Week 8 : Session 1.

POLAR CODES : AN INTRODUCTION

As we aware, Polar codes have find application in 5G communication system. They have been used in control channel. In this section, We'll talk about

- What are Polar codes, how do they work.
- How do we encode data using Polar code.

⊙ Polar codes are a class of capacity-achieving codes introduced by Arikan.

⊙ The channel polarization phenomenon consists of a transformation, which produces N synthetic bit-channels from N independent copies of a binary input discrete memoryless channel (B-DMC).

⊙ The new synthetic channels are then polarized in the sense that, each of them can transmit a single bit at a different reliability. (ii) with a different probability of being decoded correctly.

Channel polarization - The channels are polarized in such a way that, there are some channels which are very good and there are channels which are very bad. We would like to transmit bits over channels which are good, and not transmit anything over channels which are bad.

We can think of it like, if we are talking about single bit channel, Capacity of a good channel is 1 bit and capacity of a bad channel is 0 bit.

So as a result ~~of~~ polarization, we are kind-of producing this N synthetic channels. Some of them are very good channels, so we can send the data through them.

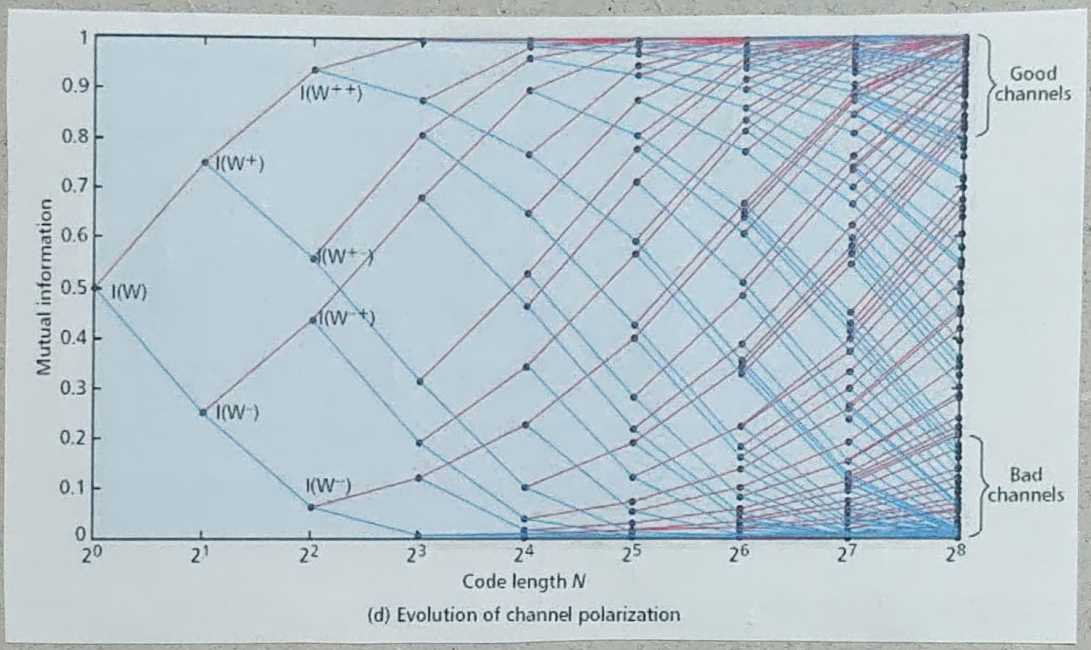
without any problem, and some of them are very bad channels which we will not use for communication.

" So, when I say the channels are polarized, I should be able to transmit a bit through that channel reliably.

⊙ If N is large enough, the mutual information of the synthetic channels is either close to 0 (completely noisy channels) or close to 1 (perfectly noiseless channels), resulting in channels of extreme capacities.

So, as a result of this polarization, we'll get some channels which are very good and we'll get some channels which are very bad.

⊙ These codes have explicit proof for achieving capacity.



(d) Evolution of channel polarization

In this plot, On the x-axis, we have Codeword length N, and on the y-axis, we have Mutual information.

Initially we have mutual information of 0.5 (we are talking about single bit channel) and we want to see as a result of polarization, what is happening.

When $N = 2$, as a result of polarization we get two Synthetic channels, One which has mutual Information of 0.75 and another which has mutual information of 0.25.

When $N = 4$, now we have four Synthetic channels. with mutual information of 0.95, 0.55, 0.45 and 0.05.

As N increases, we see that, we get Good channels and Bad channels. (ii) It is kind of polarizing the channel towards Good channels (capacity close to 1) and Bad channels (capacity close to 0).

So, this is how the effect of polarization happens.

Polar Transform (Simple case of 2 bits)

⊙ The Polar transform for 2 bits is given by

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Codeword $\underbrace{[v_1 \; v_2]}_{\text{output}} = \overbrace{[u_1 \; u_2]}^{\text{Input}} G_2 = \underset{1\times 2}{[u_1 \; u_2]} \underset{2\times 2}{\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}}$

$$= \underset{1\times 2}{[u_1 + u_2 \quad u_2]}$$

⊙ The Polar transform for 4 bits is given by

$$G_4 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Kronecker product

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Codeword $[v_1\ v_2\ v_3\ v_4] = [u_1\ u_2\ u_3\ u_4]\ G_4$

$$= \underset{1\times 4}{[u_1\ u_2\ u_3\ u_4]} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}_{4\times 4}$$

$$= \begin{bmatrix} u_1+u_2+u_3+u_4 & u_2+u_4 & u_3+u_4 & u_4 \end{bmatrix}_{1\times 4}$$

(a) $v_1 = u_1+u_2+u_3+u_4$

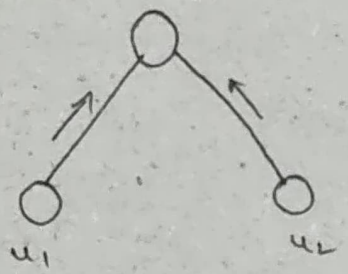
$v_2 = u_2+u_4$

$v_3 = u_3+u_4$

$v_4 = u_4$

This can also be represented using Binary Tree. This would be useful when we do decoding of polar codes.

