734	17 054	17 390	17 726	18 062	18 414
index	232	233	234	235	236	237	238	239
QP_ac	18 766	19 134	19 502	19 886	20 270	20 670	21 070	21 486
index	240	241	242	243	244	245	246	247
QP_ac	21 902	22 334	22 766	23 214	23 662	24 126	24 590	25 070
index	248	249	250	251	252	253	254	255
QP_ac	25 551	26 047	26 559	27 071	27 599	28 143	28 687	29 247

5.3.6.4.2 反量化

根据量化参数 QP(QPY_dc 或 QPY_ac 或 QPC_dc 或 QPC_ac), 将 4×4 、 8×8 、 16×16 、 32×32 量化系数矩阵 QuantCoeffMatrix 转换为 4×4 、 8×8 、 16×16 、 32×32 变换系数矩阵 CoeffMatrix。

变换系数矩阵 CoeffMatrix 的元素 w_{ji} 由量化系数矩阵 QuantCoeffMatrix 的元素 val_{ji} 经下式得到:

$$w_{ji} = (val_{ji} \times QP) \gg dq_shift,$$

其中 tx_size=4 时, $dq_shift=0$, $i, j=0 \dots 3$;

tx_size=8 时, $dq_shift=0$, $i, j=0 \dots 7$;

tx_size=16 时, $dq_shift=0$, $i, j=0 \dots 15$;

tx_size=32 时, $dq_shift=1$, $i, j=0 \dots 31$ 。

5.3.6.5 反变换

5.3.6.5.1 4×4 反变换

4×4 反变换分为行反变换和列反变换, 行/列反变换均可为 dct 的反变换或者 adst 的反变换。其组合形式有: {idct_idct}、{idct_iadst}、{iadst_idct}、{iadst_iadst}。

其中, 4×4 IADST 变换核矩阵为:

$$DST_4 = \begin{bmatrix} 5283 & 9929 & 13377 & 15212 \\ 13377 & 13377 & 0 & -13377 \\ 15212 & -5238 & -13377 & 9929 \\ 9929 & -15212 & 13377 & -5283 \end{bmatrix}$$

4×4 IDCT 变换核矩阵为:

$$DCT_4 = \begin{bmatrix} 11585 & 11585 & 11585 & 11585 \\ 15137 & 6270 & -6270 & -15137 \\ 11585 & -11585 & -11585 & 11585 \\ 6270 & -15137 & 15137 & -6270 \end{bmatrix}$$

4×4 反变换步骤如下:

对变换系数矩阵进行水平反变换, 如果反变换形式为 iadst, $T_4 = DST_4$; 如果反变换形式为 idct, 则 $T_4 = DCT_4$;

$$H' = (\text{CoeffMatrix} \times T_4^T + 2^{13}) \gg 14$$

其中, T_4 为 4×4 反变换矩阵, T_4^T 为 T_4 的转置矩阵, H 表示水平反变换后的中间结果。对矩阵 H 进行垂直反变换, 如果反变换形式为 iadst, 则 $T_4 = DST_4$; 如果反变换形式为 idct, 则 $T_4 = DCT_4$;

$$H = (T_4 \times H' + 2^{13}) \gg 14$$

其中, H 表示反变换后的 4×4 矩阵。从符合本标准的比特流中解码得到的 H 矩阵元素取值范围应为 $-2^{4+\text{BitDepth}} \sim (2^{4+\text{BitDepth}} - 1)$ 。

残差样点矩阵 ResidueMatrix 的元素 r_{ji} 计算如下:

$$r_{ji} = (h_{ji} + 2^3) \gg 4 \quad \dots \quad (i, j = 0 \dots 3)$$

其中, h_{ji} 为 H 矩阵的元素。

5.3.6.5.2 8×8 反变换

8×8 反变换分为行反变换和列反变换分别进行, 行/列反变换都可以为 dct 的反变换或者 adst 的反变换。其组合形式有: {idct_idct}、{idct_iadst}、{iadst_idct}、{iadst_iadst}。

其中 8×8 IADST 变换核矩阵为:

$$DST_8 = \begin{bmatrix} 1606 & 4756 & 7723 & 10394 & 12665 & 14449 & 15679 & 16305 \\ 4756 & 12665 & 16305 & 14449 & 7723 & -1606 & -10394 & -15679 \\ 7723 & 16305 & 10394 & -4756 & -15679 & -12665 & 1606 & 14449 \\ 10394 & 14449 & -4756 & -16305 & -1606 & 15679 & 7723 & -12665 \\ 12665 & 7723 & -15679 & -1606 & 16305 & -4756 & -14449 & 10394 \\ 14449 & -1606 & -12665 & 15679 & -4756 & -10394 & 16305 & -7723 \\ 15679 & -10394 & 1606 & 7723 & -14449 & 16305 & -12665 & 4756 \\ 16305 & -15679 & 14449 & -12665 & 10394 & -7723 & 4756 & -1606 \end{bmatrix}$$

8×8 IDCT 变换核矩阵为:

$$DCT_8 = \begin{bmatrix} 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 \\ 16069 & 13623 & 9102 & 3196 & -3196 & -9102 & -13623 & -16069 \\ 15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137 \\ 13623 & -3196 & -16069 & -9102 & 9102 & 16069 & 3196 & -13623 \\ 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 \\ 9102 & -16069 & 3196 & 13623 & -13623 & -3196 & 16069 & -9102 \\ 6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270 \\ 3196 & -9102 & 13623 & -16069 & 16069 & -13623 & 9102 & -3196 \end{bmatrix}$$

8×8 反变换步骤如下：

- a) 对变换系数矩阵进行水平反变换,如果反变换形式为 iadst, $T_8 = DST_8$; 如果反变换形式为 idct, 则 $T_8 = DCT_8$;

$$H' = (\text{CoeffMatrix} \times T_8^T + 2^{13}) \gg 14$$

其中, T_8 为 8×8 反变换矩阵, T_8^T 为 T_8 的转置矩阵, H' 表示水平反变换后的中间结果。

- b) 对矩阵 H' 进行垂直反变换,如果反变换形式为 iadst, 则 $T_8 = DST_8$; 如果反变换形式为 idct, 则 $T_8 = DCT_8$;

$$H = (T_8 \times H' + 2^{13}) \gg 14$$

其中, H 表示反变换后的 8×8 矩阵。从符合本标准的比特流中解码得到的 H 矩阵元素取值范围应为 $-2^{5+\text{BitDepth}} \sim (2^{5+\text{BitDepth}} - 1)$ 。

- c) 残差样点矩阵 ResidueMatrix 的元素 r_{ji} 计算如下:

$$r_{ji} = (h_{ji} + 2^4) \gg 5 \quad \dots \dots \dots \quad (i, j = 0 \dots 7)$$

其中, h_{ji} 为 H 矩阵的元素。

5.3.6.5.3 16×16 反变换

16×16 反变换分为行反变换和列反变换分别进行,行/列反变换均可为 dct 的反变换或者 adst 的反变换。其组合形式有:{idct_idct}、{idct_iadst}、{iadst_idct}、{iadst_iadst}。

其中 16×16IADST 变换核矩阵为:

$$DST_{16} = \begin{bmatrix} 804 & 2404 & 3981 & 5520 & 7005 & 8423 & 9760 & 11003 & 12140 & 13160 & 14053 & 14811 & 15426 & 15893 & 16207 & 16364 \\ 2404 & 7005 & 11003 & 14053 & 15893 & 16364 & 15426 & 13160 & 9760 & 5520 & 804 & -3981 & -8423 & -12140 & -14811 & -16207 \\ 3981 & 11003 & 15426 & 16207 & 13160 & 7005 & -804 & -8423 & -14053 & -16364 & -14811 & -9760 & -2404 & 5520 & 12140 & 15893 \\ 5520 & 14053 & 16207 & 11003 & 804 & -9760 & -15893 & -14811 & -7005 & 3981 & 13160 & 16364 & 12140 & 2404 & -8423 & -15426 \\ 7005 & 15893 & 13160 & 804 & -12140 & -16207 & -8423 & 5520 & 15426 & 14053 & 2404 & -11003 & -16364 & -9760 & 3981 & 14811 \\ 8423 & 16364 & 7005 & -9760 & -16207 & -5520 & 11003 & 15893 & 3981 & -12140 & -15426 & -2404 & 13160 & 14811 & 804 & -14053 \\ 9760 & 15426 & -804 & -15893 & -8423 & 11003 & 14811 & -2404 & -16207 & -7005 & 12140 & 14053 & -3981 & -16364 & -5520 & 13160 \\ 11003 & 13160 & -8423 & -14811 & 5520 & 15893 & -2404 & -16364 & -804 & 16207 & 3981 & -15426 & -7005 & 14053 & 9760 & -12140 \\ 12140 & 9760 & -14053 & -7005 & 15426 & 3981 & -16207 & -804 & 16364 & -2404 & -15893 & 5520 & 14811 & -8423 & -13160 & 11003 \\ 13160 & 5520 & -16364 & 3981 & 14053 & -12140 & -7005 & 16207 & -2404 & -14811 & 11003 & 8423 & -15893 & 804 & 15426 & -9760 \\ 14053 & 804 & -14811 & 13160 & 2404 & -15426 & 12140 & 3981 & -15893 & 11003 & 5520 & -16207 & 9760 & 7005 & -16364 & 8423 \\ 14811 & -3981 & -9760 & 16364 & -11003 & -2404 & 14053 & -15426 & 5520 & 8423 & -16207 & 12140 & 804 & -13160 & 15893 & -7005 \\ 15426 & -8423 & -2404 & 12140 & -16364 & 13160 & -3981 & -7005 & 14811 & -15893 & 9760 & 804 & -11003 & 16207 & -14053 & 5520 \\ 15893 & -12140 & 5520 & 2404 & -9760 & 14811 & -16364 & 14053 & -8423 & 804 & 7005 & -13160 & 16207 & -15426 & 11003 & -3981 \\ 16207 & -14811 & 12140 & -8423 & 3981 & 804 & -5520 & 9760 & -13160 & 15426 & -16364 & 15893 & -14053 & 11003 & -7005 & 2404 \\ 16364 & -16207 & 15893 & -15426 & 14811 & -14053 & 13160 & -12140 & 11003 & -9760 & 8423 & -7005 & 5520 & -3981 & 2404 & -804 \end{bmatrix}$$

16×16idct 变换核矩阵为:

$DCT_{16} =$	[11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585]
	16305 15679 14449 12665 10394 7723 4756 1606 -1606 -4756 -7723 -10394 -12665 -14449 -15679 -16305
	16069 13623 9102 3196 -3196 -9102 -13623 -16069 -16069 -13623 -9102 -3196 3196 9102 13623 16069
	15679 10394 1606 -7723 -14449 -16305 -12665 -4756 4756 12665 16305 14449 7723 -1606 -10394 -15679
	15137 6270 -6270 -15137 -15137 -6270 6270 15137 15137 6270 -6270 -15137 -15137 -6270 6270 15137
	14449 1606 -12665 -15679 -4756 10394 16305 7723 -7723 -16305 -10394 4756 15679 12665 -1606 -14449
	13623 -3196 -16069 -9102 9102 16069 3196 -13623 -13623 3196 16069 9102 -9102 -16069 -3196 13623
	12665 -7723 -15679 1606 16305 4756 -14449 -10394 10394 14449 -4756 -16305 -1606 15679 7723 -12665
	11585 -11585 -11585 11585 11585 -11585 -11585 11585 11585 -11585 -11585 11585 -11585 -11585 -11585 11585
	10394 -14449 -4756 16305 -1606 -15679 7723 12665 -12665 -7723 15679 1606 -16305 4756 14449 -10394
	9102 -16069 3196 13623 -13623 -3196 16069 -9102 -9102 16069 -3196 -13623 13623 3196 -16069 9102
	7723 -16305 10394 4756 -15679 12665 1606 -14449 14449 -1606 -12665 15679 -4756 -10394 16305 -7723
	6270 -15137 15137 -6270 -6270 15137 -15137 6270 6270 -15137 15137 -6270 -6270 15137 -15137 6270
	4756 -12665 16305 -14449 7723 1606 -10394 15679 -15679 10394 -1606 -7723 14449 -16305 12665 -4756
	3196 -9102 13623 -16069 16069 -13623 9102 -3196 -3196 9102 -13623 16069 -16069 13623 -9102 3196
	1606 -4756 7723 -10394 12665 -14449 15679 -16305 16305 -15679 14449 -12665 10394 -7723 4756 -1606

16×16 反变换步骤如下：

- a) 对变换系数矩阵进行水平反变换,如果反变换形式为 iadst,则 $T_{16} = DST_{16}$; 如果反变换形式为 idct,则 $T_{16} = DCT_{16}$:

$$H' = (\text{CoeffMatrix} \times T_{16}^T + 2^{13}) \gg 14$$

其中, T_{16} 为 16×16 反变换矩阵, T_{16}^T 为 T_{16} 的转置矩阵, H' 表示水平反变换后的中间结果。

- b) 对矩阵 H' 进行垂直反变换,如果反变换形式为 iadst,则 $T_{16} = DST_{16}$; 如果反变换形式为 idct,则 $T_{16} = DCT_{16}$:

$$H = (T_{16} \times H' + 2^{13}) \gg 14$$

其中, H 表示反变换后的 16×16 矩阵。从符合本标准的比特流中解码得到的 H 矩阵元素取值范围应为 $-2^{6+\text{BitDepth}} \sim (2^{6+\text{BitDepth}} - 1)$ 。

- c) 残差样点矩阵 ResidueMatrix 的元素 r_{ji} 计算如下:

$$r_{ji} = (h_{ji} + 2^5) \gg 6 \quad (i, j = 0 \dots 15)$$

5.3.6.5.4 32×32 反变换

32×32 反变换分为行反变换和列反变换分别进行,行/列反变换都只能为 dct 的反变换

32×32IDCT 变换核矩阵为

$$DCT_{32} = [DCT_{L16 \times 32} \ DCT_{R16 \times 32}]$$

	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585	11585
	16364	16207	15893	15426	14811	14053	13160	12140	11003	9760	8423	7005	5520	3981	2404	804	
	16305	15679	14449	12665	10394	7723	4756	1606	-1606	-4756	-7723	-10394	-12665	-14449	-15679	-16305	
	16207	14811	12140	8423	3981	-804	-5520	-9760	-13160	-15426	-16364	-15893	-14053	-11003	-7005	-2404	
	16069	13623	9102	3196	-3196	-9102	-13623	-16069	-16069	-13623	-9102	-3196	3196	9102	13623	16069	
	15893	12140	5520	-2404	-9760	-14811	-16364	-14053	-8423	-804	7005	13160	16207	15426	11003	3981	
	15679	10394	1606	-7723	-14449	-16305	-12665	-4756	4756	12665	16305	14449	7723	-1606	-10394	-15679	
	15426	8423	-2404	-12140	-16364	-13160	-3981	7005	14811	15893	9760	-804	-11003	-16207	-14053	-5520	
	15137	6270	-6270	-15137	-15137	-6270	6270	15137	15137	6270	-6270	-15137	-15137	-6270	6270	15137	
	14811	3981	-9760	-16364	-11003	2404	14053	15426	5520	-8423	-16207	-12140	804	13160	15893	7005	
	14449	1606	-12665	-15679	-4756	10394	16305	7723	-7723	-16305	-10394	4756	15679	12665	-1606	-14449	
	14053	-804	-14811	-13160	2404	15426	12140	-3981	-15893	-11003	5520	16207	9760	-7005	-16364	-8423	
	13623	-3196	-16069	-9102	9102	16069	3196	-13623	-13623	3196	16069	9102	-9102	-16069	-3196	13623	
	13160	-5520	-16364	-3981	14053	12140	-7005	-16207	-2404	14811	11003	-8423	-15893	-804	15426	9760	
	12665	-7723	-15679	1606	16305	4756	-14449	-10394	10394	14449	-4756	-16305	-1606	15679	7723	-12665	
DCT _{L16×32} =	12140	-9760	-14053	7005	15426	-3981	-16207	804	16364	2404	-15893	-5520	14811	8423	-13160	-11003	
	11585	-11585	-11585	11585	11585	-11585	-11585	11585	11585	-11585	11585	-11585	-11585	-11585	-11585	11585	
	11003	-13160	-8423	14811	5520	-15893	-2404	16364	-804	-16207	3981	15426	-7005	-14053	9760	12140	
	10394	-14449	-4756	16305	-1606	-15679	7723	12665	-12665	-7723	15679	1606	-16305	4756	14449	-10394	
	9760	-15426	-804	15893	-8423	-11003	14811	2404	-16207	7005	12140	-14053	-3981	16364	-5520	-13160	
	9102	-16069	3196	13623	-13623	-3196	16069	-9102	-9102	16069	-3196	-13623	13623	3196	-16069	9102	
	8423	-16364	7005	9760	-16207	5520	11003	-15893	3981	12140	-15426	2404	13160	-14811	804	14053	
	7723	-16305	10394	4756	-15679	12665	1606	-14449	14449	-1606	-12665	15679	-4756	-10394	16305	-7723	
	7005	-15893	13160	-804	-12140	16207	-8423	-5520	15426	-14053	2404	11003	-16364	9760	3981	-14811	
	6270	-15137	15137	-6270	-6270	15137	-15137	6270	6270	-15137	15137	-6270	-6270	15137	-15137	6270	
	5520	-14053	16207	-11003	804	9760	-15893	14811	-7005	-3981	13160	-16364	12140	-2404	-8423	15426	
	4756	-12665	16305	-14449	7723	1606	-10394	15679	-15679	10394	-1606	-7723	14449	-16305	12665	-4756	
	3981	-11003	15426	-16207	13160	-7005	-804	8423	-14053	16364	-14811	9760	-2404	-5520	12140	-15893	
	3196	-9102	13623	-16069	16069	-13623	9102	-3196	-3196	9102	-13623	16069	-16069	13623	-9102	3196	
	2404	-7005	11003	-14053	15893	-16364	15426	-13160	9760	-5520	804	3981	-8423	12140	-14811	16207	
	1606	-4756	7723	-10394	12665	-14449	15679	-16305	16305	-15679	14449	-12665	10394	-7723	4756	-1606	
	804	-2404	3981	-5520	7005	-8423	9760	-11003	12140	-13160	14053	-14811	15426	-15893	16207	-16364	

DCT _{R16×32} =	[11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 11585 - 804 - 2404 - 3981 - 5520 - 7005 - 8423 - 9760 - 11003 - 12140 - 13160 - 14053 - 14811 - 15426 - 15893 - 16207 - 16364 - 16305 - 15679 - 14449 - 12665 - 10394 - 7723 - 4756 - 1606 1606 4756 7723 10394 12665 14449 15679 16305 2404 7005 11003 14053 15893 16364 15426 13160 9760 5520 804 - 3981 - 8423 - 12140 - 14811 - 16207 16069 13623 9102 3196 - 3196 - 9102 - 13623 - 16069 - 16069 - 13623 - 9102 - 3196 3196 9102 13623 16069 - 3981 - 11003 - 15426 - 16207 - 13160 - 7005 804 8423 14053 16364 14811 9760 2404 - 5520 - 12140 - 15893 - 15679 - 10394 - 1606 7723 14449 16305 12665 4756 - 4756 - 12665 - 16305 - 14449 - 7723 1606 10394 15679 5520 14053 16207 11003 804 - 9760 - 16207 - 5520 11003 - 10394 - 14449 4756 16305 1606 - 15679 - 7723 12665 15137 6270 - 6270 - 15137 - 15137 - 6270 6270 15137 15137 6270 - 6270 - 15137 - 15137 - 6270 6270 15137 - 7005 - 15893 - 13160 - 804 12140 16207 8123 - 5520 - 15426 - 14053 - 2404 11003 16364 9760 - 3981 - 14811 - 14449 - 1606 12665 15679 4756 - 10394 - 16305 - 7723 7723 16305 10394 - 4756 - 15679 - 12665 1606 14449 8423 16364 7005 - 9760 - 16207 - 5520 11003 15893 3981 - 12140 - 15426 - 2404 13160 14811 804 - 14053 13623 - 3196 - 16069 - 9102 9102 16069 3196 - 13623 - 13623 3196 16069 9102 - 9102 - 16069 - 3196 13623 - 9760 - 15426 804 15893 8423 - 11003 - 14811 2404 16207 7005 - 12140 - 14053 3981 16364 5520 - 13160 - 12665 7723 15679 - 1606 - 16305 - 4756 14449 10394 - 10394 - 14449 4756 16305 1606 - 15679 - 7723 12665 11003 13160 - 8423 - 11811 5520 15893 - 2404 - 16364 - 804 16207 3981 - 15426 - 7005 14053 9760 - 12140 11585 - 11585 - 11585 11585 11585 - 11585 - 11585 11585 - 11585 - 11585 11585 - 11585 - 11585 - 11585 - 12140 - 9760 14053 7005 - 15426 - 3981 16207 804 - 16364 2404 15893 - 5520 - 14811 8423 13160 - 11003 - 10394 14449 4756 - 16305 1606 15679 - 7723 - 12665 12665 7723 - 15679 - 1606 16305 - 4756 - 14449 10394 13160 5520 - 16364 3981 14053 - 12140 - 7005 16207 - 2404 - 14811 11003 8423 - 15893 804 15426 - 9760 9102 - 16069 3196 13623 - 13623 - 3196 16069 - 9102 - 9102 16069 - 3196 - 13623 13623 3196 - 16069 9102 - 14053 - 804 14811 - 13160 - 2404 15426 - 12140 - 3981 15893 - 11003 - 5520 16207 - 9760 - 7005 16364 - 8423 - 7723 16305 - 10394 - 4756 15679 - 12665 - 1606 14449 - 14449 1606 12665 - 15679 4756 10394 - 16305 7723 14811 - 3981 - 9760 16364 - 11003 - 2404 14053 - 15426 5520 8423 - 16207 12140 804 - 13160 15893 - 7005 6270 - 15137 15137 - 6270 - 6270 15137 - 15137 6270 6270 - 15137 15137 - 6270 - 6270 15137 - 15137 6270 - 15426 8423 2404 - 12140 16364 - 13160 3981 7005 - 14811 15893 - 9760 - 804 11003 - 16207 14053 - 5520 - 4756 12665 - 16305 14449 - 7723 - 1606 10394 - 15679 15679 - 10394 1606 7723 - 14449 16305 - 12665 4756 15893 - 12140 5520 2404 - 9760 14811 - 16364 14053 - 8423 804 7005 - 13160 16207 - 15426 11003 - 3981 3196 - 9102 13623 - 16069 16069 - 13623 9102 - 3196 - 3196 9102 - 13623 16069 - 16069 13623 - 9102 3196 - 16207 14811 - 12140 8423 - 3981 - 804 5520 - 9760 13160 - 15426 16364 - 15893 14053 - 11003 7005 - 2404 - 1606 4756 - 7723 10394 - 12665 14449 - 15679 16305 - 16305 15679 - 14449 12665 - 10394 7723 - 4756 1606 16364 - 16207 15893 - 15426 14811 - 14053 13160 - 12140 11003 - 9760 8423 - 7005 5520 - 3981 2404 - 804]
-------------------------	--

32×32 反变换步骤如下：

- a) 对变换系数矩阵进行水平反变换, $T_{32} = DCT_{32}$:

$$H' = (\text{CoeffMatrix} \times T_{32}^T + 2^{13}) \gg 14$$

其中, T_{32} 为 32×32 反变换矩阵, T_{32}^T 为 T_{32} 的转置矩阵, H' 表示水平反变换后的中间结果。

- b) 对矩阵 H' 进行垂直反变换, $T_{32} = DCT_{32}$:

$$H = (T_{32} \times H' + 2^{13}) \gg 14$$

其中, H 表示反变换后的 32×32 矩阵。从符合本标准的比特流中解码得到的 H 矩阵元素取值范围应为 $-2^{6+\text{BitDepth}} \sim (2^{6+\text{BitDepth}} - 1)$ 。

- c) 残差样点矩阵 ResidueMatrix 的元素 r_{ji} 计算如下:

$$r_{ji} = (h_{ji} + 2^5) \gg 6 \quad (i, j = 0 \dots 31)$$

5.3.6.6 重建

本过程的输入为残差样值矩阵 ResidueMatrix 和预测样点矩阵 predMatrix, 输出为重建样点矩阵 RecMatrix。

如果当前块为 Intra 块类型, 重建样点矩阵 RecMatrix 计算如下:

$$\text{RecMatrix}[x, y] = \text{Clip1}(\text{predMatrix}[x, y] + \text{ResidueMatrix}[x, y])$$

如果当前块为 Inter 块类型, 采用 Single 预测模式, 重建样点矩阵 RecMatrix 计算如下:

$$\text{RecMatrix}[x, y] = \text{Clip1}(\text{predMatrix}[x, y] + \text{ResidueMatrix}[x, y])$$

如果当前块为 Inter 块类型,采用 Compound 预测模式,并且有两帧参考图像,重建样点矩阵 RecMatrix 计算如下:

$$\text{RecMatrix}[x, y] = \text{Clip1}(((\text{predMatrix0}[x, y] + \text{predMatrix1}[x, y] + 1) \gg 1) + \text{ResidueMatrix}[x, y])$$

5.3.7 去块效应滤波过程

5.3.7.1 进行滤波的边界及滤波的顺序

如果 filter_level 大于 0,在位于去块效应滤波过程之前的图像重建过程完成以后,对整幅重建图像调用去块效应滤波过程。去块效应滤波过程以编码单元为单位对图像中的所有的编码单元按照光栅扫描的顺序进行。否则将不进行调用去块效应滤波过程。

在一个编码单元内,对亮度和色度分别做去块效应滤波,去块效应滤波的单位是滤波块。对于每个树形编码单元以及每个分量按滤波块尺寸进行划分进行滤波,亮度滤波块的尺寸是 8×8 ,色度滤波块的尺寸是 8×8 。每个滤波块包括一条垂直边界和一条水平边界。先滤波纵向的边界,从编码单元的左侧的边界开始,按照从左到右的顺序进行处理,然后滤波横向的边界,从编码单元的上部边界开始,按照从上到下的顺序进行处理。

图 15 表示图像的树形编码单元尺寸为 64×64 ,滤波块尺寸为 8×8 。如果需要滤波,则根据 5.3.7.2 计算边界滤波参数,根据 5.3.7.3 采用对应的方式进行滤波。

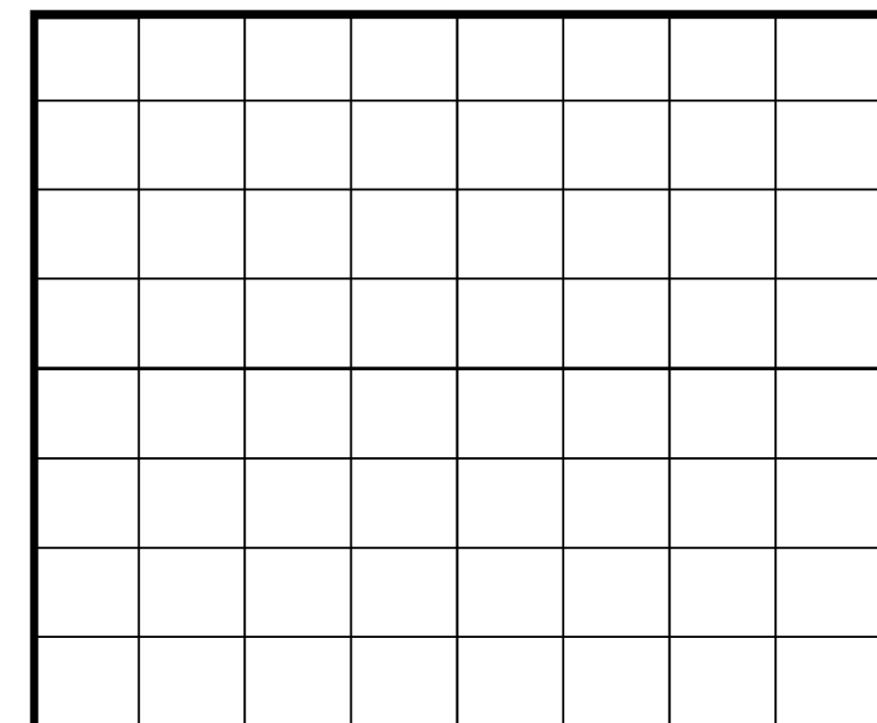


图 15 块边界去块滤波示意图

注 1: 因为在解码当前编码单元时,下边与右边的编码单元数据没有解码出来,因此先做上一行编码单元的滤波。

注 2: 帧内预测使用去块效应滤波前的重建图像样点值。

注 3: 当前滤波块的垂直边界的滤波过程中修改的样值作为水平边界滤波过程的输入。

满足以下条件之一的边界不需要滤波:

- a) 待滤波边界是图像边界或 Tile 边界;

- b) 待滤波边界是 skip_flag 等于 1 的预测单元的内部边界；
- c) 待滤波边界既不是编码单元的边界，又不是预测单元的边界也不是变换块的边界。

5.3.7.2 块边界滤波参数的推导过程

块边界滤波参数的推导过程如下：

- a) 根据图像参数集中的 filter_level 计算得到每个编码块单元在不同情况下的 cu_filter_level 值：

首先计算每个 segment 对应的 delta，不同的 segment_id、不同的参考帧、不同的块模式对应不同的 delta 值。通过在图像参数集中 filter_level 的基础上，增加 delta 最终获得数组 lvl_lookup [8][6][2] 的值。具体实现如下：

```

scale=1 << (filter_level >> 5)
for (seg_id=0; seg_id<8; seg_id++){
    lvl_seg=filter_level
    if(feature_enable[seg_id][1]){
        data=features_data[seg_id][1]
        lvl_seg=clip3(0,63,seg_abs_delta==1 ? data:filter_level+data)
    }
    if(! lf_delta_enable)
        lvl_lookup[seg_id][6][2]={lvl_seg,lvl_seg,...}
    else{
        intra_lvl=lvl_seg+ref_deltas[0]*scale
        lvl_lookup[seg_id][0][0]=clip3(0,63,intra_lvl)
        for (ref=1; ref<6; ++ref) {
            for(mode=0; mode<2; ++mode) {
                inter_lvl=lvl_seg+ref_deltas[ref]*scale+mode_deltas[mode]*scale
                lvl_lookup[seg_id][ref][mode]=clip3(0,63,inter_lvl)
            }
        }
    }
}

```

根据当前编码块的对应的 segment_id、参考帧类型 ref_frame、预测模式 mode 作为索引即可以计算得到当前编码块的 cu_filter_level。当块的预测模式为帧内预测时，mode 等于 0；当预测模式为帧间预测时，如果当前编码块的 mv_mode 为 ZERO_MV，则 mode 等于 0，否则 mode 等于 1。cu_filter_level 等于 lvl_lookup[segment_id][ref_frame][mode]。

- b) 根据 sharpness_level 计算不同的 cu_filter_level 对应的阈值参数 limit、blimit、thresh 的值：

根据 sharpness_level 的取值阈值，参数数组 limit[64]、blimit[64]、thresh[64] 的计算过程如下所示：

```

for(lvl=0; lvl<=63; lvl++){
    block_inside_limit=lvl >> ((sharpness_level > 0)+(sharpness_level > 4))
    if((sharpness_lvl>0) && (block_inside_limit > (9-sharpness_level)))
        block_inside_limit=(9-sharpness_level);
    if(block_inside_limit < 1)
        block_inside_limit=1
}

```

```

limit[lvl]=block_inside_limit
blimit[lvl] = 2×(lvl+2)+block_inside_limit
thresh[lvl] = (lvl >> 4)
}

```

根据当前编码块的 cu_filter_level 作为索引即可得到当前编码块的滤波阈值参数 limit、blimit 和 thresh。

c) 计算 filterSize:

如果该边界所属滤波块的 tx_size 为 TX_4x4，并且该边界不位于按 32×32 块划分的边界，则 filterSize 等于 0；否则 filterSize 等于 1。

5.3.7.3 块边界滤波过程

图 16 表示在水平或垂直边界两侧的 8 个样点，分别属于块 p 和块 q，边界用黑色粗线表示。

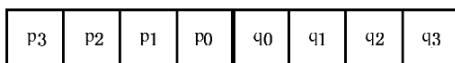


图 16 滤波边界(垂直或水平)两侧的样点

hev 的计算过程如下：

```

hev=0
threshBd=thresh << ( BitDepth - 8 )
hev |= (abs( p1-p0 )>threshBd) × -1
hev |= (abs( q1-q0 )>threshBd) × -1

```

mask 的计算过程如下：

```

limitBd=limit << ( BitDepth - 8 )
blimitBd=blimit<<( BitDepth - 8 )
mask=0
mask |= (abs(p3-p2)>blimitBd) × -1
mask |= (abs(p2-p1)>blimitBd) × -1
mask |= (abs(p1-p0)>blimitBd) × -1
mask |= (abs(q1-q0)>blimitBd) × -1
mask |= (abs(q2-q1)>blimitBd) × -1
mask |= (abs(q3-q2)>blimitBd) × -1
mask |= ((abs(p0-q0)<<1)+(abs(p1-q1))>>1)>blimitBd) × -1
mask=~ mask

```

flat 的计算过程如下：

如果 filtersize 等于 0，则 flat 等于 0，否则按如下过程计算 flat：

```

thresholdBd=1 << (BitDepth-8)
flat |= (abs(p1-p0)>thresholdBd) × -1
flat |= (abs(q1-q0)>thresholdBd) × -1
flat |= (abs(p2-p0)>thresholdBd) × -1
flat |= (abs(q2-q0)>thresholdBd) × -1
flat |= (abs(p3-p0)>thresholdBd) × -1
flat |= (abs(q3-q0)>thresholdBd) × -1
flat=~ flat

```



根据 flat 和 mask 的取值按如下方法执行滤波：

——如果 flat 与 mask 都不为 0，则：

```
op2=ROUND_POWER_OF_TWO(p3+p3+p3+2×p2+p1+p0+q0,3)
op1=ROUND_POWER_OF_TWO(p3+p3+p2+2×p1+p0+q0+q1,3)
op0=ROUND_POWER_OF_TWO(p3+p2+p1+2×p0+q0+q1+q2,3)
oq0=ROUND_POWER_OF_TWO(p2+p1+p0+2×q0+q1+q2+q3,3)
oq1=ROUND_POWER_OF_TWO(p1+p0+q0+2×q1+q2+q3+q3,3)
oq2=ROUND_POWER_OF_TWO(p0+q0+q1+2×q2+q3+q3+q3,3)
```

——否则，令 shift=BitDepth-8，有：

```
ps1=p1-(0x80 << shift)
ps0=p0-(0x80 << shift)
qs0=q0-(0x80 << shift)
qs1=q1-(0x80 << shift)
filter=Clip3((-128)<<shift,(128<<shift)-1,(ps1-qs1)) & hevSAC
filter=Clip3((-128)<<shift,(128<<shift)-1,(filter+3×(qs0-ps0))) & mask
filter1=Clip3((-128)<<shift,(128<<shift)-1,(filter+4)) >> 3
filter2=Clip3((-128)<<shift,(128<<shift)-1,(filter+3)) >> 3
filter=ROUND_POWER_OF_TWO(filter1,1) & (~hev)
oq0=Clip3((-128)<<shift,(128<<shift)-1,qs0-filter1)+(0x80 << shift)
op0=Clip3((-128)<<shift,(128<<shift)-1,ps0+filter2)+(0x80 << shift)
oq1=Clip3((-128)<<shift,(128<<shift)-1,qs1-filter)+(0x80 << shift)
op1=Clip3((-128)<<shift,(128<<shift)-1,ps1+filter)+(0x80 << shift)
```

其中 BitDepth 为图像的样点比特精度，op2~op0、oq0~oq2 对应 p2~p0、q0~q2 滤波后的样点值，ROUND_POWER_OF_TWO(value,n) 定义为 (((value)+(1 << ((n)-1))) >> (n))。

5.3.8 样点偏移补偿

5.3.8.1 概述

如果 sao_enable 等于 1，调用样点偏移补偿(SAO)过程。

如果 sao_component_on[Y],sao_component_on[U],sao_component_on[V] 的取值为 1，则对对应分量进行 SAO 操作。如果三个参数全为 0，则本帧数据跳过 SAO 操作。

根据 5.3.8.2 导出样点偏移补偿单元，根据 5.3.8.3 导出与当前样点偏移补偿单元对应的样点偏移补偿信息，然后按照 5.3.8.4 对当前样点偏移补偿单元内各个样点的各分量进行操作，得到偏移后样点值。

5.3.8.2 样点偏移补偿单元导出

SAO 操作以树形编码单元为单位，根据当前树形编码单元的区域按下列步骤得到当前样点偏移补偿单元的区域：

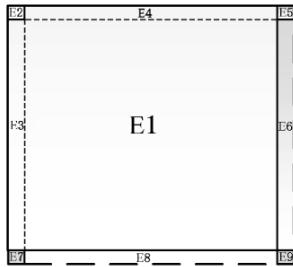


图 17 SAO 单元区域的导出

- 将当前树形编码单元所在样点区域的亮度区域与色度区域均(图 17 中黑色虚线所包含的矩形区域)向左移四个样点单位,再向上移四个样点单位,得到区域 S1(由图 17 中的 E1,E2,E3,E4 四个部分组成的矩形区域);
- 如果区域 S1 超出当前图像边界,则将超出边界部分去除(例:如当前树形编码单元为左边界 的树形编码单元,则去除图 17 中的 E2,E3 部分;如当前树形编码单元为上边界的树形编码单 元,则去除图 17 中的 E2,E4 部分,),得到区域 S2;否则令 S2 等于 S1;
- 如果当前树形编码单元 C 包含图像最右列的样点且不包含图像最下行的样点,则将区域 S2 的右边界向右扩展至图像的右边界(即将图 17 中的 E5(若存在),则 E6 纳入当前 SAO 区域), 得到区域 S3;
- 否则,如果当前树形编码单元 C 包含图像最下行的样点且不包含图像最右列的样点,则将区 域 S2 的下边界向下扩展至图像的边界(即将图 17 中的 E7(若存在),则 E8 纳入当前 SAO 区 域),得到区域 S3;
- 否则,如果当前树形编码单元 C 同时包含图像最右列的样点和最下行的样点,则将区域 S2 的 右边界向右扩展至图像的右边界后,再将新区域的下边界向下至图像的下边界,(将 E5(若存 在),E6,E7(若存在),E8,E9 均纳入当前 SAO 区域)得到区域 S3;
- 否则,令 S3 等于 S2;
- 将区域 S3 作为当前样点偏移补偿单元的区域。

5.3.8.3 样点偏移补偿信息导出

下述信息用于样点偏移补偿过程:参数融合标志 sao_merge_flag、融合类型 sao_merge_type、样点 偏移补偿合并模式 sao_merge_mode、样点偏移补偿模式 sao_mode[compIdx]、样点偏移补偿值 sao_offset[compIdx][j]、样点偏移补偿边缘模式类型 sao_edge_type[compIdx]、样点偏移补偿区间模式类 型的起始偏移子区间 sao_start_band[compIdx]等。

如果当前树形编码单元不是图像或者 Tile 的左边界树形编码单元, MergeLeftAvail 等于 1, 否则 MergeLeftAvail 等于 0。如果当前树形编码单元不是图片的上边界树形编码单元, MergeUpAvail 等于 1, 否则 MergeUpAvail 等于 0。

根据 MergeLeftAvail、MergeUpAvail、sao_merge_flag 及 sao_merge_type 的值查表 56 得到样点 偏移补偿合并模式 sao_merge_mode。

表 56 样点偏移补偿合并模式

MergeLeftAvail	MergeUpAvail	sao_merge_flag	sao_merge_type	sao_merge_mode
0	0	—	—	SAO_NON_MERGE
1	0	0	—	SAO_NON_MERGE
		1	—	SAO_MERGE_LEFT
0	1	0	—	SAO_NON_MERGE
		1	—	SAO_MERGE_UP
1	1	0	—	SAO_NON_MERGE
		1	1	SAO_MERGE_LEFT
			0	SAO_MERGE_UP

如果 sao_merge_mode 的值为‘SAO_MERGE_LEFT’，sao_mode[compIdx] (compIdx=0,1,2) 的值等于左侧相邻 CTU 的样点偏移补偿单元的 sao_mode[compIdx] 的值，当前 CTU 的 SAO 参数使用左侧相邻 CTU 的 SAO 参数；等于 0 表示当前 CTU 的 SAO 参数使用上方相邻 CTU 的 SAO 参数。

如果 sao_merge_mode 值为‘SAO_MERGE_UP’，sao_mode[compIdx] (compIdx=0,1,2) 的值等于上方相邻 CTU 样点偏移补偿单元的 sao_mode[compIdx] 的值，当前 CTU 的 SAO 参数使用上方相邻 CTU 的 SAO 参数。

如果 sao_merge_mode 的值为‘SAO_NON_MERGE’，则首先从码流中解析得到 sao_mode[compIdx] 的值，再根据 sao_mode[compIdx] 的值从码流中获取相应的其他信息：

- 如果 sao_mode[compIdx] 的值为‘SAO_BO’，则从码流中解析得到 sao_start_band[compIdx]，并有：

$$\text{sao_offset[compIdx][j]} = \text{sao_offset_abs[compIdx][j]} \times \text{Sign}(0 - \text{sao_offset_sign[compIdx][j]}) \quad (j=0 \sim 3)$$
- 如果 sao_mode[compIdx] 的值为‘SAO_EO’，则从码流中解析得到 sao_edge_offset[compIdx][j] (j=0~3) 和 sao_edge_type[compIdx]，并有：

$$\text{sao_offset[compIdx][j]} = \text{sao_edge_offset[compIdx][j]}$$
- 如果 sao_mode[compIdx] 的值为‘SAO_OFF’，则不需要从码流继续读取该分量的 SAO 信息。

5.3.8.4 SAO 操作步骤

5.3.8.4.1 SAO_BO 操作

如果样点偏移补偿单元的 sao_mode[i] 为‘SAO_BO’，则将样点取值划分为 32 个相等的区间，区间依次为 0~31，只对落入其中 4 个连续区间的样点值进行偏移补偿操作，每个区间对应一个偏移值，如图 18 所示。

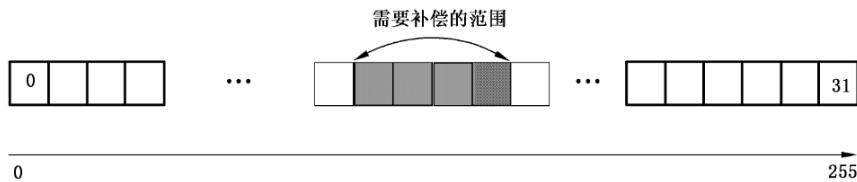


图 18 BO 示意图

其具体操作如下：

- a) 第一步,根据样点的 i 分量值大小所处的范围查表 57 得到该分量对应的 SaoOffset[i]。

表 57 样点取值范围对应的偏移值

样点取值范围	偏移值(SaoOffset)
SaoBandPos[i][0]<< shift ~ SaoBandPos[i][0]<< shift+(2^ shift-1)	sao_offset[i][0]
SaoBandPos[i][1]<< shift ~ SaoBandPos[i][1]<< shift+(2^ shift-1)	sao_offset[i][1]
SaoBandPos[i][2]<< shift ~ SaoBandPos[i][2]<< shift+(2^ shift-1)	sao_offset[i][2]
SaoBandPos[i][3]<< shift ~ SaoBandPos[i][3]<< shift+(2^ shift-1)	sao_offset[i][3]

表中：

$$\text{shift} = \text{BitDepth} - 5$$

$$\text{SaoBandPos}[i][0] = \text{sao_start_band}[i]$$

$$\text{SaoBandPos}[i][1] = (\text{sao_start_band}[i] + 1) \% 32$$

$$\text{SaoBandPos}[i][2] = (\text{sao_start_band}[i] + 2) \% 32$$

$$\text{SaoBandPos}[i][3] = (\text{sao_start_band}[i] + 3) \% 32$$

- b) 第二步,当前样点的 i 分量偏移后取值 y[i]计算如下:

$$y[i] = \text{Clip3}(0, (1 \ll \text{BitDepth}) - 1, x[i] + \text{SaoOffset}[i])$$

其中,x[i]是该样点的 i 分量在去块滤波后的值。

5.3.8.4.2 SAO_EO 操作

如果样点偏移补偿单元的 sao_mode[i]为‘SAO_EO’则进行以下操作：

- a) 第一步,根据 sao_edge_type[i]的值确定与当前样点 c 相邻的样点 a 和 b,如图 19 所示;如果 a 或 b 不在当前 Tile 内或者 a 或 b 不在当前图像内,则不改变样点 c 的值。

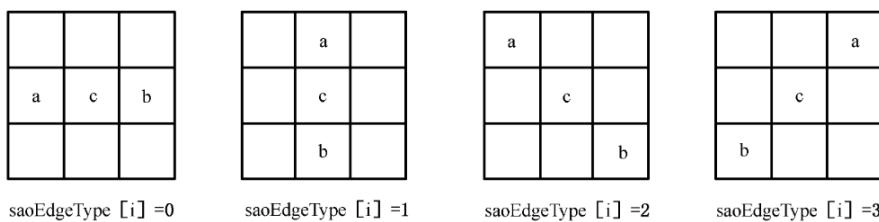


图 19 当前样点和相邻样点的位置关系

- b) 第二步,根据表 58 及图 20,利用当前样点 c 的在去块效应滤波后的 i 分量值 xc 与相邻样点 a 和 b 的滤波后样点 i 分量值 xa 和 xb 的关系确定当前样点 i 分量的 SaoOffset[i]。

表 58 相邻样点取值与偏移补偿值的对应关系

样点分量值的关系		偏移补偿值(SaoOffset[i])
Valley	$xc < xa \&\& xc < xb$	sao_offset[i][0]
Half valley	$(xc < xa \&\& xc == xb) (xc == xa \&\& xc < xb)$	sao_offset[i][1]
Half peak	$(xc > xa \&\& xc == xb) (xc == xa \&\& xc > xb)$	sao_offset[i][2]
peak	$xc > xa \&\& xc > xb$	sao_offset[i][3]
flat	其他	0

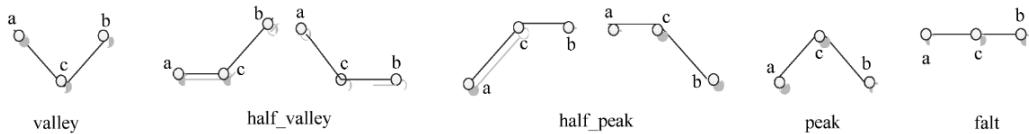


图 20 样点 c 与其相邻样点 a 和 b 的关系

c) 第三步,当前样点的 i 分量在偏移补偿后的取值 $y[i]$ 计算如下:

$$y[i] = Clip3(0, (1 \ll BitDepth) - 1, x[i] + SaoOffset[i])$$

其中, $x[i]$ 是该样点的 i 分量在去块效应滤波后的取值。

5.3.8.4.3 SAO_OFF 操作

如果 $sao_mode[compIdx]$ (compIdx 可以为 0,1,2 代表三个不同的分量) 为 SAO_OFF, 或当前样点是图像的边界, 则将去块滤波后样点对应分量的值直接作为偏移后该样点分量的值。

5.3.9 样点滤波补偿

5.3.9.1 概述

如果 $picture_alf_enable[comp_idx]$ 的值为 0, 则将偏移后样点分量的值直接作为解码图像中样点分量的值, 否则, 对相应的偏移后样点分量进行样点滤波补偿。其中 $comp_idx$ 等于 0 表示亮度分量; 等于 1 表示 Cb 分量; 等于 2 表示 Cr 分量。

样点滤波补偿的单位是由树形编码单元导出的样点滤波补偿单元, 按照光栅扫描顺序依次处理。首先根据 5.3.9.2 解码各分量的样点滤波补偿系数, 然后根据 5.3.9.3 导出样点滤波补偿单元, 根据 5.3.9.4 确定当前样点滤波补偿单元亮度分量的样点滤波补偿系数索引, 最后根据 5.3.9.5 对样点补偿滤波单元的亮度和色度分量进行样点滤波补偿, 得到解码图像的样点。