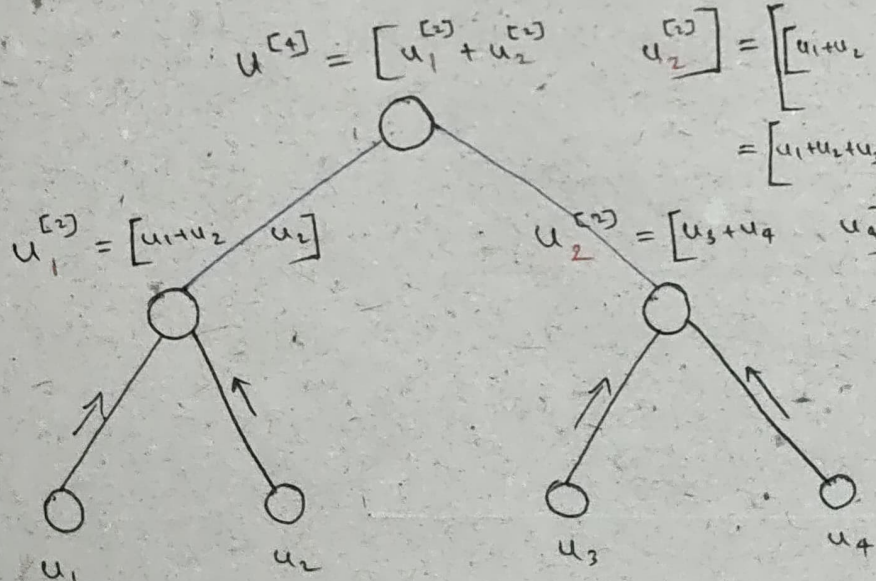
$u^{[2]} = [u_1+u_2 \quad u_2]$



$u^{[4]} = [u_1^{[2]} + u_2^{[2]} \quad u_2^{[2]}] = [[u_1+u_2 \ u_2] + [u_3+u_4 \ u_4] \quad [u_3+u_4 \ u_4]]$

$= [u_1+u_2+u_3+u_4 \quad u_2+u_4 \quad u_3+u_4 \quad u_4]$

$u_1^{[2]} = [u_1+u_2 \quad u_2]$ $\qquad$ $u_2^{[2]} = [u_3+u_4 \quad u_4]$



④ Similarly, the polar transform for 8-bits is given by

$$G_8 = G_4 \otimes G_2$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Codeword, $v = [v_1\ v_2\ v_3\ v_4\ v_5\ v_6\ v_7\ v_8]$

$$= [u_1\ u_2\ u_3\ u_4\ u_5\ u_6\ u_7\ u_8]\ G_8$$

$\Rightarrow v_1 = u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8$

$v_2 = u_2 + u_4 + u_6 + u_8$

$v_3 = u_3 + u_4 + u_7 + u_8$

$v_4 = u_4 + u_8$

$v_5 = u_5 + u_6 + u_7 + u_8$

$v_6 = u_6 + u_8$

$v_7 = u_7 + u_8$

$v_8 = u_8$

In general,

$$G_{2^n} = G_2 \otimes G_{2^{n-1}}$$

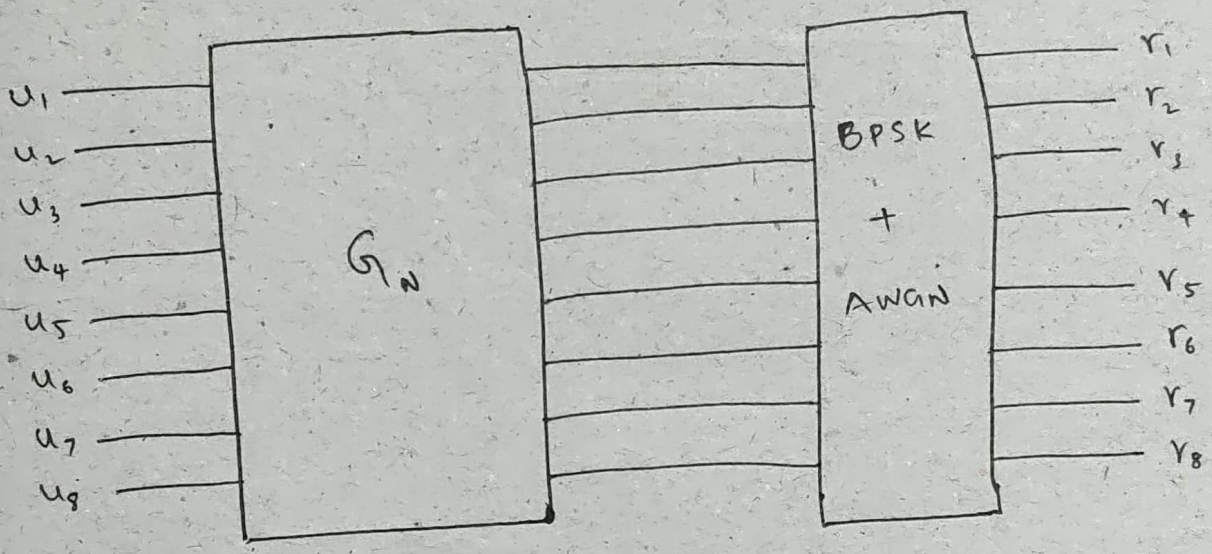
$$= G_2 \otimes G_2 \otimes G_{2^{n-2}}$$

$$= \underbrace{G_2 \otimes G_2 \otimes \cdots \otimes G_2}_{n \text{ times}}$$

Thus, $G_{2^n} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n}$ where $N = 2^n$.

G_N is a $N \times N$ matrix obtained from Kronecker product of 2×2 kernel.