5.3.9.2 样点滤波补偿系数解码

样点滤波补偿系数的解码过程如下:

a) 从比特流中解析得到亮度样点的第 i 组补偿系数 $alf_coeff_luma[i][j]$ ($i=0 \sim alf_filter_num_minus1, j=0 \sim 8$)。对系数 $alf_coeff_luma[i][9]$ (即图 21 的系数 C9) 做以下处理:

$$alf_coeff_luma[i][9] += 64 - \sum_{j=0}^8 2 \times alf_coeff_luma[i][j]$$

其中 $alf_coeff_luma[i][j]$ ($j=0 \sim 8$) 的位宽是 7 位, 取值应为 $-64 \sim 63$ 。经上述处理后 $alf_coeff_luma[i][9]$ 的取值应为 $0 \sim 127$ 。

- b) 根据 alf_region_distance[i] ($i > 1$) 得到亮度分量样点滤波补偿系数索引数组(记作 alf_coeff_idx_tab [16]):

```

count=0
alf_coeff_idx_tab[0]=0
for(i=1; i<alf_filter_num_minus1+1; i++) {
    for(j=0; j<alf_region_distance[i]-1; j++) {
        alf_coeff_idx_tab [count+1]=alf_coeff_idx_tab [count]
        count=count+1
    }
    alf_coeff_idx_tab [count+1]=alf_coeff_idx_tab [count]+1
    count=count+1
}
for(i=count; i<16; i++)
    alf_coeff_idx_tab [i]=alf_coeff_idx_tab [count]
```

- c) 从比特流中解析得到色度样点的补偿系数 $\text{alf_coeff_chroma}[0][j]$ 和 $\text{alf_coeff_chroma}[1][j]$ ($j=0 \sim 8$)。对系数 $\text{alf_coeff_chroma}[0][9]$ 和 $\text{alf_coeff_chroma}[1][9]$ (即图 21 的系数 C9), 分别做以下处理:

$$\text{alf_coeff_chroma}[i][9] += 64 - \sum_{j=0}^8 2 \times \text{alf_coeff_chroma}[i][j], i=0,1$$

其中 $\text{alf_coeff_chroma}[i][j]$ ($j=0 \sim 7$) 的位宽是 7 位, 取值应为 $-64 \sim 63$ 。经上述处理后 $\text{alf_coeff_chroma}[i][9]$ 的取值应为 $0 \sim 127$ 。

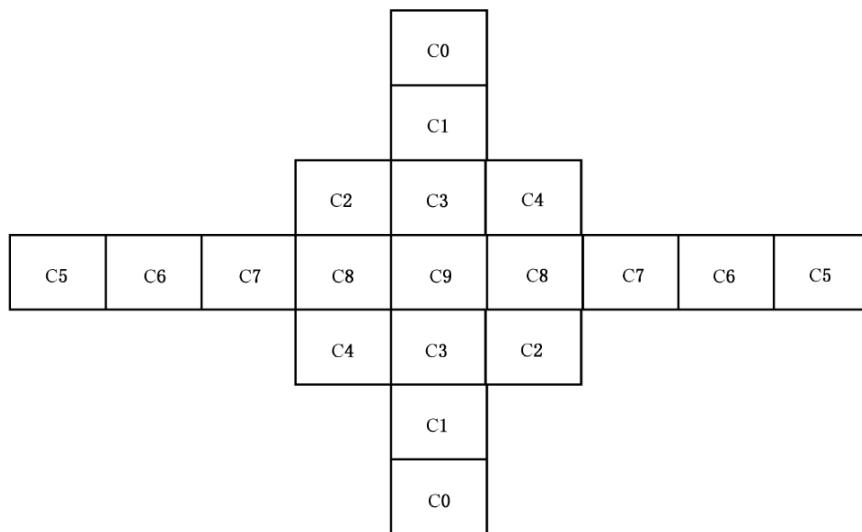


图 21 样点滤波补偿系数

5.3.9.3 导出样点滤波补偿单元

根据当前树形编码单元按下列步骤导出样点滤波补偿单元:

- 将当前树形编码单元 C 所在样点区域超出图像边界的部分删除, 得到样点区域 D;
- 如果区域 D 的下边界所在样点不属于图像的下边界, 将亮度分量和色度分量样点区域 D 的下边界向上收缩四行, 得到区域 E1; 否则, 令 E1 等于 D;

- 如果区域 E1 的上边界所在样点属于图像的上边界令 E2 等于 E1; 否则, 将亮度分量和色度分量样点区域 E1 的上边界向上扩展四行, 得到区域 E2;
- 将区域 E2 作为当前样点滤波补偿单元。图像的第一行样点为图像的上边界, 最后一行样点为图像的下边界。亮度分量和色度分量样点区域(D, E1, E2)的第一行样点为区域的上边界, 最后一行样点为区域的下边界。

5.3.9.4 确定亮度分量样点滤波补偿单元样点滤波补偿系数索引

根据以下方法计算当前亮度分量样点滤波补偿单元的样点滤波补偿系数索引, 记作 filter_idx:

```

horizontal_size= width_minus_1+1
vertical_size=height_minus_1+1
如果 extended_sb_size_flag 等于 1, ctu_size_in_bit 等于 6, 否则 ctu_size_in_bit 等于 5
x_interval=(((horizontal_size+(1 << ctu_size_in_bit)-1) >> ctu_size_in_bit)+1) >> 2) << ctu_size_in_bit
y_interval=(((vertical_size+(1 << ctu_size_in_bit)-1) >> ctu_size_in_bit)+1) >> 2) << ctu_size_in_bit
if (x_interval==0 && y_interval==0)
    index=15
else if (x_interval==0)
    index=Min(3,y/ y_interval)×4+3
else if (y_interval==0)
    index=Min(3,x/ x_interval)+12
else
    index=Min(3,y/ y_interval)×4+Min(3,x/ x_interval)
filter_idx=alf_coeff_idx_tab [region_table[index]]
```

其中 region_table[16]={0,1,4,5,15,2,3,6,14,11,10,7,13,12,9,8}, (x,y)是导出当前样点滤波补偿单元的树形编码单元左上角样点在图像中的坐标。

5.3.9.5 样点滤波操作

如果 alf_ctu_enable[comp_idx][ctu_index]等于 1, 则对 comp_idx 分量进行样点滤波补偿, 否则不进行样点补偿滤波。如果样点滤波补偿过程中用到了样点滤波补偿单元外的样点, 则按照如下方式处理:

- 如果当前样点滤波补偿单元左边界或右边界为图像边界, 则样点滤波补偿单元左边界或右边界外的样点分别用样点滤波单元内距离该样点最近的样点代替;
- 样点滤波补偿单元上边界和下边界外的样点分别用样点滤波单元内距离该样点最近的样点代替;

样点滤波补偿单元亮度分量的样点滤波补偿操作如下:

$$\begin{aligned}
 ptmp &= \text{alf_coeff_luma}[filter_idx][9] \times p(x, y) + \sum_{j=0}^8 2 \times \\
 &\quad \text{alf_coeff_luma}[filter_idx][j] \times (p(x - hor[j], y - ver[j]) + p(x + hor[j], y + ver[j])) \\
 p'(\mathbf{x}, \mathbf{y}) &= \text{Clip3}(0, (1 \ll \text{BitDepth}) - 1, ptmp)
 \end{aligned}$$

其中, $p(x, y)$ 为偏移后样点, $p'(\mathbf{x}, \mathbf{y})$ 为重建样点, $hor[j]$ 和 $ver[j]$ ($j = 0 \sim 7$) 见表 59。

表 59 样点补偿滤波坐标偏移值

j 的值	hor[j]的值	ver[j]的值
0	0	3
1	0	2
2	1	1
3	0	1
4	1	-1
5	3	0
6	2	0
7	1	0

5.3.10 空域可伸缩性视频编码(SVC)增强层编码片中 CTU 的解码过程

5.3.10.1 概述

解码 nal_unit_type 值为 3 和 4 的增强层编码片 NAL 单元中的树形编码单元时, 调用本过程。空域 SVC 的增强层与基本层之间的图像宽度比与高度比支持 4:3, 2:1, 4:1, 6:1 和 8:1, 由 svc_ratio 决定。

当 svc_mode 等于 0 时, 增强层的解码不使用跨层预测, 只使用增强层内的帧间或帧内预测, 解码过程与基本层相同。其帧内预测过程同 5.3.4, 帧间预测过程同 5.3.5, 重建过程同 5.3.6。

当 svc_mode 等于 1 时, 增强层的解码过程中使用层内预测和跨层预测, 解码过程见 5.3.10.3。

当 svc_roi_flag 等于 1 时, 增强层的解码过程见 5.3.10.5。

5.3.10.2 低分辨率层图像样点到对应位置的高分辨率层图像样点的插值计算

从低分辨率图像样点到对应位置的高分辨率图像样点的插值计算, 通过以下数组实现:

```
svac2_sub_pel_filters_8[]

{
    { 0, 0, 0, 128, 0, 0, 0, 0 },
    { 0, 1, -5, 126, 8, -3, 1, 0 },
    { -1, 3, -10, 122, 18, -6, 2, 0 },
    { -1, 4, -13, 118, 27, -9, 3, -1 },
    { -1, 4, -16, 112, 37, -11, 4, -1 },
    { -1, 5, -18, 105, 48, -14, 4, -1 },
    { -1, 5, -19, 97, 58, -16, 5, -1 },
    { -1, 6, -19, 88, 68, -18, 5, -1 },
    { -1, 6, -19, 78, 78, -19, 6, -1 },
    { -1, 5, -18, 68, 88, -19, 6, -1 },
    { -1, 5, -16, 58, 97, -19, 5, -1 },
    { -1, 4, -14, 48, 105, -18, 5, -1 },
    { -1, 4, -11, 37, 112, -16, 4, -1 },
    { -1, 3, -9, 27, 118, -13, 4, -1 },
    { 0, 2, -6, 18, 122, -10, 3, -1 },
    { 0, 1, -3, 8, 126, -5, 1, 0 }
}
```

};

设低分辨率图像的宽高分别为 src_w 和 src_h , 高分辨率图像的宽高分别为 dst_w 和 dst_h 。 factor_x 和 factor_y 为根据色度采样格式和块类型确定的缩放因子。计算出每个样点对应的缩放比例:

```
factor=(Y ? 1 : 2);
x_q4=x×(16/factor)×src_w/dst_w;
y_q4=y×(16/factor)×src_h/dst_h;
```

根据缩放比例 x_{q4} 和 y_{q4} , 分别在插值数组中选择相应的插值系数, 进行水平和垂直的插值计算, 得到高分辨率图像样点。

5.3.10.3 `svc_mode` 等于 1 时增强层的预测及图像重建过程

本过程输入为增强层码流, 对应基本层解码图像及各 8×8 块的 MV, 输出为增强层的重构图像。

增强层的残差样点由码流解析生成的残差系数矩阵经逆扫描, 反量化及反变换过程得到, 具体过程见 5.3.6.2~5.3.6.5。

增强层预测矩阵的计算方式如下:

增强层的参考图像集应与基本层的参考图像集对应, 即相同的参考帧类型对应同一图像的增强层和基本层。另外, 根据增强层图像的参考帧类型来判断是跨层预测还是层内预测:

- 如果参考帧为 `static_ref`, 则增强层采用跨层预测。使用基本层解码图像经 5.3.10.2 插值放大后的图像作为参考帧, 并根据运动矢量计算得到预测矩阵, 其中运动矢量的计算过程见 5.3.10.4;

- 如果参考帧类型不是 `static_ref`, 则增强层采用层内预测, 帧间预测同 5.3.5, 得到预测矩阵。

增强层的重建过程同 5.3.6.6。

5.3.10.4 跨层预测时运动矢量的计算

跨层预测且当在增强层与基本层的宽度比和高度比为 2:1 时, 基本层放大后图像中每个 16×16 块中各 8×8 块的亮度运动矢量由基本层图像中对应的 8×8 块的亮度运动矢量进行如下扩展获得:

$\text{EL_MV}=\text{BL_MV} \times 2$

$\text{EL_REF}=\text{BL_REF}$

其中 EL_MV 和 EL_REF 分别为增强层的 MV 和参考帧类型, BL_MV 和 BL_REF 分别为基本层的 MV 和参考帧类型。

增强层在进行候选运动矢量集导出且当增强层与基本层的宽度比和高度比等于 2:1 时, 在 5.3.5.2.1 节中第一步与第二步之间插入一个步骤: 如果基本层放大后的当前块相同位置的块使用的参考帧与当前块的参考帧相同, 则该块对应的基本层扩展放大后的 MV 加入候选运动矢量集。当增强层与基本层的宽度比和高度比为 2:1 时, 候选运动矢量集的导出过程同 5.3.5.2.1。

5.3.10.5 `svc_roi_flag` 等于 1 时增强层的预测及重建过程

当 `svc_roi_flag` 等于 1 时, 增强层支持 ROI 解码。ROI 区域的预测及重建过程同 5.3.10.3, 非 ROI 区域的重建样点矩阵的取值等于相邻低分辨率层的对应位置的块的样点矩阵, 经 5.3.10.2 插值后生成的样点矩阵。

5.4 解析过程

5.4.1 概述

本过程的输入为 RBSP 的比特流, 输出为语法元素值。

对于 5.2.3 中的语法元素, ae(v) 解析过程见 5.4.2, ue(v)、se(v) 的解析过程见 5.4.3。

5.4.2 算术码解析过程

5.4.2.1 概述

本过程的输入为 RBSP 的比特流。本过程的输出为语法元素值。

ae(v) 描述的语法元素解析过程如下:

- 对 Tile 进行解析前,首先进行初始化,还会对一些语法元素的概率表进行更新,见 5.4.2.2;如果 wpp_enable 等于 1,在每 CTU 行解析开始前,需要重新初始化算术解码器,见 5.4.2.2.3;
- 对二进制位串进行解析,见 5.4.2.4:
 - 二进制位串中每个二进制位的索引号为 binIdx, 对应概率的获取见 5.4.2.4.4;
 - 根据 ctxIdx 解析二进制位, 见 5.4.2.4.5;
 - 完成一个二进制位的解析后, 将得到的二进制位串与二值化过程得到的二进制位串集合进行比较。如果得到的二进制位串与集合中某个二进制位串相匹配, 则输出相应语法元素值; 否则继续解析二进制位。

ae(v) 描述的语法元素解析过程用伪代码描述如下:

```

if ( 当前语法元素为编码片的第一个 ae(v) 语法元素 ) {
    初始化算术码解码器
}
binIdx = -1
do {
    binIdx++
    得到与 binIdx 对应的 ctxIdx
    得到与 ctxIdx 对应的概率模型 prob
    根据 prob 解析二进制位
} while ( (b0, ..., bbinIdx) 不是语法元素的二进制位串 )
输出语法元素值

```

5.4.2.2 初始化

5.4.2.2.1 初始化概率模型

本过程的输出是初始化后算术码上下文变量所对应的 prob, 即二进制位解析所需的概率值。该变量通过 binIdx、treeIdx 及 ctxIdx 索引。

 comp 用来表示 MV 的概率表索引。comp 为 0 表示垂直坐标, comp 为 1 表示水平坐标。

对于每个上下文变量, 应初始化每个 ctxIdx 索引所对应的概率状态。

初始化过程所涉及的语法元素概率表见表 60。

表 60 初始过程所涉及的语法元素概率表汇总

语法元素	概率表
alf_ctu_enable	表 61
partition	表 62~表 63
skip_flag	表 64

表 60 (续)

语法元素	概率表
inter_block	表 65
tx_size	表 66
prev_intra_luma_pred_flag	表 67
uv_fllow_y_flag	表 68
block_reference_mode	表 69
ref_frame	表 70~表 71
mv_mode	表 72~表 74
mv_joint	表 75
mvd_sign_0 和 mvd_sign_1	表 76
mvd_value_0 和 mvd_value_1	表 77~表 83
interp_filter_mode	表 84
dqp_abs	表 85
coef_value	表 86~表 96

语法元素 alf_ctu_enable 的初始概率表见表 61。

表 61 alf_ctu_enable_prob

ctxIdx	binIdx	
	0	1
0	229	
1		115
2	25	
3		9

frame_type 等于 1 时, 语法元素 partition 的初始概率表见表 62。frame_type 等于 0 时, 语法元素 partition 的初始概率表见表 63。

表 62 inter_partition_prob

ctxIdx	treeIdx		
	0	1 _{SAC}	2
	8×8 → 4×4		
0(a/l both not split)	199	122	141
1(a split,l not split)	147	63	159
2(l split,a not split)	148	133	118
3(a/l both split)	121	104	114

表 62 (续)

ctxIdx	treeIdx		
	0	1	2
$16 \times 16 \rightarrow 8 \times 8$			
4(a/l both not split)	174	73	87
5(a split,l not split)	92	41	83
6(l split,a not split)	82	99	50
7(a/l both split)	53	39	39
$32 \times 32 \rightarrow 16 \times 16$			
8(a/l both not split)	177	58	59
9(a split,l not split)	68	26	63
10(l split,a not split)	52	79	25
11(a/l both split)	17	14	12
$64 \times 64 \rightarrow 32 \times 32$			
12(a/l both not split)	222	34	30
13(a split,l not split)	72	16	44
14(l split,a not split)	58	32	12
15(a/l both split)	10	7	6
$128 \times 128 \rightarrow 64 \times 64$			
16(a/l both not split)	220	33	28
17(a split,l not split)	70	15	43
18(l split,a not split)	57	31	11
19(a/l both split)	9	6	6

表 63 intra_partition_prob

ctxIdx	treeIdx		
	0	1	2
$8 \times 8 \rightarrow 4 \times 4$			
0(a/l both not split)	158	97	94
1(a split,l not split)	93	24	99
2(l split,a not split)	85	119	44
3(a/l both split)	62	59	67
$16 \times 16 \rightarrow 8 \times 8$			
4(a/l both not split)	149	53	53
5(a split,l not split)	94	20	48
6(l split,a not split)	83	53	24
7(a/l both split)	52	18	18

表 63 (续)

ctxIdx	treeIdx		
	0	1	2
$32 \times 32 \rightarrow 16 \times 16$			
8(a/l both not split)	150	40	39
9(a split,l not split)	78	12	26
10(l split,a not split)	67	33	11
11(a/l both split)	24	7	5
$64 \times 64 \rightarrow 32 \times 32$			
12(a/l both not split)	174	35	49
13(a split,l not split)	68	11	27
14(l split,a not split)	57	15	9
15(a/l both split)	12	3	3
$128 \times 128 \rightarrow 64 \times 64$			
16(a/l both not split)	160	32	45
17(a split,l not split)	65	10	23
18(l split,a not split)	54	12	8
19(a/l both split)	11	2	2

语法元素 skip_flag 使用的初始概率表见表 64。

表 64 skip_prob

ctxIdx	binIdx	
	0	1
0	192	
1		128
2		64

语法元素 inter_block 的初始概率表见表 65。

表 65 intra_inter_prob

ctxIdx	binIdx	
	0	1
0	9	
1		102
2		187
3		225

MAX_TX_SIZE 取值不同时,语法元素 tx_size 的初始概率表见表 66。

表 66 tx_probs

ctxIdx	binIdx		
	0	1	2
TX_32×32			
0	3	136	37
1	5	52	13
TX_16×16			
0	20	152	—
1	15	101	—
TX_8×8			
0	100	—	—
1	66	—	—

语法元素 prev_intra_luma_pred_flag 的初始概率表见表 67。

表 67 mpm_flag_probs

ctxIdx	binIdx	
	0	1
0	80	
1	84	
2		110
3		128
4		128
5		110
6		80

语法元素 uv_fllow_y_flag 的初始概率表见表 68。

表 68 uv_fllow_y_prob

ctxIdx	binIdx	
	0	1
0	175	

语法元素 block_reference_mode 的初始概率表见表 69。

表 69 comp_inter_prob

ctxIdx	binIdx	
	0	1
0	239	
1		183
2		119
3		96

语法元素 ref_frame 在 COMPOUND_REFERENCE 模式下的初始概率表见表 70, 在 SINGLE_REFERENCE 模式下的初始概率表见表 71。

表 70 comp_ref_prob

ctxIdx	binIdx		
	0	1	2
0	50	50	50
1	126	126	126
2	123	123	123
3	221	221	221
4	226	226	226

表 71 single_ref_prob

ctxIdx	binIdx			
	0	1	2	3
0	33	16	16	16
1	77	74	74	74
2	142	142	142	142
3	172	170	170	170
4	238	247	247	247

语法元素 mv_mode 用到三个初始概率表, 分别见表 72~表 74。

表 72 newmv_mode_prob

ctxIdx	binIdx	
	0	1
0	200	
1	180	
2	150	
3	150	
4	110	
5	70	
6	60	

表 73 zeromv_mode_prob

ctxIdx	binIdx	
	0	1
0	192	
1	64	

表 74 refmv_mode_prob

ctxIdx	binIdx	
	0	1
0	220	
1	220	
2	200	
3	200	
4	180	
5	128	
6		1
7	250	

运动矢量解析相关的初始概率表见表 75～表 83。

表 75 mv_joint_probs

ctxIdx	treeIdx		
	0	1	2
0	190	155	212

表 76 mv_sign_probs

comp	binIdx	
	0	1
0	128	
1	128	

表 77 mv_bits_probs

comp	binIdx									
	0	1	2	3	4	5	6	7	8	9
0	136	140	148	160	176	192	224	234	234	240
1	136	140	148	160	176	192	224	234	234	240

表 78 mv_class0_bit_probs

comp	binIdx	
	0	1
0	216	
1	208	

表 79 mv_class_probs

comp	treeIdx									
	0	1	2	3	4	5	6	7	8	9
0	224	144	192	168	192	176	192	198	198	245
1	216	128	176	160	176	176	192	198	198	208

表 80 mv_class0_fr_probs

comp	mv_class0_bit	treeIdx		
		0	1	2
0	0	128	128	64
0	1	96	112	64
1	0	128	128	64
1	1	96	112	64

表 81 mv_class0_hp_probs

comp	binIdx	
	0	1
0		160
1		160

表 82 mv_fr_probs

comp	treeIdx		
	0	1	2
0	64	96	64
1	64	96	64

表 83 mv_hp_probs

comp	binIdx	
	0	1
0		128
1		128

语法元素 interp_filter_mode 的初始化概率见表 84。

表 84 switchable_interp_prob

ctxIdx	treeIdx		
	0	1	2
0	235	192	128
1	36	243	208
2	34	16	128
3	36	243	48
4	149	160	128

语法元素 dqp_abs 的初始概率表见表 85。

表 85 dqp_abs_prob

ctxIdx	treeIdx						
	0	1	2	3	4	5	6
0	180	190	161	144	138	124	79
1	121	133	161	107	80	100	50

语法元素 coef_value 的解析,包含 token 和 extra 的计算。token 的计算需要用到 coef_probs 概率表和 svac2_pareto8_full 概率表,见表 86~表 90;extra 的计算需要用到 svac2_cat1_prob、svac2_cat2_prob、svac2_cat3_prob、svac2_cat4_prob、svac2_cat5_prob 和 svac2_cat6_prob 概率表。coef_probs 的上下文模型共有 6 个维度的信息: TX_SIZES、PLANE_TYPES、REF_TYPES、COEF_BANDS、COEFF_CONTEXTS、binIdx。

表 86 coef_probs 4×4

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	195	29	183
			1	84	49	136
			2	8	42	71
		1	0	31	107	169
			1	35	99	159
			2	17	82	140
			3	8	66	114
			4	2	44	76
			5	1	19	32
		2	0	40	132	201
			1	29	114	187
			2	13	91	157

表 86 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	2	3	7	75	127
			4	3	58	95
			5	1	28	47
		3	0	69	142	221
			1	42	122	201
			2	15	91	159
			3	6	67	121
			4	1	42	77
			5	1	17	31
		4	0	102	148	228
			1	67	117	204
			2	17	82	154
			3	6	59	114
			4	2	39	75
			5	1	15	29
Inter	Inter	5	0	156	57	233
			1	119	57	212
			2	58	48	163
			3	29	40	124
			4	12	30	81
			5	3	12	31
		0	0	191	107	226
			1	124	117	204
			2	25	99	155
		1	0	29	148	210
			1	37	126	194
			2	8	93	157
			3	2	68	118
			4	1	39	69
			5	1	17	33
		2	0	41	151	213
			1	27	123	193
			2	3	82	144

表 86 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	2	3	1	58	105
			4	1	32	60
			5	1	13	26
		3	0	59	159	220
			1	23	126	198
			2	4	88	151
		4	3	1	66	114
			4	1	38	71
			5	1	18	34
		5	0	114	136	232
			1	51	114	207
			2	11	83	155
			3	3	56	105
			4	1	33	65
			5	1	17	34
UV plane	Intra	0	0	149	65	234
			1	121	57	215
			2	61	49	166
		1	3	28	36	114
			4	12	25	76
			5	3	16	42
		2	0	214	49	220
			1	132	63	188
			2	42	65	137
		3	0	85	137	221
			1	104	131	216
			2	49	111	192
			3	21	87	155
			4	2	49	87
			5	1	16	28
		4	0	89	163	230
			1	90	137	220
			2	29	100	183

表 86 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	2	3	10	70	135
			4	2	42	81
			5	1	17	33
		3	0	108	167	237
			1	55	133	222
			2	15	97	179
		4	3	4	72	135
			4	1	45	85
			5	1	19	38
		5	0	124	146	240
			1	66	124	224
			2	17	88	175
			3	4	58	122
			4	1	36	75
			5	1	18	37
	Inter	0	0	141	79	241
			1	126	70	227
			2	66	58	182
			3	30	44	136
			4	12	34	96
			5	2	20	47
		1	0	229	99	249
			1	143	111	235
			2	46	109	192
			0	82	158	236
			1	94	146	224
			2	25	117	191
		2	3	9	87	149
			4	3	56	99
			5	1	33	57

表 86 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	SAC	2	3	2	72
				4	1	41
				5	1	20
			3	0	99	167
				1	47	141
				2	10	104
			4	3	2	73
				4	1	44
				5	1	22
			5	0	127	145
				1	71	129
				2	17	93
			6	3	3	61
				4	1	41
				5	1	21
			7	0	157	78
				1	140	72
				2	69	58
				3	31	44
				4	14	38
				5	8	23
				61		

表 87 coef_probs 8×8

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	0	125	34
				1	52	41
				2	6	31
			1	0	37	109
				1	51	102
				2	23	87
				3	8	67
				4	1	41
				63		

表 87 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	1	5	1	19	29
			0	31	154	185
			1	17	127	175
			2	6	96	145
			3	2	73	114
			4	1	51	82
			5	1	28	45
		2	0	23	163	200
			1	10	131	185
			2	2	93	148
			3	1	67	111
			4	1	41	69
			5	1	14	24
		3	0	29	176	217
			1	12	145	201
			2	3	101	156
			3	1	69	111
			4	1	39	63
			5	1	14	23
		4	0	57	192	233
			1	25	154	215
			2	6	109	167
			3	3	78	118
			4	1	48	69
			5	1	21	29
	Inter	0	0	202	105	245
			1	108	106	216
			2	18	90	144
		1	0	33	172	219
			1	64	149	206
			2	14	117	177
			3	5	90	141
			4	2	61	95

表 87 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	SAC	1	5	1	37
			2	0	33	179
			2	1	11	140
			2	2	1	89
			2	3	1	60
			2	4	1	33
			2	5	1	12
		3	3	0	30	181
			3	1	8	141
			3	2	1	87
			3	3	1	58
			3	4	1	31
			3	5	1	12
		4	4	0	32	186
			4	1	7	142
			4	2	1	86
			4	3	1	58
			4	4	1	31
			4	5	1	12
		5	5	0	57	192
			5	1	20	143
			5	2	3	96
			5	3	1	68
			5	4	1	42
			5	5	1	19
UV plane	Intra	0	0	0	212	35
			0	1	113	47
			0	2	29	48
		1	1	0	74	129
			1	1	106	120
			1	2	49	107
			1	3	19	84
			1	4	4	50
			1	5	1	15
			1			84

表 87 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	2	0	71	172	217
			1	44	141	209
			2	15	102	173
			3	6	76	133
			4	2	51	89
			5	1	24	42
		3	0	64	185	231
			1	31	148	216
			2	8	103	175
			3	3	74	131
			4	1	46	81
			5	1	18	30
		4	0	65	196	235
			1	25	157	221
			2	5	105	174
			3	1	67	120
			4	1	38	69
			5	1	15	30
		5	0	65	204	238
			1	30	156	224
			2	7	107	177
			3	2	70	124
			4	1	42	73
			5	1	18	34
	Inter	0	0	225	86	251
			1	144	104	235
			2	42	99	181
		1	0	85	175	239
			1	112	165	229
			2	29	136	200
			3	12	103	162
			4	6	77	123
			5	2	53	84

表 87 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	2	0	75	183	239
			1	30	155	221
			2	3	106	171
			3	1	74	128
			4	1	44	76
			5	1	17	28
		3	0	73	185	240
			1	27	159	222
			2	2	107	172
			3	1	75	127
			4	1	42	73
			5	1	17	29
		4	0	62	190	238
			1	21	159	222
			2	2	107	172
			3	1	72	122
			4	1	40	71
			5	1	18	32
		5	0	61	199	240
			1	27	161	226
			2	4	113	180
			3	1	76	129
			4	1	46	80
			5	1	23	41

表 88 coef_probs 16 × 16

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	7	27	153
			1	5	30	95
		1	2	1	16	30
			0	50	75	127

表 88 coef_probs 16×16

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	1	1	57	75	124
			2	27	67	108
			3	10	54	86
			4	1	33	52
			5	1	12	18
		2	0	43	125	151
			1	26	108	148
			2	7	83	122
			3	2	59	89
			4	1	38	60
			5	1	17	27
		3	0	23	144	163
			1	13	112	154
			2	2	75	117
			3	1	50	81
			4	1	31	51
			5	1	14	23
		4	0	18	162	185
			1	6	123	171
			2	1	78	125
			3	1	51	86
			4	1	31	54
			5	1	14	23
		5	0	15	199	227
			1	3	150	204
			2	1	91	146
			3	1	55	95
			4	1	30	53
			5	1	11	20
	Inter	0	0	19	55	240
			1	19	59	196
			2	3	52	105

表 88 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	1	0	41	166	207
			1	104	153	199
			2	31	123	181
		2	3	14	101	152
			4	5	72	106
			5	1	36	52
		3	0	35	176	211
			1	12	131	190
			2	2	88	144
			3	1	60	101
			4	1	36	60
			5	1	16	28
		4	0	28	183	213
			1	8	134	191
			2	1	86	142
			3	1	56	96
			4	1	30	53
			5	1	12	20
		5	0	20	190	215
			1	4	135	192
			2	1	84	139
			3	1	53	91
			4	1	28	49
			5	1	11	20
UV plane	Intra	0	0	13	196	216
			1	2	137	192
			2	1	86	143
			3	1	57	99
			4	1	32	56
			5	1	13	24



表 88 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	1	0	78	120	193
			1	111	116	186
			2	46	102	164
			3	15	80	128
			4	2	49	76
			5	1	18	28
		2	0	71	161	203
			1	42	132	192
			2	10	98	150
			3	3	69	109
			4	1	44	70
			5	1	18	29
		3	0	57	186	211
			1	30	140	196
			2	4	93	146
			3	1	62	102
			4	1	38	65
			5	1	16	27
		4	0	47	199	217
			1	14	145	196
			2	1	88	142
			3	1	57	98
			4	1	36	62
			5	1	15	26
		5	0	26	219	229
			1	5	155	207
			2	1	94	151
			3	1	60	104
			4	1	36	62
			5	1	16	28
	Inter	0	0	233	29	248
			1	146	47	220
			2	43	52	140

表 88 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	1	0	100	163	232
			1	179	161	222
			2	63	142	204
			3	37	113	174
			4	26	89	137
			5	18	68	97
		2	0	85	181	230
			1	32	146	209
			2	7	100	164
			3	3	71	121
			4	1	45	77
			5	1	18	30
		3	0	65	187	230
			1	20	148	207
			2	2	97	159
			3	1	68	116
			4	1	40	70
			5	1	14	29
		4	0	40	194	227
			1	8	147	204
			2	1	94	155
			3	1	65	112
			4	1	39	66
			5	1	14	26
		5	0	16	208	228
			1	3	151	207
			2	1	98	160
			3	1	67	117
			4	1	41	74
			5	1	17	31

SAC

表 89 coef_probs 32×32

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	17	38	140
			1	7	34	80
			2	1	17	29
		1	0	37	75	128
			1	41	76	128
			2	26	66	116
			3	12	52	94
			4	2	32	55
			5	1	10	16
		2	0	50	127	154
			1	37	109	152
			2	16	82	121
			3	5	59	85
			4	1	35	54
			5	1	13	20
		3	0	40	142	167
			1	17	110	157
			2	2	71	112
			3	1	44	72
			4	1	27	45
			5	1	11	17
		4	0	30	175	188
			1	9	124	169
			2	1	74	116
			3	1	48	78
			4	1	30	49
			5	1	11	18
		5	0	10	222	223
			1	2	150	194
			2	1	83	128
			3	1	48	79
			4	1	27	45
			5	1	11	17

表 89 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	0	0	36	41	235
			1	29	36	193
			2	10	27	111
		1	0	85	165	222
			1	177	162	215
			2	110	135	195
			3	57	113	168
			4	23	83	120
			5	10	49	61
		2	0	85	190	223
			1	36	139	200
			2	5	90	146
			3	1	60	103
			4	1	38	65
			5	1	18	30
		3	0	72	202	223
			1	23	141	199
			2	2	86	140
			3	1	56	97
			4	1	36	61
			5	1	16	27
		4	0	55	218	225
			1	13	145	200
			2	1	86	141
			3	1	57	99
			4	1	35	61
			5	1	13	22
		5	0	15	235	212
			1	1	132	184
			2	1	84	139
			3	1	57	97
			4	1	34	56
			5	1	14	23

表 89 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	0	0	181	21	201
			1	61	37	123
			2	10	38	71
		1	0	47	106	172
			1	95	104	173
			2	42	93	159
			3	18	77	131
			4	4	50	81
			5	1	17	23
		2	0	62	147	199
			1	44	130	189
			2	28	102	154
			3	18	75	115
			4	2	44	65
			5	1	12	19
		3	0	55	153	210
			1	24	130	194
			2	3	93	146
			3	1	61	97
			4	1	31	50
			5	1	10	16
		4	0	49	186	223
			1	17	148	204
			2	1	96	142
			3	1	53	83
			4	1	26	44
			5	1	11	17
		5	0	13	217	212
			1	2	136	180
			2	1	78	124
			3	1	50	83
			4	1	29	49
			5	1	14	23

表 89 (续)

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	0	0	197	13	247
			1	82	17	222
			2	25	17	162
		1	0	126	186	247
			1	234	191	243
			2	176	177	234
			3	104	158	220
			4	66	128	186
			5	55	90	137
		2	0	111	197	242
			1	46	158	219
			2	9	104	171
			3	2	65	125
			4	1	44	80
			5	1	17	91
		3	0	104	208	245
			1	39	168	224
			2	3	109	162
			3	1	79	124
			4	1	50	102
			5	1	43	102
		4	0	84	220	246
			1	31	177	231
			2	2	115	180
			3	1	79	134
			4	1	55	77
			5	1	60	79
		5	0 	43	243	240
			1	8	180	217
			2	1	115	166
			3	1	84	121
			4	1	51	67
			5	1	16	6

表 90 svac2_pareto8_full

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
0	3	86	128	6	86	23	88	29
1	6	86	128	11	87	42	91	52
2	9	86	129	17	88	61	94	76
3	12	86	129	22	88	77	97	93
4	15	87	129	28	89	93	100	110
5	17	87	129	33	90	105	103	123
6	20	88	130	38	91	118	106	136
7	23	88	130	43	91	128	108	146
8	26	89	131	48	92	139	111	156
9	28	89	131	53	93	147	114	163
10	31	90	131	58	94	156	117	171
11	34	90	131	62	94	163	119	177
12	37	90	132	66	95	171	122	184
13	39	90	132	70	96	177	124	189
14	42	91	132	75	97	183	127	194
15	44	91	132	79	97	188	129	198
16	47	92	133	83	98	193	132	202
17	49	92	133	86	99	197	134	205
18	52	93	133	90	100	201	137	208
19	54	93	133	94	100	204	139	211
20	57	94	134	98	101	208	142	214
21	59	94	134	101	102	211	144	216
22	62	94	135	105	103	214	146	218
23	64	94	135	108	103	216	148	220
24	66	95	135	111	104	219	151	222
25	68	95	135	114	105	221	153	223
26	71	96	136	117	106	224	155	225
27	73	96	136	120	106	225	157	226
28	76	97	136	123	107	227	159	228
29	78	97	136	126	108	229	160	229
30	80	98	137	129	109	231	162	231
31	82	98	137	131	109	232	164	232
32	84	98	138	134	110	234	166	233

表 90 (续)

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
33	86	98	138	137	111	235	168	234
34	89	99	138	140	112	236	170	235
35	91	99	138	142	112	237	171	235
36	93	100	139	145	113	238	173	236
37	95	100	139	147	114	239	174	237
38	97	101	140	149	115	240	176	238
39	99	101	140	151	115	241	177	238
40	101	102	140	154	116	242	179	239
41	103	102	140	156	117	242	180	239
42	105	103	141	158	118	243	182	240
43	107	103	141	160	118	243	183	240
44	109	104	141	162	119	244	185	241
45	111	104	141	164	119	244	186	241
46	113	104	142	166	120	245	187	242
47	114	104	142	168	121	245	188	242
48	116	105	143	170	122	246	190	243
49	118	105	143	171	122	246	191	243
50	120	106	143	173	123	247	192	244
51	121	106	143	175	124	247	193	244
52	123	107	144	177	125	248	195	244
53	125	107	144	178	125	248	196	244
54	127	108	145	180	126	249	197	245
55	128	108	145	181	127	249	198	245
56	130	109	145	183	128	249	199	245
57	132	109	145	184	128	249	200	245
58	134	110	146	186	129	250	201	246
59	135	110	146	187	130	250	202	246
60	137	111	147	189	131	251	203	246
61	138	111	147	190	131	251	204	246
62	140	112	147	192	132	251	205	247
63	141	112	147	193	132	251	206	247
64	143	113	148	194	133	251	207	247
65	144	113	148	195	134	251	207	247

表 90 (续)

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
66	146	114	149	197	135	252	208	248
67	147	114	149	198	135	252	209	248
68	149	115	149	199	136	252	210	248
69	150	115	149	200	137	252	210	248
70	152	115	150	201	138	252	211	248
71	153	115	150	202	138	252	212	248
72	155	116	151	204	139	253	213	249
73	156	116	151	205	139	253	213	249
74	158	117	151	206	140	253	214	249
75	159	117	151	207	141	253	215	249
76	161	118	152	208	142	253	216	249
77	162	118	152	209	142	253	216	249
78	163	119	153	210	143	253	217	249
79	164	119	153	211	143	253	217	249
80	166	120	153	212	144	254	218	250
81	167	120	153	212	145	254	219	250
82	168	121	154	213	146	254	220	250
83	169	121	154	214	146	254	220	250
84	171	122	155	215	147	254	221	250
85	172	122	155	216	147	254	221	250
86	173	123	155	217	148	254	222	250
87	174	123	155	217	149	254	222	250
88	176	124	156	218	150	254	223	250
89	177	124	156	219	150	254	223	250
90	178	125	157	220	151	254	224	251
91	179	125	157	220	151	254	224	251
92	180	126	157	221	152	254	225	251
93	181	126	157	221	152	254	225	251
94	183	127	158	222	153	254	226	251
95	184	127	158	223	154	254	226	251
96	185	128	159	224	155	255	227	251
97	186	128	159	224	155	255	227	251
98	187	129	160	225	156	255	228	251

表 90 (续)

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
99	188	130	160	225	156	255	228	251
100	189	131	160	226	157	255	228	251
101	190	131	160	226	158	255	228	251
102	191	132	161	227	159	255	229	251
103	192	132	161	227	159	255	229	251
104	193	133	162	228	160	255	230	252
105	194	133	162	229	160	255	230	252
106	195	134	163	230	161	255	231	252
107	196	134	163	230	161	255	231	252
108	197	135	163	231	162	255	231	252
109	198	135	163	231	162	255	231	252
110	199	136	164	232	163	255	232	252
111	200	136	164	232	164	255	232	252
112	201	137	165	233	165	255	233	252
113	201	137	165	233	165	255	233	252
114 	202	138	166	233	166	255	233	252
115	203	138	166	233	166	255	233	252
116	204	139	166	234	167	255	234	252
117	205	139	166	234	167	255	234	252
118	206	140	167	235	168	255	235	252
119	206	140	167	235	168	255	235	252
120	207	141	168	236	169	255	235	252
121	208	141	168	236	170	255	235	252
122	209	142	169	237	171	255	236	252
123	209	143	169	237	171	255	236	252
124	210	144	169	237	172	255	236	252
125	211	144	169	237	172	255	236	252
126	212	145	170	238	173	255	237	252
127	213	145	170	238	173	255	237	252
128	214	146	171	239	174	255	237	253
129	214	146	171	239	174	255	237	253
130	215	147	172	240	175	255	238	253
131	215	147	172	240	175	255	238	253

表 90 (续)

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
132	216	148	173	240	176	255	238	253
133	217	148	173	240	176	255	238	253
134	218	149	173	241	177	255	239	253
135	218	149	173	241	178	255	239	253
136	219	150	174	241	179	255	239	253
137	219	151	174	241	179	255	239	253
138	220	152	175	242	180	255	240	253
139	221	152	175	242	180	255	240	253
140	222	153	176	242	181	255	240	253
141	222	153	176	242	181	255	240	253
142	223	154	177	243	182	255	240	253
143	223	154	177	243	182	255	240	253
144	224	155	178	244	183	255	241	253
145	224	155	178	244	183	255	241	253
146	225	156	178	244	184	255	241	253
147	225	157	178	244	184	255	241	253
148	226	158	179	244	185	255	242	253
149	227	158	179	244	185	255	242	253
150	228	159	180	245	186	255	242	253
151	228	159	180	245	186	255	242	253
152	229	160	181	245	187	255	242	253
153	229	160	181	245	187	255	242	253
154	230	161	182	246	188	255	243	253
155	230	162	182	246	188	255	243	253
156	231	163	183	246	189	255	243	253
157	231	163	183	246	189	255	243	253
158	232	164	184	247	190	255	243	253
159	232	164	184	247	190	255	243	253
160	233	165	185	247	191	255	244	253
161	233	165	185	247	191	255	244	253
162	234	166	185	247	192	255	244	253
163	234	167	185	247	192	255	244	253
164	235	168	186	248	193	255	244	253

表 90 (续)

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
165	235	168	186	248	193	255	244	253
166	236	169	187	248	194	255	244	253
167	236	169	187	248	194	255	244	253
168	236	170	188	248	195	255	245	253
169	236	170	188	248	195	255	245	253
170	237	171	189	249	196	255	245	254
171	237	172	189	249	196	255	245	254
172	238	173	190	249	197	255	245	254
173	238	173	190	249	197	255	245	254
174	239	174	191	249	198	255	245	254
175	239	174	191	249	198	255	245	254
176	240	175	192	249	199	255	246	254
177	240	176	192	249	199	255	246	254
178	240	177	193	250	200	255	246	254
179	240	177	193	250	200	255	246	254
180	241	178	194	250	201	255	246	254
181	241	178	194	250	201	255	246	254
182	242	179	195	250	202	255	246	254
183	242	180	195	250	202	255	246	254
184	242	181	196	250	203	255	247	254
185	242	181	196	250	203	255	247	254
186	243	182	197	251	204	255	247	254
187	243	183	197	251	204	255	247	254
188	244	184	198	251	205	255	247	254
189	244	184	198	251	205	255	247	254
190	244	185	199	251	206	255	247	254
191	244	185	199	251	206	255	247	254
192	245	186	200	251	207	255	247	254
193	245	187	200	251	207	255	247	254
194	246	188	201	252	207	255	248	254
195	246	188	201	252	207	255	248	254
196	246	189	202	252	208	255	248	254
197	246	190	202	252	208	255	248	254

表 90 (续)

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
198	247	191	203	252	209	255	248	254
199	247	191	203	252	209	255	248	254
200	247	192	204	252	210	255	248	254
201	247	193	204	252	210	255	248	254
202	248	194	205	252	211	255	248	254
203	248	194	205	252	211	255	248	254
204	248	195	206	252	212	255	249	254
205	248	196	206	252	212	255	249	254
206	249	197	207	253	213	255	249	254
207	249	197	207	253	213	255	249	254
208	249	198	208	253	214	255	249	254
209	249	199	209	253	214	255	249	254
210	250	200	210	253	215	255	249	254
211	250	200	210	253	215	255	249	254
212	250	201	211	253	215	255	249	254
213	250	202	211	253	215	255	249	254
214	250	203	212	253	216	255	249	254
215	250	203	212	253	216	255	249	254
216	251	204	213	253	217	255	250	254
217	251	205	213	253	217	255	250	254
218	251	206	214	254	218	255	250	254
219	251	206	215	254	218	255	250	254
220	252	207	216	254	219	255	250	254
221	252	208	216	254	219	255	250	254
222	252	209	217	254	220	255	250	254
223	252	210	217	254	220	255	250	254
224	252	211	218	254	221	255	250	254
225	252	212	218	254	221	255	250	254
226	253	213	219	254	222	255	250	254
227	253	213	220	254	222	255	250	254
228	253	214	221	254	223	255	250	254
229	253	215	221	254	223	255	250	254
230	253	216	222	254	224	255	251	254

表 90 (续)

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
231	253	217	223	254	224	255	251	254
232	253	218	224	254	225	255	251	254
233	253	219	224	254	225	255	251	254
234	254	220	225	254	225	255	251	254
235	254	221	226	254	225	255	251	254
236	254	222	227	255	226	255	251	254
237	254	223	227	255	226	255	251	254
238	254	224	228	255	227	255	251	254
239	254	225	229	255	227	255	251	254
240	254	226	230	255	228	255	251	254
241	254	227	230	255	229	255	251	254
242	255	228	231	255	230	255	251	254
243	255	229	232	255	230	255	251	254
244	255	230	233	255	231	255	252	254
245	255	231	234	255	231	255	252	254
246	255	232	235	255	232	255	252	254
247	255	233	236	255	232	255	252	254
248	255	235	237	255	233	255	252	254
249	255	236	238	255	234	255	252	254
250	255	238	240	255	235	255	252	255
251	255	239	241	255	235	255	252	254
252	255	241	243	255	236	255	252	254
253	255	243	245	255	237	255	252	254
254	255	246	247	255	239	255	253	255
255	255	246	247	255	239	255	253	255

在 token 大于等于 5 时, 需要计算 extra。

token 等于 5 时, extra 使用 svac2_cat1_prob 概率表, 见表 91。

表 91 svac2_cat1_prob

ctx	binIdx	
	0	1
0	159	

token 等于 6 时, extra 使用 svac2_cat2_prob 概率表, 见表 92。

表 92 svac2_cat2_prob

ctx	binIdx	
	0	1
0	165	145

token 等于 7 时, extra 使用 svac2_cat3_prob 概率表, 见表 93。

表 93 svac2_cat3_prob

ctx	binIdx		
	0	1	2
0	173	148	140

token 等于 8 时, extra 使用 svac2_cat4_prob 概率表, 见表 94。

表 94 svac2_cat4_prob

ctx	binIdx			
	0	1	2	3
0	176	155	140	135

token 等于 9 时, extra 使用 svac2_cat5_prob 概率表, 见表 95。

表 95 svac2_cat5_prob

ctx	binIdx				
	0	1	2	3	4
0	180	157	141	134	130

token 等于 10 时, extra 使用 svac2_cat6_prob 概率表, 见表 96。

表 96 svac2_cat6_prob

ctx	binIdx													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	254	254	254	252	249	243	230	196	177	153	140	133	130	129

下述语法元素使用固定概率值 128 进行解析: tx_mode, tx_mode_delta, coeff_update_prob_flag, not_single_ref, not_compound_ref, tile_idx₅₇₄, sao_merge_flag, sao_merge_type, sao_mode, sao_type, sao_start_band, sao_offset_abs, sao_offset_sign, sao_edge_type, sao_edge_offset[compIdx][0], sao_edge_offset[compIdx][1], sao_edge_offset[compIdx][2], sao_edge_offset[compIdx][3], coeff_sign, mpm_idx0, mpm_idx1, rem_pred_intra_mode, chroma_intra_mode。