Polar Transform and BPSK + AWGN

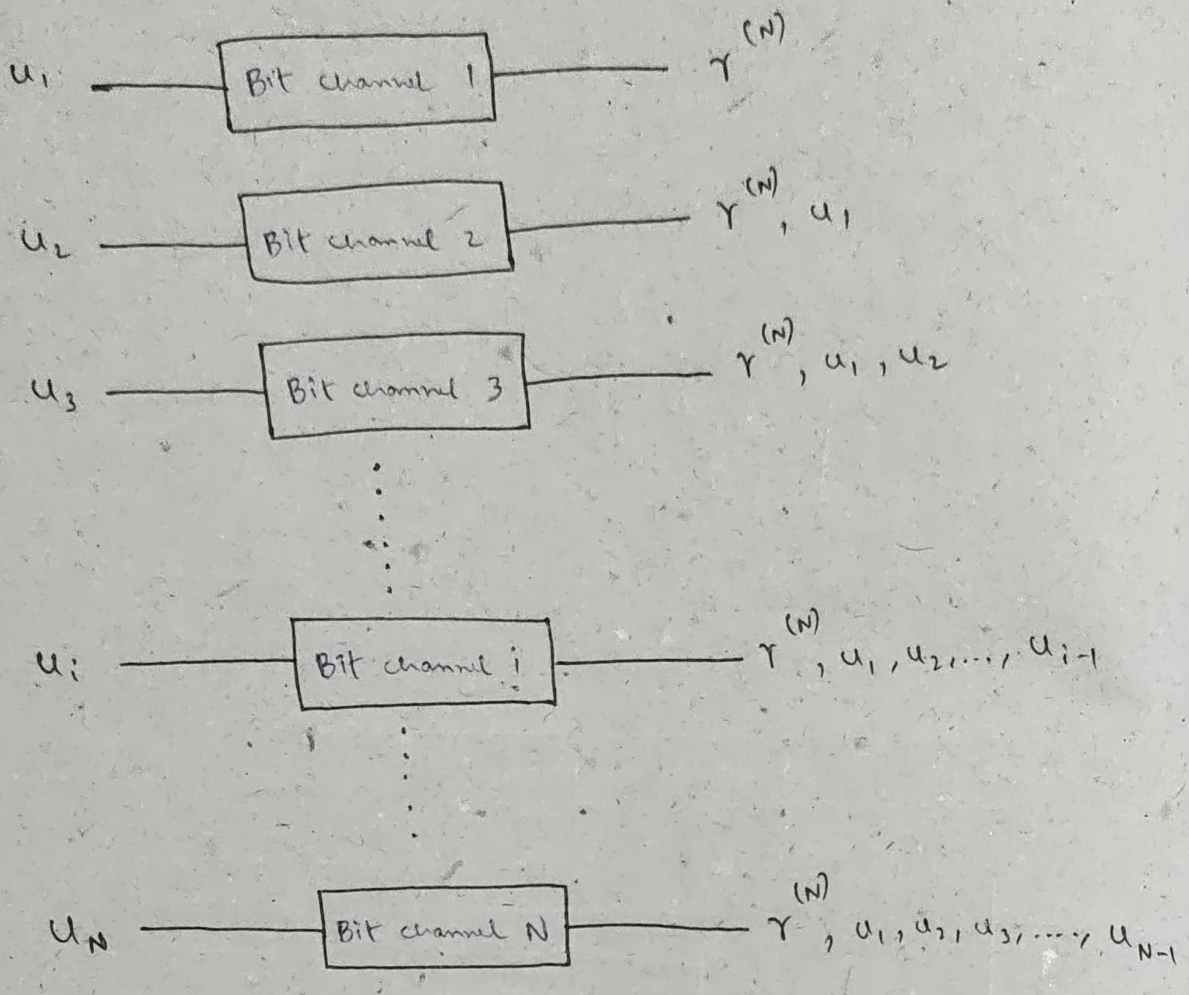


So, our communication is something like this. We are sending these u_i's, we apply polar transformation. Then these bits are sent toward a communication channel (AWGN), where bits are mapped to +1 and -1 (BPSK modulated signal). And we get the received sequence

$$r^{(N)} = [r_1\ r_2\ r_3\ \cdots\ r_N] \quad : \text{received vector.}$$

Now, from this received sequence, we need to estimate the u_i's.

Bit channels and Polarization



So, as a result of this polarization, we are kind-of creating these synthetic bit channels, as above shown. And we have the outputs from each channel. What is happening here is, as a result of polarization, the mutual information is going towards 1 (increases), and some of their mutual information is going towards 0 (decreases).

So, when we want to use these channels, we would use the good channels, while we would not use the bad channels.

⊙ Bit channels polarize and can be ordered based on "quality": quality varies from very good to very bad.

Reliability sequence. (Increasing order)

As we saw that, as a result of this polarization, we are creating these N synthetic channels, where some of them are good and some of them are bad. So, we are kind of arranging them in the increasing order of their capacity / reliability.

So, what we would do is, given a block size, and given a rate, we know how many information bits we need to send. And we are going to send those information only through these best K synthetic channels.

Example:

⊙ Design a Rate, $R = \frac{1}{2}$ code.

Given $N = 8$

WKT, $R = \frac{k}{N} = \frac{1}{2}$

$\Rightarrow \boxed{k = 4}$

This means, we need to send 4 information bits through this channel. Now, the question is, which channels should be used?

Reliability order.

$N = 8$; 1 2 3 5 4 6 7 8

↑ ..... ↑

WORST CHANNEL BEST CHANNEL

So, we can use channels 4 6 7 8 and not going to use the channels 1 2 3 5. Instead, we put a Frozen bit of 0. and send it over the channel. So, we are not going to send any information through these "Bad channels".

Similarly, for N = 16, the reliability order is

N = 16 ; 1 2 3 5 9 4 6 10 | 7 11 13 8 12 14 15 16

WORST CHANNELS | BEST CHANNELS

To design a Rate, $R = \frac{1}{2}$ code,

$$R = \frac{k}{N} = \frac{1}{2^n}$$

$$\Rightarrow \boxed{k = 8}$$

So, we are going to use the best 8 channels to send the information bits, and not going to use the worst channels. In other words, we just send a Frozen bit '0' through these 8 worst channels.

So, we are effectively utilizing the BEST 8 channels to send information

---

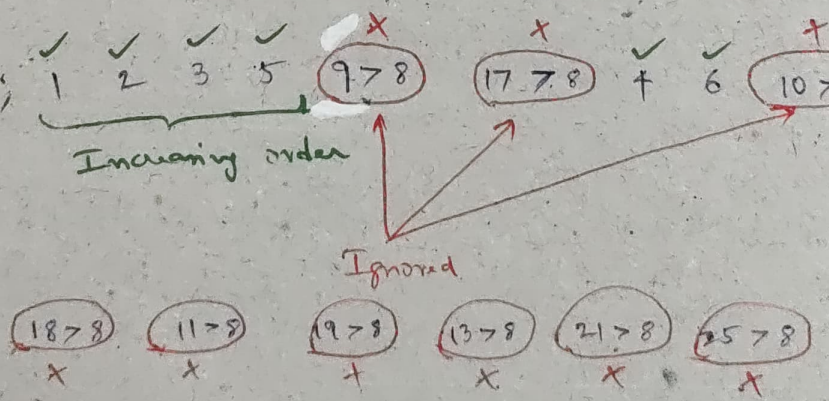
Similarly for N = 32, the reliability sequence is