5.4.2.2.2 概率表的更新

5.4.2.2.2.1 svac2_diff_update_prob

每帧图像的编码片开始解码前,会根据条件选择是否对语法元素的概率表进行更新。svac2_diff_update_prob 部分用以更新 tx_probs、switchable_interp_prob、newmv_prob、zeromv_prob、refmv_prob、comp_inter_prob、single_ref_prob、comp_ref_prob、coef_probs、skip_probs、alf_ctu_enable_prob、intra_inter_prob、partition_prob 和 abs_delta_q_prob 的概率表。

此过程的输入为原始的概率值 prob,输出为更新后的概率值 prob。

过程如下:

```
svac2_diff_update_prob(prob) {
    update_prob = ae(252)
    if(update_prob == 1){
        deltaProb = decode_term_subexp();
        prob = inv_remap_prob(deltaProb, prob);
    }
}
```

首先使用固定概率 252 通过 ae(v)解码出 1bit。若为 0 则无概率更新,采用默认初始化的概率值;若为 1 则表示概率做了更新,继续解码并计算出所采用的新概率。

概率更新分为两步,第一步解码得到概率数组的索引 deltaProb:

```
decode_term_subexp() {
    bit = ae(128)
    if ( bit == 0 ) {
        sub_exp_val = u(4)
        return sub_exp_val
    }
    bit = ae(128)
    if ( bit == 0 ) {
        sub_exp_val_minus_16 = u(4)
        return (sub_exp_val_minus_16 + 16)
    }
    bit = ae(128)
    if ( bit == 0 ) {
        sub_exp_val_minus_32 = u(5)
        return (sub_exp_val_minus_32 + 32)
    }
    sub_exp_val = u(7)
    if (sub_exp_val < 65) {
        return (sub_exp_val + 64)
    }
    bit = ae(128)
    return (sub_exp_val << 1) - 1 + bit
}
```

第二步根据索引 deltaProb, 得到最终的概率:

```
inv_remap_prob(deltaProb, prob) {
    m = p
    v = delp
    v = inv_map_table[v]
    m--
    if ((m << 1) <= 255)
        m = 1 + inv_recenter_nonneg(v, m)
    else
        m = 255 - inv_recenter_nonneg(v, 255 - 1 - m)
    return m
}
```

inv_map_table 的定义如下:

```
inv_map_table[254] =
{
    6, 19, 32, 45, 58, 71, 84, 97, 110, 123, 136, 149, 162, 175, 188,
    201, 214, 227, 240, 253, 0, 1, 2, 3, 4, 5, 7, 8, 9, 10,
    11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26,
    27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
    43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 59,
    60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 72, 73, 74, 75,
    76, 77, 78, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 91,
    92, 93, 94, 95, 96, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
    108, 109, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 124,
    125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 137, 138, 139, 140,
    141, 142, 143, 144, 145, 146, 147, 148, 150, 151, 152, 153, 154, 155, 156,
    157, 158, 159, 160, 161, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172,
    173, 174, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 189,
    190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 202, 203, 204, 205,
    206, 207, 208, 209, 210, 211, 212, 213, 215, 216, 217, 218, 219, 220, 221,
    222, 223, 224, 225, 226, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237,
    238, 239, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252
};
```

inv_recenter_nonneg(v,m)计算如下:

```
if (v > 2 * m)
    return v
if (v & 1)
    return m - ((v + 1) >> 1)
return m + (v >> 1)
```

5.4.2.2.2 update_mv_probs

每帧图像的编码片开始解码前,会根据条件选择是否对语法元素的概率表进行更新。update_mv_probs 部分用以更新 mv_joint_probs、mv_sign_probs、mv_bits_probs、mv_class0_bit_probs、mv_class_

probs、mv_class0_fr_probs、mv_class0_hp_probs、mv_fr_probs 和 mv_hp_probs 的概率表。

此过程的输入为原始的概率值 prob，输出为更新后的概率值 prob。

过程如下：

```
update_mv_probs(prob) {
    for(i = 0; i < n; i++) {
        update_prob = ae(252)
        if(update_prob == 1){
            prob[i] = (u(7) << 1) | 1
        }
    }
}
```



n 为概率表的元素个数。

首先使用固定概率 252 通过 ae(v) 解码出 1bit。若为 0 则无概率更新，采用默认初始化的概率值；若为 1 则表示概率做了更新，继续解码并计算出所采用的新的概率。

5.4.2.2.3 初始化算术码解码器

range、count、value、buffer、buffer_end 为算术解码器的变量。range 的位宽为 8 比特，value 位宽为 32 比特。算术解码器的初始化用如下：

第一步，变量初始化：

```
buffer = 码流地址
buffer_end = 码流地址 + 码流字节长度
range = 255
value = 0
count = -8
```

第二步，执行算术解码器的重整化过程：

```
buffer_t = buffer
buffer_start = buffer
value_t = value
count_t = count
loop_end = 0
shift = 16 - count
bytes_left = buffer_end - buffer
bits_left = bytes_left * 8
x = shift + 8 - bits_left
if(x >= 0){
    count_t += 0x40000000
    loop_end = x
}
if(x < 0 || bits_left){
    while (shift >= loop_end){
        count_t += 8
        value_t |= *buffer_t++ << shift
        shift -= 8
    }
}
```

```

    }
}

buffer += buffer_t - buffer_start
value = value_t
count = count_t

```

第三步,使用固定概率 128 通过 ae(v)解码 1bit,如等于 0,算术解码器初始化成功,否则算术解码器初始化失败。

5.4.2.3 二值化

各语法元素的二值化过程如下:

hasRows 与 hasCols 均等于 1 时,语法元素 partition 的取值与二进制位串的对应关系见表 97。hasRows 等于 0 且 hasCols 等于 1 时,语法元素 partition 的取值与二进制位串的对应关系见表 98。hasRows 等于 1 且 hasCols 等于 0 时,语法元素 partition 的取值与二进制位串的对应关系见表 99。

表 97 hasRows 和 hasCols 均等于 1 时 partition 与二进制位串的关系

partition	二进制位串		
PARTITION_NONE	0		
PARTITION_HORZ	1	0	
PARTITION_VERT	1	1	0
PARTITION_SPLIT	1	1	1
binIdx	0	1	2

表 98 hasCols 等于 1 时 partition 与二进制位串的关系

partition	二进制位串
PARTITION_HORZ=1	1
PARTITION_SPLIT=3	0
binIdx	0

表 99 hasRows 等于 1 时 partition 与二进制位串的关系

partition	二进制位串
PARTITION_VERT=2	1
PARTITION_SPLIT=3	0
binIdx	0

语法元素 segment_id 的值与二进制位串的关系见表 100。

表 100 segment_id 与二进制位串的关系

segment_id	二进制位串		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
binIdx	0	1	2

seg_id_predicted 与二进制位串的关系见表 101。

表 101 seg_id_predicted 与二进制位串的关系

seg_id_predicted 取值	二进制位串
0	0
1	1
binIdx	0

语法元素 tx_size 的值与二进制位串的关系见表 102。

表 102 变换块大小语法元素与二进制位串的关系

tx_size	二进制位串		
TX_4×4	0		
TX_8×8	1	0	
TX_16×16	1	1	0
TX_32×32	1	1	1
binIdx	0	1	2

skip_flag 与二进制位串的关系见表 103。

表 103 skip_flag 与二进制位串的关系

skip_flag 取值	二进制位串
0	0
1	1
binIdx	0

inter_block 与二进制位串的关系见表 104。

表 104 inter_block 与二进制位串的关系

inter_block 取值	二进制位串
0	0
1	1
binIdx	0

prev_intra_luma_pred_flag 与二进制位串的关系见表 105。

表 105 prev_intra_luma_pred_flag 与二进制位串的关系

prev_intra_luma_pred_flag 取值	二进制位串
0	0
1	1
binIdx	0

uv_fflow_y_flag 与二进制位串的关系见表 106。

表 106 uv_fflow_y_flag 与二进制位串的关系

uv_fflow_y_flag 取值	二进制位串
0	0
1	1
binIdx	0

帧间参考帧模式 REFERENCE_MODE 语法元素与二进制位串的关系见表 107。

表 107 block_reference_mode 与二进制位串的关系

block_reference_mode	二进制位串	
SINGLE_REFERENCE = 0	0	
COMPOUND_REFERENCE = 1	1	0
REFERENCE_MODE_SELECT = 2	1	1
binIdx	0	1

ref_frame 值与二进制位串的关系见表 108 和表 109。

如果 block_reference_mode 等于 SINGLE_REFERENCE, 那么参考帧类型 ref_frame 二值化见表 108; 如果 block_reference_mode 等于 COMPOUND_REFERENCE, 那么参考帧类型 ref_frame 二值化见表 109。

表 108 SINGLE_REFERENCE 模式下 ref_frame 与二进制位串的关系

ref_per_frame	ref_frame	二进制位串			
2	DYNAMIC_REF	0			
	STATIC_REF	1			
3	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	OPTIONAL_REF	1	1		
4	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	OPTIONAL_REF	1	1	0	
	DYNAMIC_REF_1	1	1	1	
5	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	OPTIONAL_REF	1	1	0	
	DYNAMIC_REF_1	1	1	1	0
	DYNAMIC_REF_2	1	1	1	1
binIdx		0	1	2	3



表 109 COMPOUND_REFERENCE 模式下 ref_frame 与二进制位串的关系

ref_per_frame	ref_frame	二进制位串			
3	DYNAMIC_REF	0			
	STATIC_REF	1			
4	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	DYNAMIC_REF_1	1	1		
5	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	DYNAMIC_REF_1	1	1	0	
	DYNAMIC_REF_2	1	1	1	
binIdx		0	1	2	

mv_mode 的值与二进制位串的关系见表 110。

表 110 mv_mode 与二进制位串的关系

mv_mode	二进制位串		
NEARESTMV	1	1	0

表 110 (续)

mv_mode	二进制位串		
NEARMV	1	1	1
ZEROMV	1	0	
NEWMV	0		
binIdx	0	1	2

interp_filter_mode 的值与二进制位串的关系见表 111。

表 111 interp_filter_mode 与二进制位串的关系

interp_filter_mode	二进制位串		
0 (Regular)	0		
1 (Smooth-1)	1	0	
2 (Sharp)	1	1	
3 (Smooth-2)	1	1	1
binIdx	0	1	2

编码块尺寸小于 8×8 或 is_compound 等于 1 时, mv_joint 的值与二进制位串的关系见表 112。

表 112 mv_joint 与二进制位串的关系-1

mv_joint	二进制位串		
MV_JOINT_HNZVNZ = 0 Both components nonzero	0		
MV_JOINT_HNZVZ = 1, Vert zero, hor nonzero	1	0	
MV_JOINT_HZVNZ = 2, Hor zero, vert nonzero	1	1	0
MV_JOINT_ZERO = 3 Zero vector	1	1	1
binIdx	0	1	2

如果编码块尺寸不小于 8×8 且 is_compound 等于 0 时, mv_joint 的值与二进制位串的关系见表 113。

表 113 mv_joint 与二进制位串的关系-2

mv_joint	二进制位串	
MV_JOINT_HNZVNZ = 0 Both components nonzero	0	

表 113 (续)

mv_joint	二进制位串	
MV_JOINT_HNZVZ = 1, Vert zero, hor nonzero	1	0
MV_JOINT_HZVNZ = 2, Hor zero, vert nonzero	1	1
binIdx	0	1

mvd_sign_0 和 mvd_sign_1 与二进制位串的关系见表 114。

表 114 mvd_sign_0 和 mvd_sign_1 与二进制位串的关系

mvd_sign_0	mvd_sign_1	二进制位串
0	0	0
1	1	1
binIdx		0

mvd_value_0 和 mvd_value_1 的二值化, 使用 ae(v) 解析非零部分的整数样点部分所对应的 mv_class, 其取值与二进制位串的关系见表 115。

表 115 mv_class 与二进制位串的关系

mv_class	二进制位串						
MV_CLASS_0 = 0	0						
MV_CLASS_1 = 1	1	0					
MV_CLASS_2 = 2	1	1	0	0			
MV_CLASS_3 = 3	1	1	0	1			
MV_CLASS_4 = 4	1	1	1	0	0		
MV_CLASS_5 = 5	1	1	1	0	1		
MV_CLASS_6 = 6	1	1	1	1	0		
MV_CLASS_7 = 7	1	1	1	1	1	0	0
MV_CLASS_8 = 8	1	1	1	1	1	0	1
MV_CLASS_9 = 9	1	1	1	1	1	1	0
MV_CLASS_10 = 10	1	1	1	1	1	1	1
binIdx	0	1	2	3	4	5	6

而后根据 mv_class 的值进行 ae(v) 解析其整数样点位置偏移 d, d 二值化后的二进制串的长度见表 116, 取值为 d 的二进制补码值, 高位在前。

表 116 d 的解码长度与 mv_class 的关系

mv_class	d 的解码长度
MV_CLASS_0 = 0	1
MV_CLASS_1 = 1	1
MV_CLASS_2 = 2	2
MV_CLASS_3 = 3	3
MV_CLASS_4 = 4	4
MV_CLASS_5 = 5	5
MV_CLASS_6 = 6	6
MV_CLASS_7 = 7	7
MV_CLASS_8 = 8	8
MV_CLASS_9 = 9	9
MV_CLASS_10 = 10	10

最后 ae(v) 解析非零部分的分数样点的位置偏移 fr,fr 的值与二进制位串的关系见表 117。

表 117 fr 与二进制位串的关系

fr	二进制位串		
0	0		
1	1	0	
2	1	1	0
3	1	1	1
binIdx	0	1	2

hp 与二进制位串的关系见表 118。

表 118 hp 与二进制位串的关系

usehp	hp	二进制位串
0	1	无
0	1	无
1	0	0
1	1	1

usehp 等于 1 表示 MV 使用高精度部分。

由 ae(v) 解析出的 mv_class 计算 MV 偏移的基准:

$mv_class_base = mv_class ? 2 \ll (mv_class + 2) : 0$

由 ae(v) 解析出的 d 和 fr 计算 MV 在基准上的偏移, 其中 hp 高精度部分, 默认为 1:

$offset = (d \ll 3) | (fr \ll 1) | hp$

mv_value_0 或 mv_value_1 的值即等于 $(mv_class_base + offset + 1)$ 。

dqp_abs 与二进制位串的关系见表 119。当 dqp_abs 大于 7 时, dqp_abs 减 7 后的部分用 0 阶指数哥伦布码。

表 119 dqp_abs 与二进制位串的关系

dqp_abs		二进制位串						
0	0							
1	1	0						
2	1	1	0					
3	1	1	1	0				
4	1	1	1	1	0			
5	1	1	1	1	1	0		
6	1	1	1	1	1	1	0	
7	1	1	1	1	1	1	1	1
binIdx	0	1	2	3	4	5	6	

dqp_sign 与二进制位串的关系见表 120。

表 120 dqp_sign 与二进制位串的关系

dqp_sign		二进制位串
0		0
1		1
binIdx		0

alf_ctu_enable 与二进制位串的关系见表 121。

表 121 alf_ctu_enable 与二进制位串的关系

alf_ctu_enable		二进制位串
0		0
1		1
binIdx		0

块系数 coeff_value 的二值化,包括 token 和 extra 两个部分。当 coeff_value 等于 EOB 或者小于 5 时,coeff_value 等于 token。当 coeff_value 大于 5 时,对应关系如下:

$$\text{coeff_value} = 3 + 2^{\lceil (\text{token} - 4) \rceil} + \text{extra}$$



连续 $N(N \geq 0)$ 个等于 0 的系数与一个非零系数可以用一组 token 和 extra 表示。

token 的第一个二进制位为 EOB 指示位,如果该位为 0,表示对应的 token 为 EOB,见表 122。EOB 解析使用 binIdx 等于 0 的 coef_probs 概率表。

EOB 指示位后为 0 系数指示位,长度为 N,对应的 token 等于 0,见表 123。判断 token 是否为 0 的解析,使用 binIdx 等于 1 的 coef_probs 概率表。

表 122 EOB 与二进制位串的关系

token	token EOB 指示二进制位
EOB	0
非 EOB	1
binIdx	0

表 123 0 系数指示位与二进制位串的关系

token	token 0 系数指示二进制位
0	0
非 0	1
binIdx	0

token 与二进制位串的对应关系见表 124, 其中, binIdx 等于 0 的位为 0 系数指示位。解析 binIdx 等于 1 的位使用 binIdx 等于 2 的 coef_probs 概率表, 之后的位解析使用 svac2_pareto8_full 概率表。

表 124 token 与二进制位串的关系

token	二进制位串						
	0	1	0	0	0	1	0
0	0						
1	1	0					
2	1	1	0	0			
3	1	1	0	1	0		
4	1	1	0	1	1		
5	1	1	1	0	0		
6	1	1	1	0	1		
7	1	1	1	1	0	0	
8	1	1	1	1	0	0	1
9	1	1	1	1	1	1	0
10	1	1	1	1	1	1	1
binIdx	0	1	2	3	4	5	

extra 二值化的位串长度与 token 取值的对应关系见表 125, 取值为 extra 的二进制补码值, 高位在前。

表 125 extra 二值化的位串长度与 token 的对应关系

token	extra 二进制位串长度
0	无
1	无
2	无

表 125 (续)

token	extra 二进制位串长度
3	无
4	无
5	1
6	2
7	3
8	4
9	5
10	14

tx_mode_delta,coeff_update_prob_flag,not_single_ref,not_compound_ref,sao_merge_flag,sao_merge_type,sao_mode,sao_type,sao_offset_sign,coeff_sign,mpm_idx0,rem_pred_intra_mode,sao_edge_offset[compIdx][1],sao_edge_offset[compIdx][2]的取值等于0时,二进制位串为‘0’;等于1时,二进制位串为‘1’。

tx_mode,sao_edge_type,mpm_idx1,chroma_intra_mode 的二进制位串的长度为 2,取值为语法元素值的二进制补码值,高位在前。

sao_start_band 的二进制位串的长度为 5,取值为语法元素值的二进制补码值,高位在前。
tile_idx,sao_offset_abs,sao_edge_offset[compIdx][0],sao_edge_offset[compIdx][3]的二进制位串等于其值对应的 0 阶无符号指数哥伦布码。

5.4.2.4 二进制位串解析

5.4.2.4.1 概述

本过程的输入是 5.4.2.2 的初始化 ctx 索引所对应的概率状态以及算术解码引擎中的内部状态变量。

本过程的输出是二进制码值。

5.4.2.4.2 相关变量与含义

相关变量与含义如下:

- block_size:编码块的尺寸;
- AvailU:为 1,上边块可用;为 0,上边块不可用;
- AvailL:为 1,左边块可用;为 0,左边块不可用;
- AboveIntra:为 1,上边块为 Intra;为 0,上边块非 Intra;
- LeftIntra:为 1,左边块为 Intra;为 0,左边块非 Intra;
- AboveSegIdPredicted:上边块的 seg_id_predicted;
- LeftSegIdPredicted:左边块的 seg_id_predicted;
- AboveSkip:为 1,上边块为 skip;为 0,上边块非 skip;
- LeftSkip:为 1,左边块为 skip;为 0,左边块非 skip;
- maxTxSize:最大变换尺寸;
- AboveTxSize:上边块的变换尺寸;

——LeftTxSize:左边块的变换尺寸;
 ——idy:子块的纵向坐标;
 ——idx:子块的横向坐标;
 ——AbovePredMode:上块的预测模式;
 ——LeftPredMode:左块的预测模式;
 ——AboveSingle:为 1,上块为 single reference 模式;为 0,上边块为 compound reference 模式;
 ——LeftSingle:为 1,左块为 single reference 模式;为 0,左块为 compound reference 模式;
 ——CompFixedRef:参考帧预测时根据上下文得到的参考帧类型;
 ——CurSelectRef:single reference 模式下,对于 binIdx 等于 3 和 4 的情况,值分别为 OPTIONAL_REF 和 DYNAMIC_REF_1;
 ——AboveInter:为 1,上边块为 inter;为 0,上边块非 inter;
 ——LeftInter:为 1,左边块为 inter;为 0,左边块非 inter;
 ——AboveInterpFilter:上边块的 interp_filter_mode;
 ——LeftInterpFilter:左边块的 interp_filter_mode;
 ——tx_size:变换块尺寸;
 ——c:coefs 系数的计数值;
 ——plane:0 表示亮度,1 表示色度;
 ——is_plane_y:为 1 表示为亮度;为 0 表示色度;
 ——tx_type:变换类型;
 ——AboveAlfEnable:为 1,上块自适应滤波有效;为 0,则无效;
 ——LeftAlfEnable:为 0,左块自适应滤波有效;为 0,则无效;
 ——above_seg_context[]:上行块的 ctx;
 ——left_seg_context[]:左列块的 ctx;
 ——AboveMode[]:上块的预测模式;
 ——LeftMode[]:左块的预测模式;
 ——CurMode[]:当前块的预测模式;
 ——AboveRefFrame[]:上块的参考帧;
 ——LeftRefFrame[]:左块的参考帧;
 ——mode_context[]:参考帧模式的上下文索引;
 ——CurRefFrame[]:当前块的参考帧;
 ——AboveNonzeroContext[]:上行的非零系数上下文索引;
 ——LeftNonzeroContext[]:左列的非零系数上下文索引。

5.4.2.4.3 treeIdx 的计算过程

一些语法元素在查找概率表时,需要使用 treeIdx 和对应数组计算出对应的 binIdx。

这些语法元素包括:partition、segment_id、mv_joint、mvd_value_0 和 mvd_value_1、interp_filter_mode、dqp_abs 及 token。

partition:

在 hasRows 和 hasCols 均等于 1 时,使用数组 partition_tree,计算 binIdx 所对应的 treeIdx。

partition_tree[6] = {-0, 2, -1, 4, -2, -3}

segment_id: 使用 segment_tree:

segment_tree[14] = {2, 4, 6, 8, 10, 12, 0, -1, -2, -3, -4, -5, -6, -7}

interp_filter_mode: 使用 interp_filter_tree:

interp_filter_tree[6] = {-0, 2, 4, -2, -1, -3}

mv_joint:树的选择依赖于 is_compound:

——如果 is_compound 等于 1,选择 mv_joint_tree。

——否则如果 is_compound 等于 0,选择 mv_joint_tree2。

mv_joint_tree[6] = {-0, 2, -1, 4, -2, -3}

mv_joint_tree2[4] = {-0, 2, -1, -2}

mvd_value_0 和 **mvd_value_1** 主要有 mv_class、mv_class0_fp 和 mv_fp 需要通过 treeIdx 计算 binIdx。



mv_class:使用 mv_class_tree:

mv_class_tree[20] = {-0, 2, -1, 4, 6, 8, -2, -3, 10, 12, -4, -5, -6, 14, 16, 18, -7, -8, -9, -10}

mv_class0_fp 和 **mv_fp:**均使用 mv_fp_tree:

mv_fp_tree[6] = {-0, 2, -1, 4, -2, -3}

token:

在使用 svac2_pareto8_full[ctx][treeIdx]概率表计算时,需要使用 coeff_subtree_high:

coeff_subtree_high[16] = {2, 6, -2, 4, -3, -4, 8, 10, -5, -6, 12, 14, -7, -8, -9, -10}

上述变量解析时,treeIdx 通过如下计算得到:

设 T 为整型数组组成的树,P 为变量所对应的概率表。初始令 treeIdx 等于 0,n 等于 0。

以 treeIdx 索引获得的概率按照 5.4.2.4.5 解析出一个二进制位 B。如果语法元素解析完成,则本过程结束;否则令 n 等于 T[n + B], treeIdx 等于 n>>1,继续获取下一个二进制位的概率进行解析。

5.4.2.4.4 概率选择过程

此过程的输入为语法元素的 binIdx 及 treeIdx,输出为解析该语法元素二进制位所用的概率值 prob。

以下为各个语法元素的概率计算:

alf_ctu_enable:

概率等于 alf_ctu_enable_prob[ctx],其中 ctx 的计算如下:

```
if(AvailL && AvailU)
    ctx = AboveAlfEnable && LeftAlfEnable ? 3: AboveAlfEnable || LeftAlfEnable
else if(AvailL || AvailU)
    ctx = 2 * (AvailL ? LeftAlfEnable : AboveAlfEnable)
else
    ctx = 1
```

partition:

如果 frame_type 等于 0,概率值等于 intra_partition_probs[ctx][treeIdx];否则,概率值等于 inter_partition_probs[ctx][treeIdx]。

在 hasRows 和 hasCols 均等于 1 的情况下,索引二进制位对应的概率的 treeIdx 的计算见 5.4.2.4.3 节;当 hasCols 等于 1 且 hasRows 等于 0 时,treeIdx 等于 1;当 hasRows 等于 1 且 hasCols 等于 0 时,treeIdx 等于 2。

索引 ctx 的值根据如下方式计算得到:

```
above = 0
left = 0
```

```

bsl = mi_width_log2_lookup[bsize]
bs = 1 << bsl
for (i = 0; i < bs; i + +){
    above |= above_seg_context [i]
    left |= left_seg_context [i]
}
above = (above & bs) >0
left = (left & bs)>0
ctx = (left * 2 + above) + bsl * 4
mi_width_log2_lookup[BLOCK_SIZES] ={0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4}
segment_id:

```

概率等于 seg_tree_probs[treeIdx], 其中 treeIdx 的计算见 5.4.2.4.3。

seg_id_predicted:

概率等于 seg_pred_probs[ctx], 其中 ctx 的计算如下：

```

ctx = 0
if(AvailU)
    ctx += AboveSegIdPredicted
if(AvailL)
    ctx += LeftSegIdPredicted

```

skip_flag:

概率等于 skip_prob[ctx], 其中 ctx 的计算如下：

```

ctx = 0
if(AvailU)
    ctx += AboveSkip
if(AvailL)
    ctx += LeftSkip

```

inter_block:

概率等于 intra_inter_prob[ctx], 其中 ctx 的计算如下：

```

if(AvailU && AvailL)
    ctx = (LeftIntra && AboveIntra) ? 3 : LeftIntra || AboveIntra
else if ( AvailU || AvailL )
    ctx = 2 * (AvailU ? AboveIntra : LeftIntra)
else
    ctx = 0

```

tx_size:

概率值等于 tx_probs[MAX_TX_SIZE][ctx][binIdx] 得到, 其中 ctx 的计算如下：

```

above = maxTxSize
left = maxTxSize
if ( AvailU && ! AboveSkip)
    above = AboveTxSize
if ( AvailL && ! LeftSkip )
    left = LeftTxSize
if ( ! AvailL )

```



```

    left = above
    if ( ! AvailU )
        above = left
    ctx = (above + left) > maxTxSize

```

maxTxSize 由编码树划分的预测块尺寸决定,见表 126。

表 126 maxTxSize 与预测块尺寸的关系

maxTxSize	partitioned block size
TX_4×4	BLOCK_4×4
TX_4×4	BLOCK_4×8
TX_4×4	BLOCK_8×4
TX_8×8	BLOCK_8×8
TX_8×8	BLOCK_8×16
TX_8×8	BLOCK_16×8
TX_16×16	BLOCK_16×16
TX_16×16	BLOCK_16×32
TX_16×16	BLOCK_32×16
TX_32×32	BLOCK_32×32
TX_32×32	BLOCK_32×64
TX_32×32	BLOCK_64×32
TX_32×32	BLOCK_64×64
TX_32×32	BLOCK_64×128
TX_32×32	BLOCK_128×64
TX_32×32	BLOCK_128×128

prev_intra_luma_pred_flag:

概率等于 mpm_flag_probs[ctx],其中 ctx 的计算如下:

```

if(AbovePredMode == LeftPredMode) {
    if (LeftPredMode>2)
        ctx = 0
    else
        ctx = 1
} else {
    mode0 = LeftPredMode>AbovePredMode ? AbovePredMode : LeftPredMode
    mode1 = LeftPredMode>AbovePredMode ? LeftPredMode : AbovePredMode
    if (mode0>2) {
        if(mode1-mode0<4)
            ctx = 2
        else if(mode1-mode0<12)
            ctx = 3
    }
}

```

```

        else
            ctx = 4
    } else {
        if(model == 3)
            ctx = 5
        else
            ctx = 6
    }
}
}

```

uv_fflow_y_flag:

概率固定为 175。

block_reference_mode:

概率等于 comp_inter_prob[ctx], 其中 ctx 的计算如下:

```

if ( AvailU && AvailL ) {
    if ( AboveSingle && LeftSingle )
        ctx = (AboveRefFrame[0] == CompFixedRef) ^ (LeftRefFrame[0] == CompFixedRef)
    else if ( AboveSingle )
        ctx = 2 + (AboveRefFrame[0] == CompFixedRef || AboveIntra)
    else if ( LeftSingle )
        ctx = 2 + (LeftRefFrame[0] == CompFixedRef || LeftIntra)
    else
        ctx = 4
} else if ( AvailU ) {
    if ( AboveSingle )
        ctx = AboveRefFrame[0] == CompFixedRef
    else
        ctx = 3
} else if ( AvailL ) {
    if ( LeftSingle )
        ctx = LeftRefFrame[0] == CompFixedRef
    else
        ctx = 3
} else {
    ctx = 1
}

```

ref_frame

根据 block_reference_mode 分为 compound reference 模式和 single reference 模式。

compound reference 模式一共有 3 个 binIdx 的 ctxIdx 要计算;single reference 模式一共有 4 个 binIdx 的 ctxIdx 要计算。

compound reference 模式下 binIdx 等于 0 的情况:

概率等于 comp_ref_prob[ctx][0], ctx 的计算如下:

```

VarRefIdx = 0
if ( AvailU && AvailL ) {

```

```

if ( AboveIntra && LeftIntra ) {
    ctx = 2
} else if ( LeftIntra ) {
    if ( AboveSingle )
        ctx = 1 + 2 * ((AboveRefFrame[0] == STATIC_REF) ||
            (AboveRefFrame[0] == DYNAMIC_REF_1) || (AboveRefFrame[0] == DYNAMIC_REF_2))
    else
        ctx = 1 + 2 * ((AboveRefFrame[VarRefIdx] == STATIC_REF) ||
            (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_1) ||
            (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_2))
} else if ( AboveIntra ) {
    if ( LeftSingle )
        ctx = 1 + 2 * ((LeftRefFrame[0] == STATIC_REF) ||
            (LeftRefFrame[0] == DYNAMIC_REF_1) || (LeftRefFrame[0] == DYNAMIC_REF_2))
    else
        ctx = 1 + 2 * ((LeftRefFrame[VarRefIdx] == STATIC_REF) ||
            (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_1) ||
            (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_2))
} else {
    vrfa = AboveSingle ? AboveRefFrame[0] : AboveRefFrame[VarRefIdx]
    vrf1 = LeftSingle ? LeftRefFrame[0] : LeftRefFrame[VarRefIdx]
    if ( vrfa == vrf1 && ((vrfa == STATIC_REF) || (vrfa == DYNAMIC_REF_1) || (vrfa
== DYNAMIC_REF_2)) ) {
        ctx = 0
    } else if ( LeftSingle && AboveSingle ) {
        if ((vrfa == OPTIONAL_REF && vrf1 == DYNAMIC_REF || (vrf1 == OPTIONAL_REF && vrfa == DYNAMIC_REF))
            ctx = 4
        else if ( vrfa == vrf1 || (vrf1 == DYNAMIC_REF && vrfa == DYNAMIC_REF) )
            ctx = 3
        else
            ctx = 1
    } else if ( LeftSingle || AboveSingle ) {
        vrfc = LeftSingle ? vrfa : vrf1
        rfs = AboveSingle ? vrfa : vrf1
        if (((vrfc == STATIC_REF) || (vrfc == DYNAMIC_REF_1) || (vrfc == DYNAMIC_REF_2)) &&
            !((vrfc == STATIC_REF) || (vrfc == DYNAMIC_REF_1) || (vrfc == DYNAMIC_REF_2)))
            ctx = 1
        else if (((rfs == STATIC_REF) || (rfs == DYNAMIC_REF_1) || (rfs == DYNAMIC_REF_2)) && !((vrfc == STATIC_REF) || (vrfc == DYNAMIC_REF_1) || (vrfc == DYNAMIC_REF_2)))
            ctx = 2
        else
            ctx = 4
    }
}

```

```

        } else if (((vrfa == STATIC_REF) || (vrfa == DYNAMIC_REF_1) || (vrfa == DYNAMIC_REF_2)) && ((vrfl == STATIC_REF) || (vrfl == DYNAMIC_REF_1) || (vrfl == DYNAMIC_REF_2))) {
            ctx = 4
        } else {
            ctx = 2
        }
    }
} else if (AvailU) {
    if (AboveIntra) {
        ctx = 2
    } else {
        if (AboveSingle)
            ctx = 3 * (!((AboveRefFrame[0] == STATIC_REF) || (AboveRefFrame[0] == DYNAMIC_REF_1) || (AboveRefFrame[0] == DYNAMIC_REF_2)))
        else
            ctx = 4 * (!((AboveRefFrame[VarRefIdx] == STATIC_REF) || (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_1) || (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_2)))
    }
} else if (AvailL) {
    if (LeftIntra) {
        ctx = 2
    } else {
        if (LeftSingle)
            ctx = 3 * (!((LeftRefFrame[0] == STATIC_REF) || (LeftRefFrame[0] == DYNAMIC_REF_1) || (LeftRefFrame[0] == DYNAMIC_REF_2)))
        else
            ctx = 4 * (!((LeftRefFrame[VarRefIdx] == STATIC_REF) || (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_1) || (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_2)))
    }
} else {
    ctx = 2
}

```

compound reference 模式下 binIdx 等于 1 的情况：

概率等于 comp_ref_prob[ctx][1], ctx 的计算如下：

```

VarRefIdx = 0
if (AvailU && AvailL) {
    if (AboveIntra && LeftIntra) {
        ctx = 2
    } else if (LeftIntra) {
        if (AboveSingle)
            ctx = 1 + 2 * (AboveRefFrame[0] == STATIC_REF)
        else
            ctx = 1 + 2 * (AboveRefFrame[VarRefIdx] == STATIC_REF)
    }
}

```

```

} else if ( AboveIntra ) {
    if ( LeftSingle )
        ctx = 1 + 2 * (LeftRefFrame[0] == STATIC_REF)
    else
        ctx = 1 + 2 * (LeftRefFrame[VarRefIdx] == STATIC_REF)
} else {
    vrfa = AboveSingle ? AboveRefFrame[0] : AboveRefFrame[VarRefIdx]
    vrf1 = LeftSingle ? LeftRefFrame[0] : LeftRefFrame[VarRefIdx]
    if ( vrfa == vrf1 && STATIC_REF == vrfa ) {
        ctx = 0
    } else if ( LeftSingle && AboveSingle ) {
        if ( (vrfa == STATIC_REF && vrf1 == STATIC_REF) )
            ctx = 4
        else if ( vrfa == STATIC_REF || vrf1 == STATIC_REF )
            ctx = 3
        else if ( vrfa == DYNAMIC_REF || vrf1 == DYNAMIC_REF )
            ctx = 1 + (vrfa == vrf1)
        else
            ctx = 1
    } else if ( LeftSingle || AboveSingle ) {
        vrfc = LeftSingle ? vrfa : vrf1
        rfs = AboveSingle ? vrfa : vrf1
        if ( vrfc == STATIC_REF && rfs == STATIC_REF )
            ctx = 4
        else if ( vrfc == STATIC_REF && rfs != STATIC_REF )
            ctx = 3
        else if ( vrfc != STATIC_REF && rfs == STATIC_REF )
            ctx = 2 + (!vrfc == DYNAMIC_REF)
        else {
            if ( vrfc == DYNAMIC_REF || rfs == DYNAMIC_REF )
                ctx = 2
            else
                ctx = 1
        }
    } else {
        if ( vrfa == STATIC_REF && vrf1 == STATIC_REF )
            ctx = 4
        else if ( vrfa == STATIC_REF || vrf1 == STATIC_REF )
            ctx = 3
        else if ( vrfa != STATIC_REF && vrf1 != STATIC_REF )
            ctx = 1
        else
            ctx = 2
    }
}

```

```

        }
    }

} else if ( AvailU ) {
    if ( AboveIntra ) {
        ctx = 2
    } else {
        if ( ! AboveSingle )
            ctx = 4 * (AboveRefFrame[VarRefIdx] != STATIC_REF)
        else {
            if(AboveRefFrame[0] == STATIC_REF)
                ctx = 3
            else if(AboveRefFrame[0] == DYNAMIC_REF)
                ctx = 2
            else
                ctx = 1
        }
    }
} else if ( AvailL ) {
    if ( LeftIntra ) {
        ctx = 2
    } else {
        if ( ! LeftSingle)
            ctx = 4 * (LeftRefFrame [VarRefIdx] != STATIC_REF)
        else {
            if(LeftRefFrame [0] == STATIC_REF)
                ctx = 3
            else if(LeftRefFrame [0] == DYNAMIC_REF)
                ctx = 2
            else
                ctx = 1
        }
    }
} else {
    ctx = 2
}

```

compound reference 模式下 binIdx 等于 2 的情况：

概率等于 comp_ref_prob[ctx][2], ctx 的计算如下：

```

VarRefIdx = 0
if ( AvailU && AvailL ) {
    if ( AboveIntra && LeftIntra ) {
        ctx = 2
    } else if ( LeftIntra ) {
        if ( AboveSingle )

```

```

ctx = 1 + 2 * (AboveRefFrame[0] == DYNAMIC_REF_1)
else
    ctx = 1 + 2 * (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_1)
} else if ( AboveIntra ) {
    if ( LeftSingle )
        ctx = 1 + 2 * (LeftRefFrame[0] == DYNAMIC_REF_1)
    else
        ctx = 1 + 2 * (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_1)
} else {
    vrfa = AboveSingle ? AboveRefFrame[0] : AboveRefFrame[VarRefIdx]
    vrf1 = LeftSingle ? LeftRefFrame[0] : LeftRefFrame[VarRefIdx]
    if ( vrfa == vrf1 && DYNAMIC_REF_1 == vrfa ) {
        ctx = 0
    } else if ( LeftSingle && AboveSingle ) {
        if ( (vrfa == DYNAMIC_REF_1 && vrf1 == DYNAMIC_REF_1) )
            ctx = 4
        else if ( vrfa == DYNAMIC_REF_1 || vrf1 == DYNAMIC_REF_1 )
            ctx = 3
        else if (((vrfa == DYNAMIC_REF) || (vrfa == STATIC_REF)) || ((vrf1 == DYNAMIC_REF) || (vrf1 == STATIC_REF)))
            ctx = 1 + (vrfa == vrf1)
        else
            ctx = 1
    } else if ( LeftSingle || AboveSingle ) {
        vrfc = LeftSingle ? vrfa : vrf1
        rfs = AboveSingle ? vrfa : vrf1
        if ( vrfc == DYNAMIC_REF_1 && rfs == DYNAMIC_REF_1 )
            ctx = 4
        else if ( vrfc == DYNAMIC_REF_1 && rfs != DYNAMIC_REF_1 )
            ctx = 3
        else if ( vrfc != DYNAMIC_REF_1 && rfs == DYNAMIC_REF_1 )
            ctx = 2 + (!((vrfc == DYNAMIC_REF) || (vrfc == STATIC_REF)))
        else if(vrfc != DYNAMIC_REF_1 && rfs != DYNAMIC_REF_1){
            if (((vrfc == DYNAMIC_REF) || (vrfc == STATIC_REF)) || ((rfs == DYNAMIC_REF) || (rfs == STATIC_REF)))
                ctx = 2
            else
                ctx = 1
        }
    } else {
        if (vrfa == DYNAMIC_REF_1 && vrf1 == DYNAMIC_REF_1)
            ctx = 4
        else if (vrfa == DYNAMIC_REF_1 || vrf1 == DYNAMIC_REF_1)

```

```

        ctx = 3
    else if (vrafa != DYNAMIC_REF_1 && vrfl != DYNAMIC_REF_1)
        ctx = 1
    else
        ctx = 2
    }
}
} else if (AvailU) {
    if (AboveIntra) {
        ctx = 2
    } else {
        if (!AboveSingle)
            ctx = 4 * (AboveRefFrame[VarRefIdx] != DYNAMIC_REF_1)
        else {
            if (AboveRefFrame[0] == DYNAMIC_REF_1)
                ctx = 3
            else if (((AboveRefFrame[0] == DYNAMIC_REF) || (AboveRefFrame[0] == STATIC_REF)))
                ctx = 2
            else
                ctx = 1
        }
    }
} else if (AvailL) {
    if (LeftIntra) {
        ctx = 2
    } else {
        if (!LeftSingle)
            ctx = 4 * (LeftRefFrame[VarRefIdx] != DYNAMIC_REF_1)
        else {
            if (LeftRefFrame[0] == DYNAMIC_REF_1)
                ctx = 3
            else if (((LeftRefFrame[0] == DYNAMIC_REF) || (LeftRefFrame[0] == STATIC_REF)))
                ctx = 2
            else
                ctx = 1
        }
    }
} else {
    ctx = 2
}

```

single reference 模式下 binIdx 等于 0 的情况：

概率等于 single_ref_prob[ctx][0], ctx 的计算如下：

```

if ( AvailU && AvailL ) {
    if ( AboveIntra && LeftIntra ) {
        ctx = 2
    } else if ( LeftIntra ) {
        if ( AboveSingle )
            ctx = 4 * (AboveRefFrame[0] == DYNAMIC_REF)
        else
            ctx = 1 + (AboveRefFrame[0] == DYNAMIC_REF || AboveRefFrame[1] == DYNAMIC_REF)
    } else if ( AboveIntra ) {
        if ( LeftSingle )
            ctx = 4 * (LeftRefFrame[0] == DYNAMIC_REF)
        else
            ctx = 1 + (LeftRefFrame[0] == DYNAMIC_REF || LeftRefFrame[1] == DYNAMIC_REF)
    } else {
        if ( ! AboveSingle && ! LeftSingle ) {
            ctx = 1 + (AboveRefFrame[0] == DYNAMIC_REF || AboveRefFrame[1] == DYNAMIC_REF
                || LeftRefFrame[0] == DYNAMIC_REF || LeftRefFrame[1] == DYNAMIC_REF)
        } else if ( ! AboveSingle || ! LeftSingle ) {
            rfs = AboveSingle ? AboveRefFrame[0] : LeftRefFrame[0]
            crf1 = AboveSingle ? LeftRefFrame[0] : AboveRefFrame[0]
            crf2 = AboveSingle ? LeftRefFrame[1] : AboveRefFrame[1]
            if ( rfs == DYNAMIC_REF )
                ctx = 3 + (crf1 == DYNAMIC_REF || crf2 == DYNAMIC_REF)
            else
                ctx = crf1 == DYNAMIC_REF || crf2 == DYNAMIC_REF
        } else {
            ctx = 2 * (AboveRefFrame[0] == DYNAMIC_REF) + 2 * (LeftRefFrame[0] == DYNAMIC_REF)
        }
    }
} else if ( AvailU ) {
    if ( AboveIntra ) {
        ctx = 2
    } else { // inter
        if ( AboveSingle )
            ctx = 4 * (AboveRefFrame[0] == DYNAMIC_REF)
        else
            ctx = 1 + (AboveRefFrame[0] == DYNAMIC_REF || AboveRefFrame[1] == DYNAMIC_REF)
    }
} else if ( AvailL ) {
    if ( LeftIntra ) {
        ctx = 2
    }
}

```

```

} else {
    if ( LeftSingle )
        ctx = 4 * (LeftRefFrame[0] == DYNAMIC_REF)
    else
        ctx = 1 + (LeftRefFrame[0] == DYNAMIC_REF || LeftRefFrame[1] == DYNAMIC_REF)
}
} else {
    ctx = 2
}

```

single reference 模式下 binIdx 等于 1 的情况：

概率等于 single_ref_prob[ctx][1].ctx 的计算如下：

```

if ( AvailU && AvailL ) {
    if ( AboveIntra && LeftIntra ) {
        ctx = 2
    } else if ( LeftIntra ) {
        if ( AboveSingle ) {
            if ( AboveRefFrame[0] == DYNAMIC_REF )
                ctx = 3
            else
                ctx = 4 * (AboveRefFrame[0] == STATIC_REF)
        } else {
            ctx = 1 + 2 * (AboveRefFrame[0] == STATIC_REF || AboveRefFrame[1] == STATIC_REF)
        }
    } else if ( AboveIntra ) {
        if ( LeftSingle ) {
            if ( LeftRefFrame[0] == DYNAMIC_REF )
                ctx = 3
            else
                ctx = 4 * (LeftRefFrame[0] == STATIC_REF)
        } else {
            ctx = 1 + 2 * (LeftRefFrame[0] == STATIC_REF || LeftRefFrame[1] == STATIC_REF)
        }
    } else {
        if ( ! AboveSingle && ! LeftSingle ) {
            if (AboveRefFrame[0] == LeftRefFrame[0] && AboveRefFrame[1] == LeftRefFrame[1])
                ctx = 3 * (AboveRefFrame[0] == STATIC_REF || AboveRefFrame[1] == STATIC_REF)
            else
                ctx = 2
        } else if ( ! AboveSingle || ! LeftSingle ) {
            rfs = AboveSingle ? AboveRefFrame[0] : LeftRefFrame[0]
            crf1 = AboveSingle ? LeftRefFrame[0] : AboveRefFrame[0]
            crf2 = AboveSingle ? LeftRefFrame[1] : AboveRefFrame[1]
            if ( rfs == STATIC_REF )

```

```

ctx = 3 + (crf1 == STATIC_REF || crf2 == STATIC_REF)
else if ( rfs == OPTIONAL_REF )
    ctx = crf1 == STATIC_REF || crf2 == STATIC_REF
else
    ctx = 1 + 2 * (crf1 == STATIC_REF || crf2 == STATIC_REF)
}
} else {
    if ( AboveRefFrame[0] == DYNAMIC_REF && LeftRefFrame[0] == DYNAMIC_REF ) {
        ctx = 3
    } else if ( AboveRefFrame[0] == DYNAMIC_REF ) {
        ctx = 4 * (LeftRefFrame[0] == STATIC_REF)
    } else if ( LeftRefFrame[0] == DYNAMIC_REF ) {
        ctx = 4 * (AboveRefFrame[0] == STATIC_REF)
    } else {
        ctx = 2 * (AboveRefFrame[0] == STATIC_REF) + 2 * (LeftRefFrame[0] == STATIC_REF)
    }
}
} else if ( AvailU ) {
    if ( AboveIntra || (AboveRefFrame[0] == DYNAMIC_REF && AboveSingle) )
        ctx = 2
    else if ( AboveSingle )
        ctx = 4 * (AboveRefFrame[0] == STATIC_REF)
    else
        ctx = 3 * (AboveRefFrame[0] == STATIC_REF || AboveRefFrame[1] == STATIC_REF)
} else if ( AvailL ) {
    if ( LeftIntra || (LeftRefFrame[0] == DYNAMIC_REF && LeftSingle) )
        ctx = 2
    else if ( LeftSingle )
        ctx = 4 * (LeftRefFrame[0] == STATIC_REF)
    else
        ctx = 3 * (LeftRefFrame[0] == STATIC_REF || LeftRefFrame[1] == STATIC_REF)
} else {
    ctx = 2
}

```

single reference 模式下 binIdx 分别等于 2 和 3 的情况：

概率分别等于 single_ref_prob[ctx][2] 和 single_ref_prob[ctx][3]，ctx 的计算如下：

```

if(CurSelectRef == OPTIONAL_REF) {
    is_opt_ref = 1
}
if ( AvailU && AvailL ) {
    if ( AboveIntra && LeftIntra ) {
        ctx = 2
    } else if ( LeftIntra ) {

```

```

if ( AboveSingle ) {
    if ( AboveRefFrame[0] == CurSelectRef) {
        ctx = 4
    } else {
        if (is_opt_ref) {
            if (AboveRefFrame[0] == DYANMIC_REF_1)
                ctx = 0
            else
                ctx = 3
        } else {
            if (AboveRefFrame[0] == DYANMIC_REF_2)
                ctx = 0
            else
                ctx = 2
        }
    }
} else {
    ctx = 1 + 2 * (AboveRefFrame[0] == CurSelectRef || AboveRefFrame[1] == CurSelectRef)
}
} else if (AboveIntra) {
    if ( LeftSingle ) {
        if (LeftRefFrame[0] == CurSelectRef) {
            ctx = 4
        } else {
            if (is_opt_ref) {
                if (LeftRefFrame[0] == DYANMIC_REF_1)
                    ctx = 0
                else
                    ctx = 3
            } else {
                if (LeftRefFrame[0] == DYANMIC_REF_2)
                    ctx = 0
                else
                    ctx = 2
            }
        }
    }
} else {
    ctx = 1 + 2 * (LeftRefFrame[0] == CurSelectRef || LeftRefFrame[1] == CurSelectRef)
}
} else {
    if ( ! AboveSingle && ! LeftSingle ) {
        if(AboveRefFrame[0] == LeftRefFrame[0]&& AboveRefFrame[1] == LeftRefFrame[1]){
            ctx = 3 * (AboveRefFrame[0] == CurSelectRef || AboveRefFrame[1] == CurSelectRef)
        }
    }
}