N = 32 ; 1 2 3 5 9 17 4 6 10 7 18 11 19 13
21 25 8 12 20 14 15 22 27 26 23 29 16 24
28 30 31 32

Again, this reliability sequence is obtained from the ability of this bit channel to send information. So, we have arranged them in the increasing order of mutual information

---

From the reliability sequence for N = 32, we can derive the reliability sequence for N = 8 (or) N = 16. For example,



N = 16 ; 1 2 3 5 9 (17>16) 4 6 10 7 (18>16) 11 (19>16)

13 (21>16) (25>16) 8 12 (20>16) 14 15 (22>16) (27>16)

(26>16) (23>16) (24>16) 16

So, if we are given a reliability sequence for large N, we can find out the reliability sequence for small N. And, knowing the reliability sequence is important because, given an N, given a rate, which channels to use to send information would be decided by the reliability sequence of those channels.

In 5G-NR, it is specified that the reliability sequence for N = 1024.

1 2 3 5 9 17 33 4 6 65 10 7 18 11 19 129 13 34 66 21 257 35 25 37 8 130 67 513 12 41 69 131 20 14 49 15 73 258 22 133 36 259 27 514 81 38 26 23
137 261 265 39 515 97 68 42 145 29 70 43 517 50 75 273 161 521 289 529 193 545 71 45 132 82 51 74 16 321 134 53 24 135 385 77 138 83 57 28 98
40 260 85 139 146 262 30 44 99 516 89 141 31 147 72 263 266 162 577 46 101 641 52 149 47 76 267 274 518 105 163 54 194 153 78 165 769 269
275 519 55 84 58 522 113 136 79 290 195 86 277 523 59 169 140 100 87 61 281 90 291 530 525 197 142 102 148 177 143 531 322 32 201 91 546 293
323 533 264 150 103 106 305 297 164 93 48 268 386 547 325 209 387 151 154 166 107 56 329 537 578 549 114 155 80 270 109 579 225 167 520 553
196 271 642 524 276 581 292 60 170 561 115 278 157 88 198 117 171 62 532 526 643 282 279 527 178 294 389 92 585 770 199 173 121 202 337 63
283 144 104 179 295 94 645 203 593 324 393 298 771 108 181 152 210 285 649 95 205 299 401 609 353 326 534 156 211 306 548 301 110 185 535
538 116 168 226 327 307 773 158 657 330 111 118 213 172 777 331 227 550 539 388 309 217 417 272 280 159 338 551 673 119 333 580 541 390
174 122 554 200 785 180 229 339 313 705 391 175 555 582 394 284 123 449 354 562 204 64 341 395 528 583 557 182 296 286 233 125 206 183 644
563 287 586 300 355 212 402 186 397 345 587 646 594 536 241 207 96 328 565 801 403 357 308 302 418 214 569 833 589 187 647 405 228 897 595
419 303 650 772 361 540 112 332 215 310 189 450 218 409 610 597 552 651 230 160 421 311 542 774 611 658 334 120 601 340 219 369 653 231
392 314 451 543 335 234 556 775 176 124 659 613 342 778 221 315 425 396 674 584 356 288 184 235 126 558 661 617 343 317 242 779 564 346
453 398 404 208 675 559 786 433 358 188 237 665 625 588 781 706 127 243 566 399 347 457 359 406 304 570 245 596 190 567 677 362 707 590
216 787 648 349 420 407 465 681 802 363 591 410 571 789 598 573 220 312 709 599 602 652 422 793 803 612 603 411 232 689 654 249 370 191
365 655 660 336 481 316 222 371 614 423 426 452 615 544 236 413 344 373 776 318 223 427 454 238 560 834 805 713 835 662 809 780 618 605
434 721 817 837 348 898 244 663 455 319 676 619 899 782 377 429 666 737 568 841 626 239 360 458 400 788 592 679 435 678 350 246 459 667
621 364 128 192 783 408 437 627 572 466 682 247 708 351 600 669 791 461 250 683 574 412 804 790 710 366 441 629 690 375 424 467 794 251
372 482 575 414 604 367 469 656 901 806 616 685 711 430 795 253 374 606 849 691 714 633 483 807 428 905 415 224 664 693 836 620 473 456
797 810 715 722 838 717 865 811 607 913 723 697 378 436 818 320 622 813 485 431 839 668 489 240 379 460 623 628 438 381 819 462 497 670
680 725 842 630 352 468 439 738 252 463 443 442 470 248 684 843 739 900 671 784 850 821 729 929 792 368 902 631 686 845 634 712 254 692
825 903 687 741 851 376 445 471 484 416 486 906 796 474 635 745 853 961 866 694 798 907 716 808 475 637 695 255 718 576 914 799 812 380
698 432 608 490 867 724 487 909 719 814 477 857 840 726 699 915 753 869 820 815 440 930 491 624 672 740 917 464 844 382 498 931 822 727
962 873 493 632 730 701 444 742 846 921 383 823 852 731 499 881 743 446 472 636 933 688 904 826 501 847 746 827 733 447 963 937 476 854
868 638 908 488 696 747 829 754 855 858 505 800 256 965 910 720 478 916 639 749 945 870 492 700 755 859 479 969 384 911 816 977 871 918
728 494 874 702 932 757 861 500 732 824 923 875 919 503 934 744 761 882 495 703 922 502 877 848 993 448 734 828 935 883 938 964 748 506
856 925 735 830 966 939 885 507 750 946 967 756 860 941 831 912 872 640 889 480 947 751 970 509 862 758 971 920 876 863 759 949 978 924
973 762 878 953 496 704 936 979 884 763 504 926 879 736 994 886 940 995 981 927 765 942 968 887 832 948 508 890 985 752 943 997 972 891
510 950 974 1001 893 951 864 760 1009 511 980 954 764 975 955 880 982 983 928 996 766 957 888 986 998 987 944 892 999 767 512 989 1002 952
1003 894 976 895 1010 956 1005 1011 958 984 959 988 1013 1000 1017 768 990 1004 991 1006 960 1012 1014 896 1007 1015 1018 1019 992 1021
1008 1016 1020 1022 1023 1024

Now that we understood how does this polarization work, lets now talk about how are we going to encode the information.

(N, K) Polar codes

→ Information bit size

→ Block length.

- The goal of code design of an (N, K) polar code is to identify the K best synthetic channels, namely the channels providing the highest reliability, and use them to transmit the information bits.
- The estimation of the reliability of each synthetic channel allows to sort them in reliability order and assign the K information bits to the most reliable channels.
- WKT, Code word length is $N = 2^n$.
- Form a vector u of length N bits as follows.
  - Find N-K least reliable channels from reliability sequence
  - Set u_i for those N-K channels to zero. These are known as Frozen positions.
  - The remaining K bits of u will be the message bits.
- Codeword is generated as $V = u G_N$.

Example

(i) (8, 4) Polar code ; N = 8, K = 4.

Reliability sequence : 1 2 3 5 4 6 7 8

↑ Least Reliable channel

↑ Most Reliable channel

Frozen bit location : 1 2 3 5 (∵ N - K = 4).

Message bit location : 4 6 7 8

(ii) (16, 10) Polar code ; $N = 16$, $K = 10$.

- Reliability sequence : 1 2 3 5 9 4 6 10 7 11 13 8 12 14 16.

Frozen bit location : 1 2 3 5 9 4 ($\because N - K = 6$)

Message bit location : 6 10 7 11 13 8 12 14 15 16

(iii) (32, 20) Polar code ; $N = 32$, $K = 20$

Reliability sequence : 1 2 3 5 9 17 4 6 10 7 18 11, 19 13 21 25 8 12 20 14 15 22 27 26 23 29 16 24 28 30 31 32

Frozen bit location : 1 2 3 5 9 17 4 6 10 7 18 11 ($\because N - K = 12$)

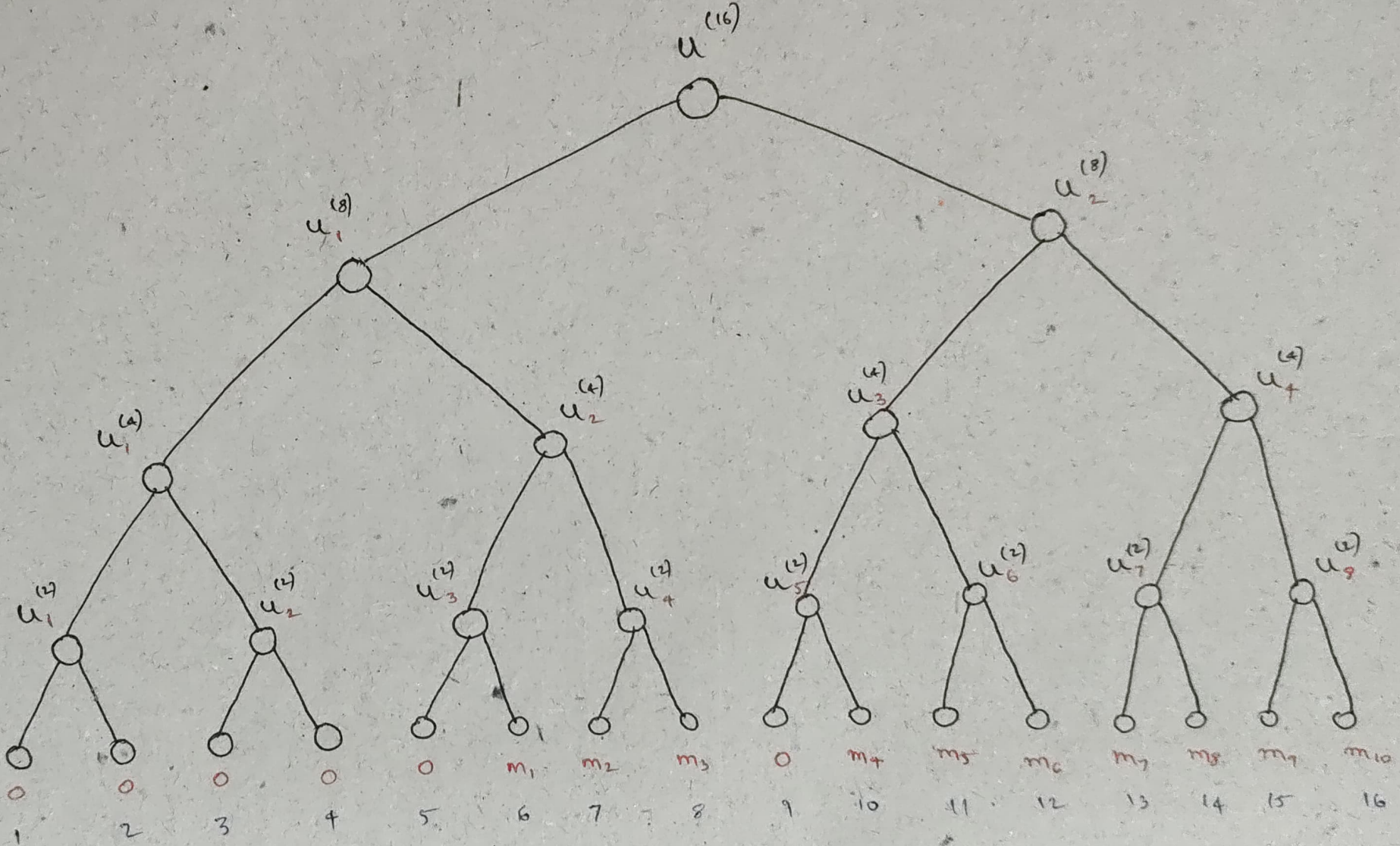
- Message bit location : 19 13 21 25 8 12 20 14 15, 22 27 26 23 29 16 24 28 30 31 32

In the Frozen bit location, we set the u_i's to be zero.

In the Message bit location, we send the actual information bits.

Binary Tree representation : (16, 10)

- PTO.
- Note that, through the channels 1, 2, 3, 4, 5, 9, we are not sending anything. These are Frozen bit locations. So we set u_i's = 0 in these locations.
- Whereas, in the locations corresponding to channels 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, we send the 10 message bits $m_1, m_2, \ldots, m_{10}$.
- This is how we encode data using Polar Transform.


$u^{(16)}$
$u_1^{(8)}$
$u_2^{(8)}$
$u_1^{(4)}$
$u_2^{(4)}$
$u_3^{(4)}$
$u_4^{(4)}$
$u_1^{(2)}$
$u_2^{(2)}$
$u_3^{(2)}$
$u_4^{(2)}$
$u_5^{(2)}$
$u_6^{(2)}$
$u_7^{(2)}$
$u_8^{(2)}$
0 0 0 0 0 m_1 m_2 m_3 0 m_4 m_5 m_6 m_7 m_8 m_9 m_{10}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

This Tree Structure representation we are going to exploit when we do decoding of Polar codes.