```

```

tRef || LeftRefFrame[0] == CurSelectRef || LeftRefFrame[1] == CurSelectRef)
} else {
    ctx = 2
}
} else if( ! AboveSingle || ! LeftSingle ) {
    rfs = AboveSingle ? AboveRefFrame[0] : LeftRefFrame[0]
    crf1 = AboveSingle ? LeftRefFrame[0] : AboveRefFrame[0]
    crf2 = AboveSingle ? LeftRefFrame[1] : AboveRefFrame[1]
    if ( rfs == CurSelectRef )
        ctx = 3 + (crf1 == CurSelectRef || crf2 == CurSelectRef)
    else {
        if (is_opt_ref)
            ctx = 2
        else {
            if (rfs == DYNAMIC_REF_2)
                ctx = 2 * (crf1 == CurSelectRef || crf2 == CurSelectRef)
            else
                ctx = 2
        }
    }
} else {
    if (is_opt_ref)
        ctx = 2 * (AboveRefFrame[0] == OPTIONAL_REF) + 2 * (LeftRefFrame[0] == OPTIONAL_REF)
    else {
        if (AboveRefFrame[0] == DYNAMIC_REF_1 && LeftRefFrame[0] == DYNAMIC_REF_1)
            ctx = 4
        else if(AboveRefFrame[0] == DYNAMIC_REF_1&&LeftRefFrame[0]! = DYNAMIC_REF_1)
            ctx = 3
        else if(AboveRefFrame[0]! = DYNAMIC_REF_1&&LeftRefFrame[0] == DYNAMIC_REF_1)
            ctx = 3
        else
            ctx = 2
    }
}
} else if ( AvailU ) {
    if ( AboveIntra || (AboveRefFrame[0] != CurSelectRef && AboveSingle)) {
        ctx = 2
    } else if (AboveSingle) {
        ctx = 4 * (AboveRefFrame[0] == CurSelectRef)
    } else {
        if (is_opt_ref) {

```



```

        if( AboveRefFrame[0] == DYNAMIC_REF_1 || AboveRefFrame[1] == DYNAMIC_REF_1 )
            ctx = 1
        else
            ctx = 2
    } else {
        if( AboveRefFrame[0] == DYNAMIC_REF_1 || AboveRefFrame[1] == DYNAMIC_REF_1 )
            ctx = 3
        else if(AboveRefFrame[0] == DYNAMIC_REF_2|| AboveRefFrame[1] == DYNAMIC_REF_2 )
            ctx = 1
        else
            ctx = 2
    }
}
} else if ( AvailL ) {
    if ( LeftIntra || (LeftRefFrame[0] != CurSelectRef && LeftSingle)) {
        ctx = 2
    } else if (LeftSingle) {
        ctx = 4 * (LeftRefFrame[0] == CurSelectRef)
    } else {
        if (is_opt_ref) {
            if(LeftRefFrame[0] == DYNAMIC_REF_1 || LeftRefFrame[1] == DYNAMIC_REF_1 )
                ctx = 1
            else
                ctx = 2
        } else {
            if(LeftRefFrame[0] == DYNAMIC_REF_1 || LeftRefFrame[1] == DYNAMIC_REF_1 )
                ctx = 3
            else if(LeftRefFrame[0] == DYNAMIC_REF_2 || LeftRefFrame[1] == DYNAMIC_REF_2 )
                ctx = 1
            else
                ctx = 2
        }
    }
} else {
    ctx = 2
}

```

mv_mode

解析 binIdx 等于 0 的二进制位时, 概率等于 newmv_mode_prob[ctx], ctx 的计算如下:

```

mode_ctx = mode_context[CurRefFrame[0]]
if(ref_frame[1]>NONE)
    mode_ctx &= (mode_context[CurRefFrame[1]] | 0x00FF)
if(block_size<BLOCK_8X8)
    mode_ctx &= 0x00FF;

```

```
ctx = mode_ctx & ((1 << ZEROMV_OFFSET)-1)
```

解析 binIdx 等于 1 的二进制位时, 概率等于 zeromv_mode_prob[ctx], ctx 的计算如下:

```
mode_ctx = mode_context[CurRefFrame[0]]
```

```
if(ref_frame[1]>NONE)
```

```
    mode_ctx &= (mode_context[CurRefFrame[1]] | 0x00FF)
```

```
if(block_size<BLOCK_8X8)
```

```
    mode_ctx &= 0x00FF;
```

```
ctx = (mode_ctx >> ZEROMV_OFFSET) & ((1 << (REFMV_OFFSET-ZEROMV_OFFSET))-1)
```

解析 binIdx 等于 2 的二进制位时, 概率等于 refmv_mode_prob[ctx], ctx 的计算如下:

```
mode_ctx = mode_context[CurRefFrame[0]]
```

```
if(ref_frame[1]>NONE)
```

```
    mode_ctx &= (mode_context[CurRefFrame[1]] | 0x00FF)
```

```
if(block_size<BLOCK_8 × 8)
```

```
    mode_ctx &= 0x00FF;
```

```
ctx = (mode_ctx >> REFMV_OFFSET) & ((1 << (8-REFMV_OFFSET))-1)
```

mv_joint:

概率等于 mv_joint_probs[treeIdx], 其中 treeIdx 的计算过程见 5.4.2.4.3。

mvd_sign_0:

概率等于 mv_sign_probs[0]。 

mvd_sign_1:

概率等于 mv_sign_probs[1]。

mvd_value_0 和 mvd_value_1: 通过以下信息计算得到, 分别对应 comp 为 0 和 1 的情况。

comp 用来索引以下概率表, 为 0 表示垂直坐标, 为 1 表示水平坐标。

使用的概率表包括 mv_bits_probs、mv_class_probs、mv_class0_fr_probs、mv_class0_hp_prob、mv_fr_probs 和 mv_hp_prob。

mv_class:

概率等于 mv_class_probs[comp][treeIdx]。

d:

在 mv_class 等于 0 时, 概率通过 mv_class0_bit_probs[comp] 得到;

在 mv_class 不等于 0 时, 概率通过 mv_bits_probs[comp][bit_offset] 得到。

fr:

在 mv_class 等于 0 时, 概率通过 mv_class0_fr[comp][mv_class0_bit][treeIdx] 得到;

在 mv_class 不等于 0 时, 概率通过 mv_fr_probs[comp][treeIdx] 得到。

treeIdx 的计算过程见 5.4.2.4.3。

hp:

在 mv_class 等于 0 时, 概率通过 mv_class0_hp_prob[comp] 得到;

在 mv_class 不等于 0 时, 概率通过 mv_hp_prob[comp] 得到。

interp_filter_mode:

概率等于 switchable_interp_prob[ctx][treeIdx], treeIdx 的计算过程见 5.4.2.4.3, ctx 的计算如下:

```
leftInterp = (AvailL && LeftInter) ? LeftInterpFilter : 3
```

```
aboveInterp = (AvailU && AboveInter) ? AboveInterpFilter : 3
```

```
if (leftInterp == aboveInterp)
```

```
    ctx = leftInterp
else if ( leftInterp == 3 && aboveInterp != 3 )
    ctx = aboveInterp
else if ( leftInterp != 3 && aboveInterp == 3 )
    ctx = leftInterp
else
    ctx = 3
dqp_abs
```

概率等于 `dqp_abs_prob[ctx][treeIdx]`, `treeIdx` 的计算过程见 5.4.2.4.3, `ctx` 的计算如下:

```
    ctx = last_dqindex != 0  
last_dqindex 为上一个解码出来的 dqp_abs。
```

coeff value;

`coeff_value` 通过 `token` 和 `extra` 计算得到。

亮度系数的 token 的概率根据 `coef_probs[tx_size][0][inter_block][band][ctx][binIdx]` 或者 `svac2_pareto8_full[ctx][treeIdx]` 确定。

band 根据 tx_size 查表 127 及表 128 获取：

```

if(tx_size == TX_4×4)
    band = coefband_trans_4×4[c]
else
    band = coefband_trans_8×8plus[c]

```

表 127 coefband_trans_4 × 4

```
coefband_trans_4×4[16] = {  
    0,1,1,2,2,2,3,3,3,3,4,4,4,4,5,5,5,  
}
```

表 128 coefband_trans_8×8plus

表 128 (续)

表 128 (续)

5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
}

ctx 通过如下计算得到：

```

if(c = 0) {
    numpts = 1 << tx_size
    above = 0
    left = 0
    for(i = 0;i<numpts;i++) {
        above |= AboveNonzeroContext[i]
        left |= LeftNonzeroContext[i]
    }
    ctx = above + left
} else {
    ctx = (1 + TokenCache[nb[MAX_NEIGHBORS * c]] + TokenCache[nb[MAX_NEIGHBORS * c + 1]]) >>1
}

```

nb 通过如下计算查表 129~138 得到：

```

if(inter_block || ! is_plane_y) {
    if(tx_size == TX_4×4) {
        nb = default_scan_4×4_neighbors
    }
}

```

```

} else if(tx_size == TX_8×8) {
    nb = default_scan_8×8_neighbors
} else if(tx_size == TX_16×16) {
    nb = default_scan_16×16_neighbors
} else {
    nb = default_scan_32×32_neighbors
}
} else {
    if(tx_size == TX_4×4) {
        if(tx_type == ADST_DCT) {
            nb = row_scan_4×4_neighbors
        } else if(tx_type == DCT_ADST) {
            nb = col_scan_4×4_neighbors
        } else {
            nb = default_scan_4×4_neighbors
        }
    } else if(tx_size == TX_8×8) {
        if(tx_type == ADST_DCT) {
            nb = row_scan_8×8_neighbors
        } else if(tx_type == DCT_ADST) {
            nb = col_scan_8×8_neighbors
        } else{
            nb = default_scan_8×8_neighbors
        }
    } else if(tx_size == TX_16×16) {
        if(tx_type == ADST_DCT) {
            nb = row_scan_16×16_neighbors
        } else if(tx_type == DCT_ADST) {
            nb = col_scan_16×16_neighbors
        } else {
            nb = default_scan_16×16_neighbors
        }
    }
} else {
    nb = default_scan_32×32_neighbors
}
}

```

表 129 default_scan_4×4_neighbors

default_scan_4×4_neighbors[17 * MAX_NEIGHBORS] = {
0,0,0,0,0,0,1,4,4,4,1,1,8,8,5,8,2,2,2,5,9,12,6,9,
3,6,10,13,7,10,11,14,0,0,
};

表 130 col_scan_4×4_neighbors

col_scan_4×4_neighbors[17 * MAX_NEIGHBORS] = {
0,0,0,0,4,4,0,0,8,8,1,1,5,5,1,1,9,9,2,2,6,6,2,2,3,
3,10,10,7,7,11,11,0,0,
};

表 131 row_scan_4×4_neighbors

row_scan_4×4_neighbors[17 * MAX_NEIGHBORS] = {
0,0,0,0,0,0,1,1,4,4,2,2,5,5,4,4,8,8,6,6,8,8,9,9,12,
12,10,10,13,13,14,14,0,0,
};

表 132 col_scan_8×8_neighbors

col_scan_8×8_neighbors[65 * MAX_NEIGHBORS] = {
0,0,0,0,8,8,0,0,16,16,1,1,24,24,9,9,1,1,32,32,17,17,2,
2,25,25,10,10,40,40,2,2,18,18,33,33,3,3,48,48,11,11,26,
26,3,3,41,41,19,19,34,34,4,4,27,27,12,12,49,49,42,42,20,
20,4,4,35,35,5,5,28,28,50,50,43,43,13,13,36,36,5,5,21,21,
51,51,29,29,6,6,44,44,14,14,6,6,37,37,52,52,22,22,7,7,30,
30,45,45,15,15,38,38,23,23,53,53,31,31,46,46,39,39,54,54,
47,47,55,55,0,0,
};

表 133 row_scan_8×8_neighbors

row_scan_8×8_neighbors[65 * MAX_NEIGHBORS] = {
0,0,0,0,1,1,0,0,8,8,2,2,8,8,9,9,3,3,16,16,10,10,16,16,
4,4,17,17,24,24,11,11,18,18,25,25,24,24,5,5,12,12,19,19,
32,32,26,26,6,6,33,33,32,32,20,20,27,27,40,40,13,13,34,34,
40,40,41,41,28,28,35,35,48,48,21,21,42,42,14,14,48,48,36,
36,49,49,43,43,29,29,56,56,22,22,50,50,57,57,44,44,37,37,
51,51,30,30,58,58,52,52,45,45,59,59,38,38,60,60,46,46,53,
53,54,54,61,61,62,62,0,0,
};

表 134 default_scan_8×8_neighbors

default_scan_8×8_neighbors[65 * MAX_NEIGHBORS] = {
0,0,0,0,0,0,8,8,1,8,1,1,9,16,16,16,2,9,2,2,10,17,17,
24,24,24,3,10,3,3,18,25,25,32,11,18,32,32,4,11,26,33,19,
26,4,4,33,40,12,19,40,40,5,12,27,34,34,41,20,27,13,20,5,
5,41,48,48,48,28,35,35,42,21,28,6,6,6,13,42,49,49,56,36,
43,14,21,29,36,7,14,43,50,50,57,22,29,37,44,15,22,44,51,
51,58,30,37,23,30,52,59,45,52,38,45,31,38,53,60,46,53,39,
46,54,61,47,54,55,62,0,0,
}

表 135 col_scan_16×16_neighbors

col_scan_16×16_neighbors[257 * MAX_NEIGHBORS] =
{
0,0,0,0,16,16,32,32,0,0,48,48,1,1,64,64,
17,17,80,80,33,33,1,1,49,49,96,96,2,2,65,65,
18,18,112,112,34,34,81,81,2,2,50,50,128,128,3,3,
97,97,19,19,66,66,144,144,82,82,35,35,113,113,3,3,
51,51,160,160,4,4,98,98,129,129,67,67,20,20,83,83,
114,114,36,36,176,176,4,4,145,145,52,52,99,99,5,5,
130,130,68,68,192,192,161,161,21,21,115,115,84,84,37,37,
146,146,208,208,53,53,5,5,100,100,177,177,131,131,69,69,
6,6,224,224,116,116,22,22,162,162,85,85,147,147,38,38,
193,193,101,101,54,54,6,6,132,132,178,178,70,70,163,163,
209,209,7,7,117,117,23,23,148,148,7,7,86,86,194,194,
225,225,39,39,179,179,102,102,133,133,55,55,164,164,8,8,
71,71,210,210,118,118,149,149,195,195,24,24,87,87,40,40,
56,56,134,134,180,180,226,226,103,103,8,8,165,165,211,211,
72,72,150,150,9,9,119,119,25,25,88,88,196,196,41,41,
135,135,181,181,104,104,57,57,227,227,166,166,120,120,151,151,
197,197,73,73,9,9,212,212,89,89,136,136,182,182,10,10,
26,26,105,105,167,167,228,228,152,152,42,42,121,121,213,213,
58,58,198,198,74,74,137,137,183,183,168,168,10,10,90,90,
229,229,11,11,106,106,214,214,153,153,27,27,199,199,43,43,
184,184,122,122,169,169,230,230,59,59,11,11,75,75,138,138,

表 135 (续)

200,200,215,215,91,91,12,12,28,28,185,185,107,107,154,154,
44,44,231,231,216,60,60,123,123,12,12,76,76,201,201,
170,170,232,232,139,139,92,92,13,13,108,108,29,29,186,186,
217,217,155,155,45,45,13,13,61,61,124,124,14,14,233,233,
77,77,14,14,171,171,140,140,202,202,30,30,93,93,109,109,
46,46,156,156,62,62,187,187,15,15,125,125,218,218,78,78,
31,31,172,172,47,47,141,141,94,94,234,234,203,203,63,63,
110,110,188,188,157,157,126,126,79,79,173,173,95,95,219,219,
142,142,204,204,235,235,111,111,158,158,127,127,189,189,220,
220,143,143,174,174,205,205,236,236,159,159,190,190,221,221,
175,175,237,237,206,206,222,222,191,191,238,238,207,207,223,
223,239,239,0,0,
}

表 136 row_scan_16×16_neighbors

row_scan_16×16_neighbors[257 * MAX_NEIGHBORS] =
{
0,0,0,0,1,1,0,0,2,2,16,16,3,3,17,17,
16,16,4,4,32,32,18,18,5,5,33,33,32,32,19,19,
48,48,6,6,34,34,20,20,49,49,48,48,7,7,35,35,
64,64,21,21,50,50,36,36,64,64,8,8,65,65,51,51,
22,22,37,37,80,80,66,66,9,9,52,52,23,23,81,81,
67,67,80,80,38,38,10,10,53,53,82,82,96,96,68,68,
24,24,97,97,83,83,39,39,96,96,54,54,11,11,69,69,
98,98,112,112,84,84,25,25,40,40,55,55,113,113,99,99,
12,12,70,70,112,112,85,85,26,26,114,114,100,100,128,128,
41,41,56,56,71,71,115,115,13,13,86,86,129,129,101,101,
128,128,72,72,130,130,116,116,27,27,57,57,14,14,87,87,
42,42,144,144,102,102,131,131,145,145,117,117,73,73,144,144,
88,88,132,132,103,103,28,28,58,58,146,146,118,118,43,43,
160,160,147,147,89,89,104,104,133,133,161,161,119,119,160,160,
74,74,134,134,148,148,29,29,59,59,162,162,176,176,44,44,
120,120,90,90,105,105,163,163,177,177,149,149,176,176,135,135,
164,164,178,178,30,30,150,150,192,192,75,75,121,121,60,60,

表 136 (续)

136,136,193,193,106,106,151,151,179,179,192,192,45,45,165,165,
166,166,194,194,91,91,180,180,137,137,208,208,122,122,152,152,
208,208,195,195,76,76,167,167,209,209,181,181,224,224,107,107,
196,196,61,61,153,153,224,224,182,182,168,168,210,210,46,46,
138,138,92,92,183,183,225,225,211,211,240,240,197,197,169,169,
123,123,154,154,198,198,77,77,212,212,184,184,108,108,226,226,
199,199,62,62,227,227,241,241,139,139,213,213,170,170,185,185,
155,155,228,228,242,242,124,124,93,93,200,200,243,243,214,214,
215,215,229,229,140,140,186,186,201,201,78,78,171,171,109,109,
156,156,244,244,216,216,230,230,94,94,245,245,231,231,125,125,
202,202,246,246,232,232,172,172,217,217,141,141,110,110,157,
157,187,187,247,247,126,126,233,233,218,218,248,248,188,188,
203,203,142,142,173,173,158,158,249,249,234,234,204,204,219,
219,174,174,189,189,250,250,220,220,190,190,205,205,235,235,
206,206,236,236,251,251,221,221,252,252,222,222,237,237,238,
238,253,253,254,254,0,0,
}

表 137 default_scan_16×16_neighbors

default_scan_16×16_neighbors[257 * MAX_NEIGHBORS] =
{
0,0,0,0,0,0,16,16,1,16,1,1,32,32,17,32,
2,17,2,2,48,48,18,33,33,48,3,18,49,64,64,64,
34,49,3,3,19,34,50,65,4,19,65,80,80,80,35,50,
4,4,20,35,66,81,81,96,51,66,96,96,5,20,36,51,
82,97,21,36,67,82,97,112,5,5,52,67,112,112,37,52,
6,21,83,98,98,113,68,83,6,6,113,128,22,37,53,68,
84,99,99,114,128,128,114,129,69,84,38,53,7,22,7,7,
129,144,23,38,54,69,100,115,85,100,115,130,144,144,130,145,
39,54,70,85,8,23,55,70,116,131,101,116,145,160,24,39,
8,8,86,101,131,146,160,160,146,161,71,86,40,55,9,24,
117,132,102,117,161,176,132,147,56,71,87,102,25,40,147,162,
9,9,176,176,162,177,72,87,41,56,118,133,133,148,103,118,
10,25,148,163,57,72,88,103,177,192,26,41,163,178,192,192,



表 137 (续)

10,10,119,134,73,88,149,164,104,119,134,149,42,57,178,193,
164,179,11,26,58,73,193,208,89,104,135,150,120,135,27,42,
74,89,208,208,150,165,179,194,165,180,105,120,194,209,43,58,
11,11,136,151,90,105,151,166,180,195,59,74,121,136,209,224,
195,210,224,224,166,181,106,121,75,90,12,27,181,196,12,12,
210,225,152,167,167,182,137,152,28,43,196,211,122,137,91,106,
225,240,44,59,13,28,107,122,182,197,168,183,211,226,153,168,
226,241,60,75,197,212,138,153,29,44,76,91,13,13,183,198,
123,138,45,60,212,227,198,213,154,169,169,184,227,242,92,107,
61,76,139,154,14,29,14,14,184,199,213,228,108,123,199,214,
228,243,77,92,30,45,170,185,155,170,185,200,93,108,124,139,
214,229,46,61,200,215,229,244,15,30,109,124,62,77,140,155,
215,230,31,46,171,186,186,201,201,216,78,93,230,245,125,140,
47,62,216,231,156,171,94,109,231,246,141,156,63,78,202,217,
187,202,110,125,217,232,172,187,232,247,79,94,157,172,126,141,
203,218,95,110,233,248,218,233,142,157,111,126,173,188,188,203,
234,249,219,234,127,142,158,173,204,219,189,204,143,158,235,
250,174,189,205,220,159,174,220,235,221,236,175,190,190,205,
236,251,206,221,237,252,191,206,222,237,207,222,238,253,223,
238,239,254,0,0,
};

表 138 default_scan_32×32_neighbors

default_scan_32×32_neighbors[1025 * MAX_NEIGHBORS] =
{
0,0,0,0,0,0,32,32,1,32,1,1,64,64,33,64,
2,33,96,96,2,2,65,96,34,65,128,128,97,128,3,34,
66,97,3,3,35,66,98,129,129,160,160,160,4,35,67,98,
192,192,4,4,130,161,161,192,36,67,99,130,5,36,68,99,
193,224,162,193,224,224,131,162,37,68,100,131,5,5,194,225,
225,256,256,256,163,194,69,100,132,163,6,37,226,257,6,6,
195,226,257,288,101,132,288,288,38,69,164,195,133,164,258,289,
227,258,196,227,7,38,289,320,70,101,320,320,7,7,165,196,
39,70,102,133,290,321,259,290,228,259,321,352,352,352,197,228,

表 138 (续)

134,165,71,102,8,39,322,353,291,322,260,291,103,134,353,384,
166,197,229,260,40,71,8,8,384,384,135,166,354,385,323,354,
198,229,292,323,72,103,261,292,9,40,385,416,167,198,104,135,
230,261,355,386,416,416,293,324,324,355,9,9,41,72,386,417,
199,230,136,167,417,448,262,293,356,387,73,104,387,418,231,262,
10,41,168,199,325,356,418,449,105,136,448,448,42,73,294,325,
200,231,10,10,357,388,137,168,263,294,388,419,74,105,419,450,
449,480,326,357,232,263,295,326,169,200,11,42,106,137,480,480,
450,481,358,389,264,295,201,232,138,169,389,420,43,74,420,451,
327,358,11,11,481,512,233,264,451,482,296,327,75,106,170,201,
482,513,512,512,390,421,359,390,421,452,107,138,12,43,202,233,
452,483,265,296,328,359,139,170,44,75,483,514,513,544,234,265,
297,328,422,453,12,12,391,422,171,202,76,107,514,545,453,484,
544,544,266,297,203,234,108,139,329,360,298,329,140,171,515,
546,13,44,423,454,235,266,545,576,454,485,45,76,172,203,330,
361,576,576,13,13,267,298,546,577,77,108,204,235,455,486,577,
608,299,330,109,140,547,578,14,45,14,14,141,172,578,609,331,
362,46,77,173,204,15,15,78,109,205,236,579,610,110,141,15,46,
142,173,47,78,174,205,16,16,79,110,206,237,16,47,111,142,
48,79,143,174,80,111,175,206,17,48,17,17,207,238,49,80,
81,112,18,18,18,49,50,81,82,113,19,50,51,82,83,114,608,608,
484,515,360,391,236,267,112,143,19,19,640,640,609,640,516,547,
485,516,392,423,361,392,268,299,237,268,144,175,113,144,20,51,
20,20,672,672,641,672,610,641,548,579,517,548,486,517,424,455,
393,424,362,393,300,331,269,300,238,269,176,207,145,176,114,
145,52,83,21,52,21,21,704,704,673,704,642,673,611,642,580,
611,549,580,518,549,487,518,456,487,425,456,394,425,363,394,
332,363,301,332,270,301,239,270,208,239,177,208,146,177,115,
146,84,115,53,84,22,53,22,22,705,736,674,705,643,674,581,612,
550,581,519,550,457,488,426,457,395,426,333,364,302,333,271,
302,209,240,178,209,147,178,85,116,54,85,23,54,706,737,675,
706,582,613,551,582,458,489,427,458,334,365,303,334,210,241,
179,210,86,117,55,86,707,738,583,614,459,490,335,366,211,242,
87,118,736,736,612,643,488,519,364,395,240,271,116,147,23,23,
768,768,737,768,644,675,613,644,520,551,489,520,396,427,365,

表 138 (续)

396,272,303,241,272,148,179,117,148,24,55,24,24,800,800,769,
800,738,769,676,707,645,676,614,645,552,583,521,552,490,521,
428,459,397,428,366,397,304,335,273,304,242,273,180,211,149,
180,118,149,56,87,25,56,25,25,832,832,801,832,770,801,739,
770,708,739,677,708,646,677,615,646,584,615,553,584,522,553,
491,522,460,491,429,460,398,429,367,398,336,367,305,336,274,
305,243,274,212,243,181,212,150,181,119,150,88,119,57,88,26,
57,26,26,833,864,802,833,771,802,709,740,678,709,647,678,585,
616,554,585,523,554,461,492,430,461,399,430,337,368,306,337,
275,306,213,244,182,213,151,182,89,120,58,89,27,58,834,865,
803,834,710,741,679,710,586,617,555,586,462,493,431,462,338,
369,307,338,214,245,183,214,90,121,59,90,835,866,711,742,587,
618,463,494,339,370,215,246,91,122,864,864,740,771,616,647,
492,523,368,399,244,275,120,151,27,27,896,896,865,896,772,803,
741,772,648,679,617,648,524,555,493,524,400,431,369,400,276,
307,245,276,152,183,121,152,28,59,28,28,928,928,897,928,866,
897,804,835,773,804,742,773,680,711,649,680,618,649,556,587,
525,556,494,525,432,463,401,432,370,401,308,339,277,308,246,
277,184,215,153,184,122,153,60,91,29,60,29,29,960,960,929,
960,898,929,867,898,836,867,805,836,774,805,743,774,712,743,
681,712,650,681,619,650,588,619,557,588,526,557,495,526,464,
495,433,464,402,433,371,402,340,371,309,340,278,309,247,278,
216,247,185,216,154,185,123,154,92,123,61,92,30,61,30,30,
961,992,930,961,899,930,837,868,806,837,775,806,713,744,682,
713,651,682,589,620,558,589,527,558,465,496,434,465,403,434,
341,372,310,341,279,310,217,248,186,217,155,186,93,124,62,93,
31,62,962,993,931,962,838,869,807,838,714,745,683,714,590,621,
559,590,466,497,435,466,342,373,311,342,218,249,187,218,94,
125,63,94,963,994,839,870,715,746,591,622,467,498,343,374,219,
250,95,126,868,899,744,775,620,651,496,527,372,403,248,279,
124,155,900,931,869,900,776,807,745,776,652,683,621,652,528,
559,497,528,404,435,373,404,280,311,249,280,156,187,125,156,
932,963,901,932,870,901,808,839,777,808,746,777,684,715,653,
684,622,653,560,591,529,560,498,529,436,467,405,436,374,405,
312,343,281,312,250,281,188,219,157,188,126,157,964,995,933,

表 138 (续)

964,902,933,871,902,840,871,809,840,778,809,747,778,716,747,
685,716,654,685,623,654,592,623,561,592,530,561,499,530,468,
499,437,468,406,437,375,406,344,375,313,344,282,313,251,282,
220,251,189,220,158,189,127,158,965,996,934,965,903,934,841,
872,810,841,779,810,717,748,686,717,655,686,593,624,562,593,
531,562,469,500,438,469,407,438,345,376,314,345,283,314,221,
252,190,221,159,190,966,997,935,966,842,873,811,842,718,749,
687,718,594,625,563,594,470,501,439,470,346,377,315,346,222,
253,191,222,967,998,843,874,719,750,595,626,471,502,347,378,
223,254,872,903,748,779,624,655,500,531,376,407,252,283,904,
935,873,904,780,811,749,780,656,687,625,656,532,563,501,532,
408,439,377,408,284,315,253,284,936,967,905,936,874,905,812,
843,781,812,750,781,688,719,657,688,626,657,564,595,533,564,
502,533,440,471,409,440,378,409,316,347,285,316,254,285,968,
999,937,968,906,937,875,906,844,875,813,844,782,813,751,782,
720,751,689,720,658,689,627,658,596,627,565,596,534,565,503,
534,472,503,441,472,410,441,379,410,348,379,317,348,286,317,
255,286,969,1000,938,969,907,938,845,876,814,845,783,814,721,
752,690,721,659,690,597,628,566,597,535,566,473,504,442,473,
411,442,349,380,318,349,287,318,970,1001,939,970,846,877,815,
846,722,753,691,722,598,629,567,598,474,505,443,474,350,381,
319,350,971,1002,847,878,723,754,599,630,475,506,351,382,876,
907,752,783,628,659,504,535,380,411,908,939,877,908,784,815,
753,784,660,691,629,660,536,567,505,536,412,443,381,412,940,
971,909,940,878,909,816,847,785,816,754,785,692,723,661,692,
630,661,568,599,537,568,506,537,444,475,413,444,382,413,972,
1003,941,972,910,941,879,910,848,879,817,848,786,817,755,786,
724,755,693,724,662,693,631,662,600,631,569,600,538,569,507,
538,476,507,445,476,414,445,383,414,973,1004,942,973,911,942,
849,880,818,849,787,818,725,756,694,725,663,694,601,632,570,
601,539,570,477,508,446,477,415,446,974,1005,943,974,850,881,
819,850,726,757,695,726,602,633,571,602,478,509,447,478,975,
1006,851,882,727,758,603,634,479,510,880,911,756,787,632,663,
508,539,912,943,881,912,788,819,757,788,664,695,633,664,540,
571,509,540,944,975,913,944,882,913,820,851,789,820,758,789,

表 138 (续)

696,727,665,696,634,665,572,603,541,572,510,541,976,1007,945,
976,914,945,883,914,852,883,821,852,790,821,759,790,728,759,
697,728,666,697,635,666,604,635,573,604,542,573,511,542,977,
1008,946,977,915,946,853,884,822,853,791,822,729,760,698,729,
667,698,605,636,574,605,543,574,978,1009,947,978,854,885,823,
854,730,761,699,730,606,637,575,606,979,1010,855,886,731,762,
607,638,884,915,760,791,636,667,916,947,885,916,792,823,761,
792,668,699,637,668,948,979,917,948,886,917,824,855,793,824,
762,793,700,731,669,700,638,669,980,1011,949,980,918,949,887,
918,856,887,825,856,794,825,763,794,732,763,701,732,670,701,
639,670,981,1012,950,981,919,950,857,888,826,857,795,826,733,
764,702,733,671,702,982,1013,951,982,858,889,827,858,734,765,
703,734,983,1014,859,890,735,766,888,919,764,795,920,951,889,
920,796,827,765,796,952,983,921,952,890,921,828,859,797,828,
766,797,984,1015,953,984,922,953,891,922,860,891,829,860,798,
829,767,798,985,1016,954,985,923,954,861,892,830,861,799,830,
986,1017,955,986,862,893,831,862,987,1018,863,894,892,923,924,
955,893,924,956,987,925,956,894,925,988,1019,957,988,926,957,
895,926,989,1020,958,989,927,958,990,1021,959,990,991,1022,0,0,
}

Token_Cache 为 32×32 的表, 其值的初始化由 token 决定, $\text{token}[\text{scan}[c]]$ 与 token 关系见表 139:

表 139 $\text{token}[\text{scan}[c]]$ 与 token 的关系

token	$\text{token}[\text{scan}[c]]$
0	0
1	1
2	2
3	3
4	3
5	4
6	4
7	5
8	5
9	5
10	5
11	5

svac2_pareto8_full[ctx][treeIdx]中 ctx 的计算如下：

$\text{ctx} = \text{prob}[2] - 1$

$\text{prob}[2]$ 是 binIdx 等于 2 的 coef_probs 的概率表。

treeIdx 的计算过程见 5.4.2.4.3。

extra 会使用到 svac2_cat1_prob 、 svac2_cat2_prob 、 svac2_cat3_prob 、 svac2_cat4_prob 、 svac2_cat5_prob 和 svac2_cat6_prob 概率表，均通过 binIdx 索引得到对应的概率值，具体对应概率表见 5.4.2.2.1。

5.4.2.4.5 二进制位解析

此过程的输入为 range 、 count 、 value 、 buffer 、 buffer_end 以及概率值 prob ，输出为 bit 即解码出的二进制位。其中， range 的位宽为 8 比特， value 位宽为 32 比特。

二进制位的解析过程如下：

第一步，如果 count 的值小于 0，执行算术解码器的重整化过程，与 5.4.2.2 初始化中的重整化过程相同；

第二步，执行 decode 过程得到 bit ，该过程用伪代码描述如下：

```
bit = 0
split = (range × prob + (256 - prob)) >> 8;
value_t = value
count_t = count
range_t = split
bigsplit = split << 24
if (value_t >= bigsplit) {
    range_t = range - split
    value_t = value_t - bigsplit
    bit = 1
}
shift = norm[range_t]
range_t <<= shift
value_t <<= shift
count_t -= shift
value = value_t
count = count_t
range = range_t
```



第三步，如果 bit 的值为 0，则二进制位为‘0’；如果 bit 的值为 1，则二进制位为‘1’。

5.4.3 ue(v) 与 se(v) 的解析过程

5.4.3.1 k 阶指数哥伦布码

解析 k 阶指数哥伦布码时，首先从比特流的当前位置开始寻找第一个非零比特，并将找到的零比特个数记为 leadingZeroBits ，然后根据 leadingZeroBits 计算 CodeNum 。用伪代码描述如下：

$\text{leadingZeroBits} = -1$

```

for ( b = 0; ! b; leadingZeroBits++ )
    b = read_bits(1)
    CodeNum = 2leadingZeroBits + k - 2k + read_bits(leadingZeroBits + k)

```

表 140 给出了 0 阶、1 阶、2 阶和 3 阶指数哥伦布码的结构。指数哥伦布码的比特串分为“前缀”和“后缀”两部分。前缀由 leadingZeroBits 个连续的‘0’和一个‘1’构成。后缀由 leadingZeroBits + k 个比特构成，即表中的 xi 串，i 的范围为从 0~(leadingZeroBits + k - 1)，每个 xi 的值为 0 或 1。

表 140 k 阶指数哥伦布码表

阶数	码字结构	CodeNum 取值范围
k=0	1	0
	0 1 x0	1…2
	0 0 1 x1 x0	3…6
	0 0 0 1 x2 x1 x0	7…14

k=1	1 x0	0…1
	0 1 x1 x0	2…5
	0 0 1 x2 x1 x0	6…13
	0 0 0 1 x3 x2 x1 x0	14…29

k=2	1 x1 x0	0…3
	0 1 x2 x1 x0	4…11
	0 0 1 x3 x2 x1 x0	12…27
	0 0 0 1 x4 x3 x2 x1 x0	28…59

k=3	1 x2 x1 x0	0…7
	0 1 x3 x2 x1 x0	8…23
	0 0 1 x4 x3 x2 x1 x0	24…55
	0 0 0 1 x5 x4 x3 x2 x1 x0	56…119

5.4.3.2 ue(v)

ue(v) 描述的语法元素使用 0 阶无符号指数哥伦布码，其语法元素其语法元素的取值等于 CodeNum。

5.4.3.3 se(v)

se(v) 描述的语法元素使用 0 阶指数哥伦布码，其语法元素的取值与 CodeNum 的映射关系见表 141。

表 141 se(v) 中 CodeNum 与语法元素取值的映射关系

CodeNum	语法元素值
0	0
1	1
2	-1
3	2
4	-2
5	3
6	-3
k	$(-1)^{k+1} \times \text{Ceil}(k \div 2)$

6 音频部分

6.1 总体描述

6.1.1 模拟信号和数字信号之间转换

模拟信号和数字信号之间的转换描述如下：

- a) 模拟信号到数字线性 PCM 信号转换：
 - 麦克风；
 - 输入电平调节设备；
 - 抗混叠滤波器；
 - 采样保持设备，采样频率为 16 kHz、24 kHz、32 kHz 和 48 kHz；
 - 模拟信号转换为 16 比特数字线性 PCM，采用二进制补码表示。
- b) 数字线性 PCM 信号到模拟信号转换：
 - 16 比特数字线性 PCM 信号转换为模拟信号；
 - 转换保持设备；
 - 补偿重建滤波器；
 - 输出电平调节设备；
 - 耳机或喇叭。

6.1.2 编解码框架的描述

图 22 和图 23 给出了音频编码器和解码器框图。

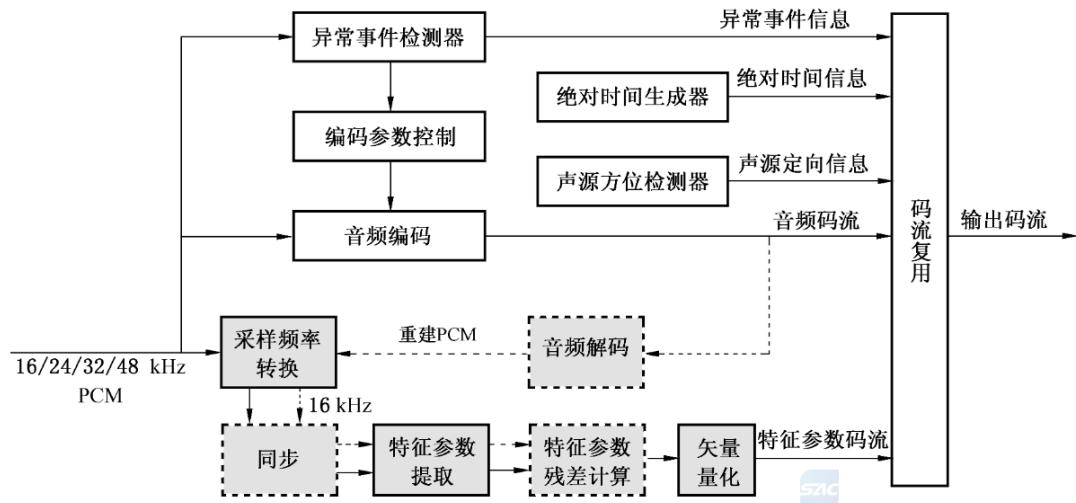


图 22 音频编码器框图

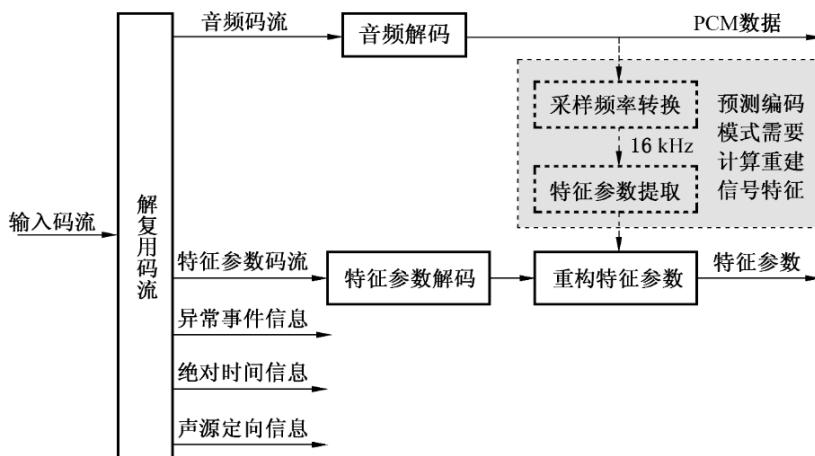


图 23 音频解码器框图

在编码器端,输入声音信号首先经过异常事件检测器,如果检测到异常(比如尖叫声、枪声、爆炸声等),将检测结果传递给编码参数控制模块,编码参数控制模块将根据检测到的事件的重要性和音频编码器的编码参数,以实现变质量音频编码。音频编码器低频和高频用不同的方法进行编码,低频使用基于 ACELP 和 TAC 切换的双核编码器进行编码,高频用相当少的比特进行 BWE 编码。识别特征参数编码提供两种编码模式:直接编码模式和预测编码模式。直接编码模式直接对提取识别特征参数进行矢量量化;而预测编码模式则需要对音频编码器的码流进行解码得到重建信号,重建信号和原始信号分别提取识别特征参数,使用重建信号识别特征参数作为原始信号识别特征参数的预测,最后对预测残差进行矢量量化。如果输入信号采样频率不是 16 kHz,则输入信号先经过采样频率转换模块,转换为 16 kHz 采样的信号,再进行识别特征参数提取。对于预测编码模式,由于重建信号和原始信号之间存在一定的延迟,为了保证所提取的识别特征参数在时间上对应关系,需要经过同步模块同步。最后音频码流、识别特征参数码流、异常事件信息、声源定向信息和绝对时间信息复用成一路码流。

在解码器端,先进行解复用以得到音频码流、识别特征参数码流、异常事件信息、声源定向信息和绝对时间信息。然后音频解码器直接解码输出重建音频信号;识别特征参数解码器从码流中解码出识别特征参数,如果当前编码模式是直接编码模式,则解码得到的就是最终的识别特征参数;如果当前解码

模式是预测编码模式,则解码得到的是识别特征参数的残差,需对音频解码器输出的重建信号,先经过采样频率转换模块,转换为16 kHz采样的信号,再提取识别特征参数作为预测值,最后这两部分相加就得到了最终的识别特征参数。

6.1.3 音频编解码描述

图24给出了双核音频编码器流程图。输入音频信号首先经过预处理,分成两个频带,分别是低频信号和高频信号,采样频率都是 $F_s/2$ 。然后低频信号使用基于ACELP和TAC切换的双核编码器,ACELP是基于时域预测编码技术,适合语音信号和瞬态信号,TAC是基于变换域编码技术,更适合音乐信号和稳态信号。高频信号使用BWE进行编码。最后将低频参数、高频参数和编码模式信息复用成一路码流。

图25给出了双核音频解码器流程图。首先通过码流解复用,得到低频参数、高频参数和编码模式信息。然后低频参数通过ACELP和TAC双核解码,高频参数通过BWE解码。解码后低频信号和高频信号通过后处理恢复成全带信号。

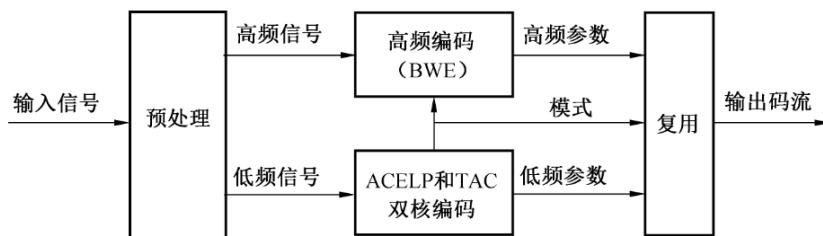


图24 双核音频编码器流程图

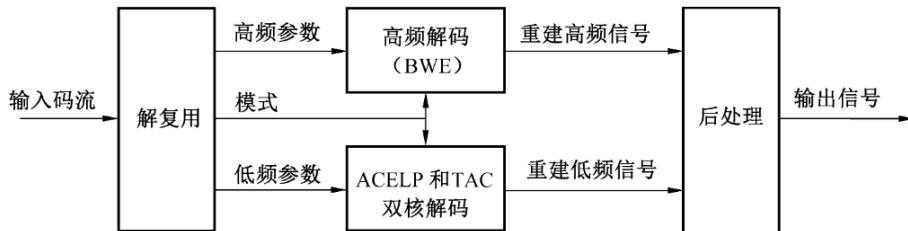


图25 双核音频解码器流程图

6.1.4 识别特征参数编码描述

图26给出了识别特征参数编码框图,整个框架主要分为两个模块:特征提取和特征压缩。16 kHz采样频率的宽带语音信号先进行去直流偏置,噪声消除,波形预处理,再进行倒谱计算,最后对得到的倒谱特征进行去信道干扰的均衡处理。VAD模块检测语音帧和非语音帧,输出1比特的VAD标志位,同识别特征参数合并压缩,具体VAD算法描述参见附录I。识别特征参数选取的是13维MFCC系数和一个对数能量系数,构成一个14维的特征矢量。特征提取时帧长为25 ms(16 kHz采样频率下400个样本),帧移为10 ms(16 kHz采样频率下160个样本)。特征压缩模块使用矢量量化对特征矢量或残差矢量进行量化。

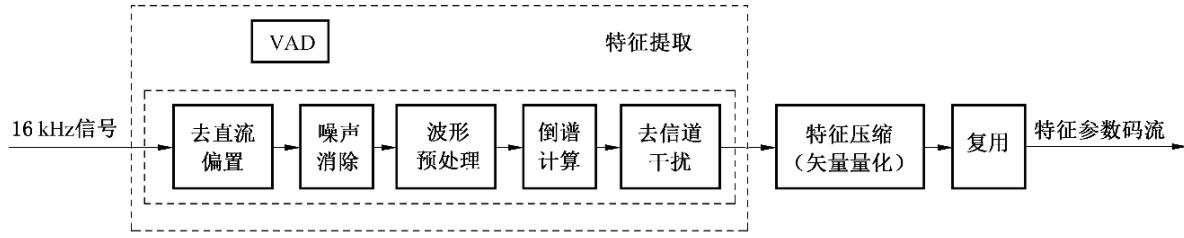


图 26 识别特征参数编码框图

识别特征参数编码提供两种编码模式：直接编码模式和预测编码模式。直接编码模式直接对提取特征进行矢量量化；而预测编码模式则需要对音频编码器的码流进行解码得到重建信号，重建信号和原始信号分别提取识别特征参数，使用重建信号识别特征参数作为原始信号识别特征参数的预测，最后对预测残差进行矢量量化。详细编码过程见 6.2.6。

6.2 编码器功能描述

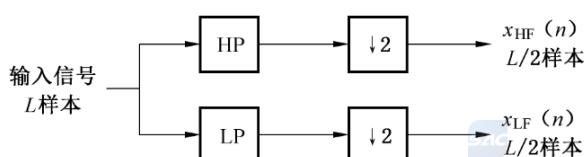
6.2.1 预处理

6.2.1.1 采样频率转换

音频信号的输入采样频率有 16 kHz、24 kHz、32 kHz 和 48 kHz，需要将各种不同采样频率的输入信号在编码之前的预处理中进行重采样，转换为内部采样频率 F_s 。同理，在解码的后处理中同样需要采样频率转换。

6.2.1.2 低频信号和高频信号分解

采样频率为 F_s 的输入信号通过截止频率在 $F_s/4$ 的低通滤波器，再作 2 倍临界下采样，得到 $F_s/2$ 采样低频信号 $x_{LF}(n)$ ；同样，经过截止频率在 $F_s/4$ 的高通滤波器，再作 2 倍临界下采样，得到 $F_s/2$ 采样高频信号 $x_{HF}(n)$ ，见图 27。



说明：

L ——音频超帧长度。

图 27 低频和高频信号分解框图

6.2.1.3 低频信号高通滤波

低频信号经过高通滤波器，目的是为了滤掉不需要的低频成分。高通滤波器的传递函数见式(1)：

$$H_{h1}(z) = \frac{b_0 - b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} + a_2 z^{-2}} \quad \dots \dots \dots (1)$$

式中的滤波器系数取决于采样频率。