Week 8 : Session 2

Decoding of Polar codes - 1 : Successive Cancellation decoder

As the name suggests, in Successive Cancellation decoder, we try to decode 1 bit at a time and then we remove the effect of that from the receive signal, and then one-by-one we are essentially decoding the bits.

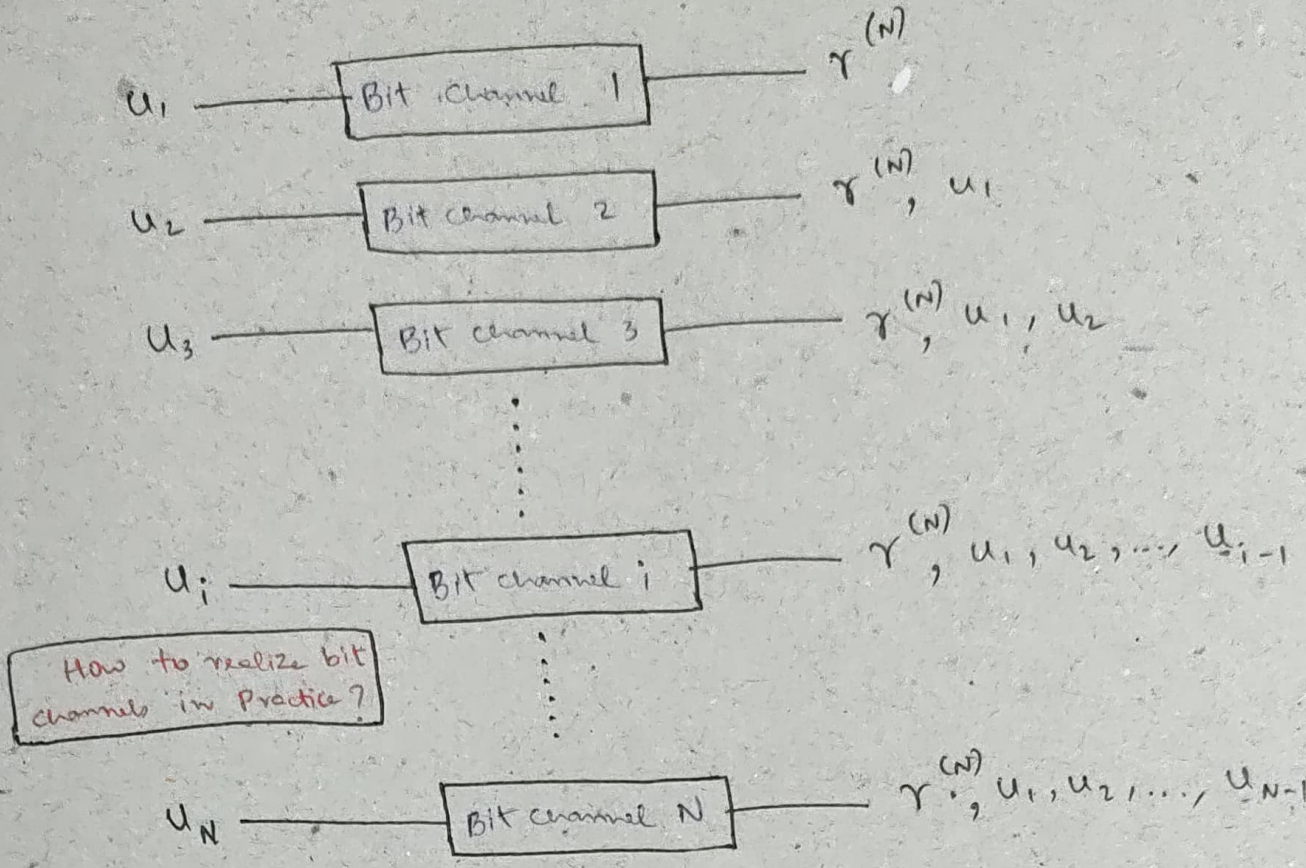
So, before we go into decoding, lets briefly go over the encoding part.

(N, K) Polar code

- We have block length, $N = 2^n$.
- We are given a code rate, $R = \frac{K}{N}$.
- So, we have to code a message m of length K bits.
- To do that, we create a vector u of length N bits as follows
  - Find $N-K$ least reliable (worst) channels from reliability sequence
  - Set u_i for those $N-K$ channels to zero (called Frozen positions)
  - m : remaining K bits of u (called Message positions)
- Codeword : $u\, G_N$

where, $G_N \rightarrow$ Polar Transformation for N bits.

Bit channels and Polarization

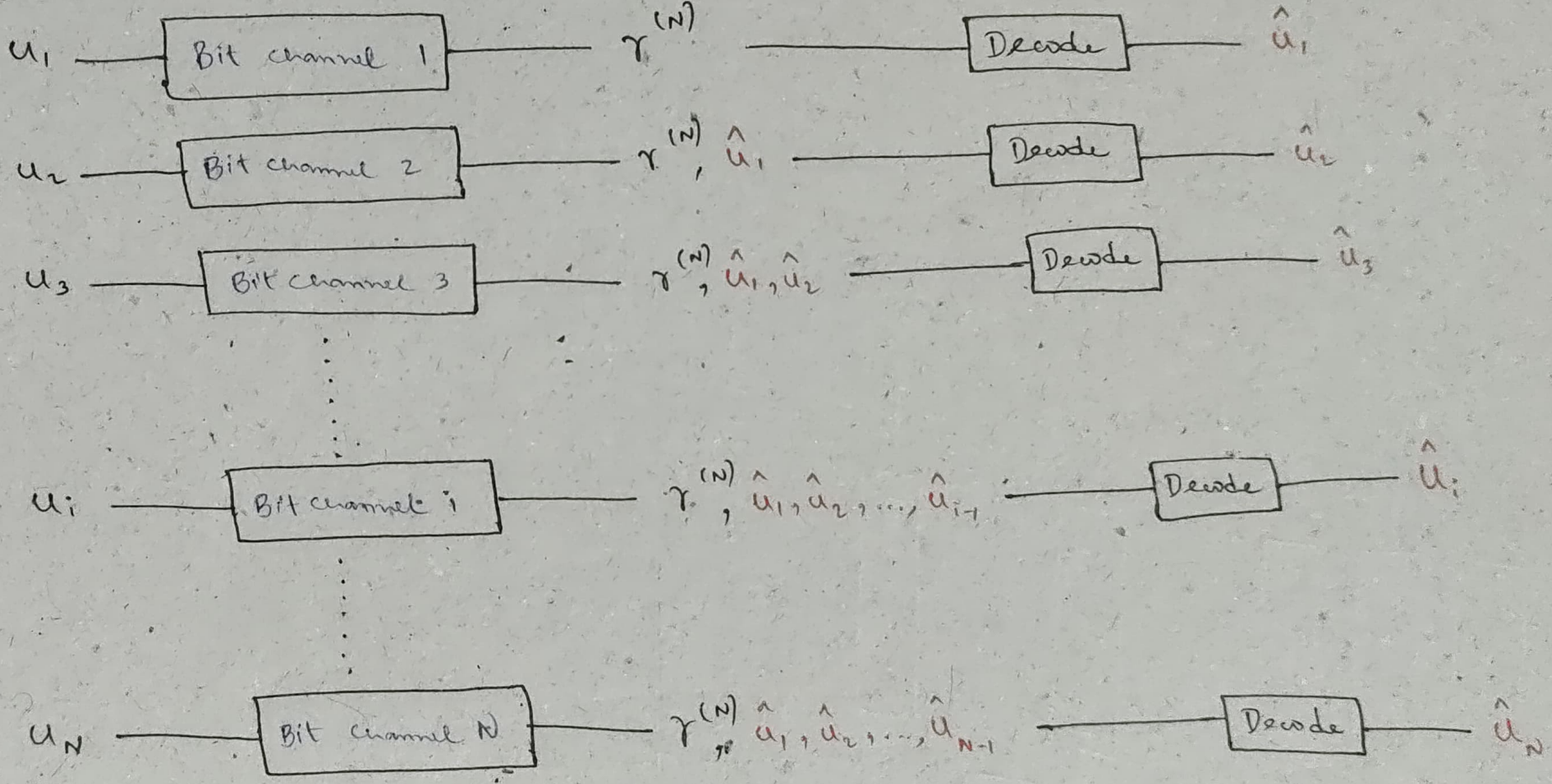


As we know, as a result of polarization, we are kind of making it into syntetic N bit channels.

Now, how do we basically realize this bit channel in practice?

Successive Cancellation (SC) Decoding

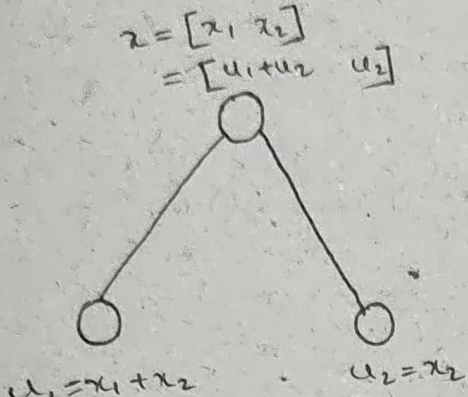
→ PTO

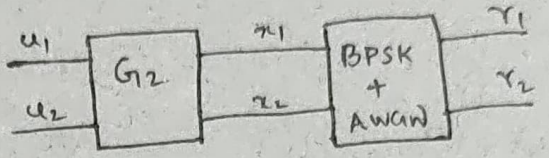

u_1
Bit channel 1
$y^{(N)}$
Decode
$\hat{u}_1$
u_2
Bit channel 2
$y^{(N)}, \hat{u}_1$
Decode
$\hat{u}_2$
u_3
Bit channel 3
$y^{(N)}, \hat{u}_1, \hat{u}_2$
Decode
$\hat{u}_3$
u_i
Bit channel i
$y^{(N)}, \hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{i-1}$
Decode
$\hat{u}_i$
u_N
Bit channel N
$y^{(N)}, \hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{N-1}$
Decode
$\hat{u}_N$

⊙ If bit i is frozen, $\hat{u}_i = 0$

⊙ If bit i is a message bit, $\hat{u}_i$ is found using SISO (Soft Input Soft output) decoder.

We'll take a very simple example of $N=2$.





Basic Building Block of SC decoder : $N=2$

Basically, we are sending u_1 and u_2. As a result of polar transformation, what we are sending is $x_1 = u_1 + u_2$ and $x_2 = u_2$.

Now, if we do $x_1 + x_2 = u_1 + u_2 + u_2$
$= u_1$ | mod-2 operation.

Now, x_1 and x_2 are sent over AWGN channel with BPSK modulation. So, what we get at the receiver is the noisy version of the transmitted sequence (i.e) y_1 and y_2.

Now, how do we decode u_1 and u_2 at the receiver side?

Wkt, $u_1 = x_1 + x_2$ (This is like Single Parity check code)

$u_2 = x_2$

⊙ SISO decode u_1 first (SPC)

⊙ Log-likelihood ratio of the bit u_1

$L(u_1) = f(y_1, y_2) = \text{sgn}(y_1)\,\text{sgn}(y_2)\,\min(|y_1|, |y_2|)$.

If $L(u_1) \geq 0$, then $\hat{u}_1 = 0$

If $L(u_1) < 0$, then $\hat{u}_1 = 1$.

Based on the Log-likelihood value, which is calculated from the received sequence r_1 and r_2, we decide whether the bit u_1 is 0 (or) 1.

⊙ Once $\hat{u}_1$ is decoded, how do we decode u_2?

If $u_1 = 0$, what we receive is

$$x = [x_1 \quad x_2] = [u_1 + u_2 \quad u_2] = [u_2 \quad u_2]$$

Two copies of u_2.
So, this is like a Repetition code.

So, if u_1 is estimated as 0, then decoding u_2 is like decoding a repetition code.

If $u_1 = 1$, what we receive is

$$x = [x_1 \quad x_2] = [u_1 + u_2 \quad u_2] = [1 + u_2 \quad u_2]$$

Complement of u_2
Here also Two copies of u_2
So, this is also like a Repetition code.

$\Rightarrow$

If $\hat{u}_1 = 0$, $L(u_2) = r_2 + r_1$ $\quad (x = [u_2 \quad u_2])$

If $\hat{u}_1 = 1$, $L(u_2) = r_2 - r_1$ $\quad (x = [u_2 \quad u_2])$

⊙ So, decoding u_1 is like decoding a Single Parity check code.

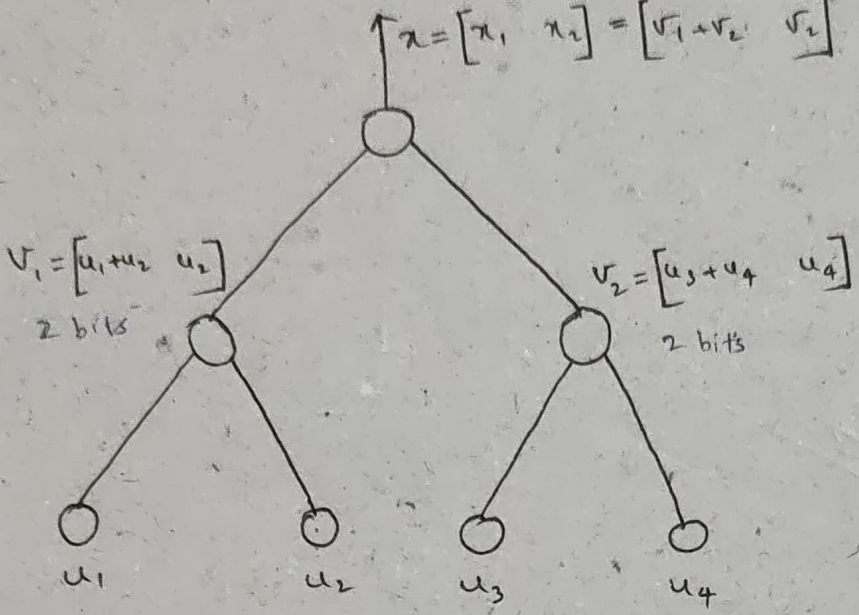
And decoding u_2 is like decoding a Repetition code.

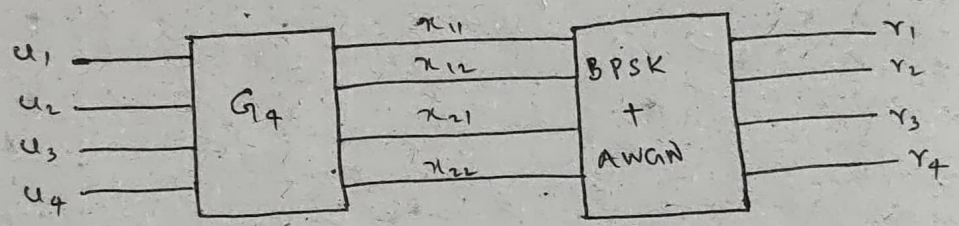
⊙ Basic Building block : Message passing on tree

$$g(x_1, x_2, b) = x_2 + (1 - 2b)x_1$$

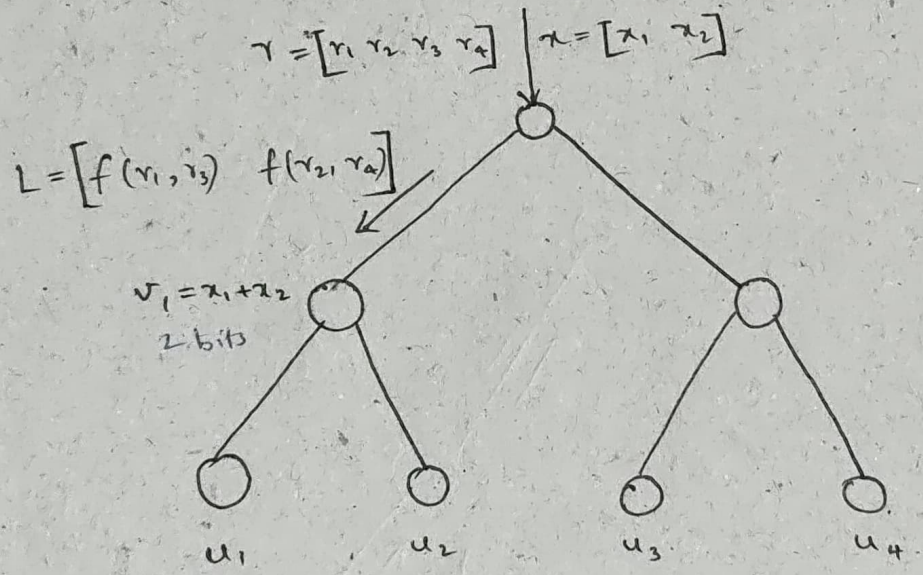
$$f(x_1, x_2, x_3) = \text{sign}(x_3)\,|x_3|\,g(x_1, x_2, |x_3|)$$

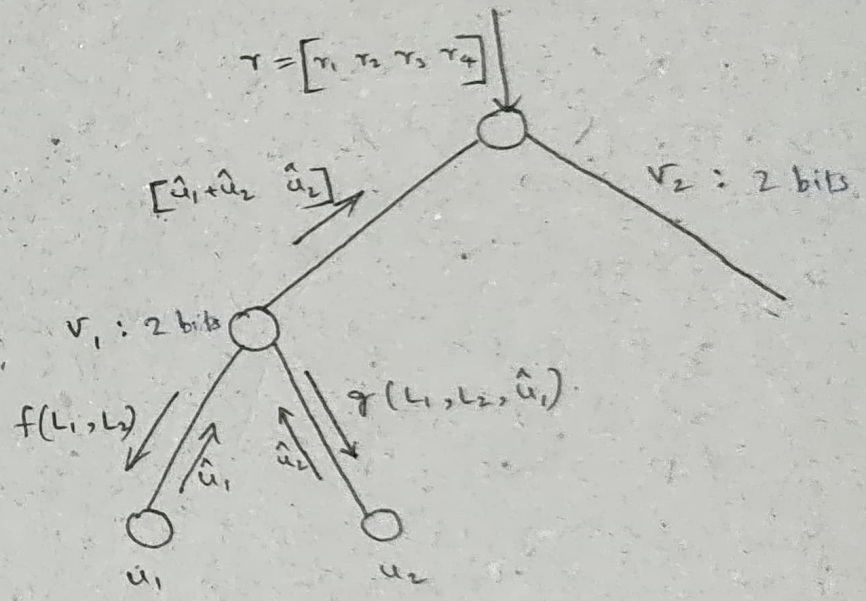
Now, the same concept can be extended for larger N.

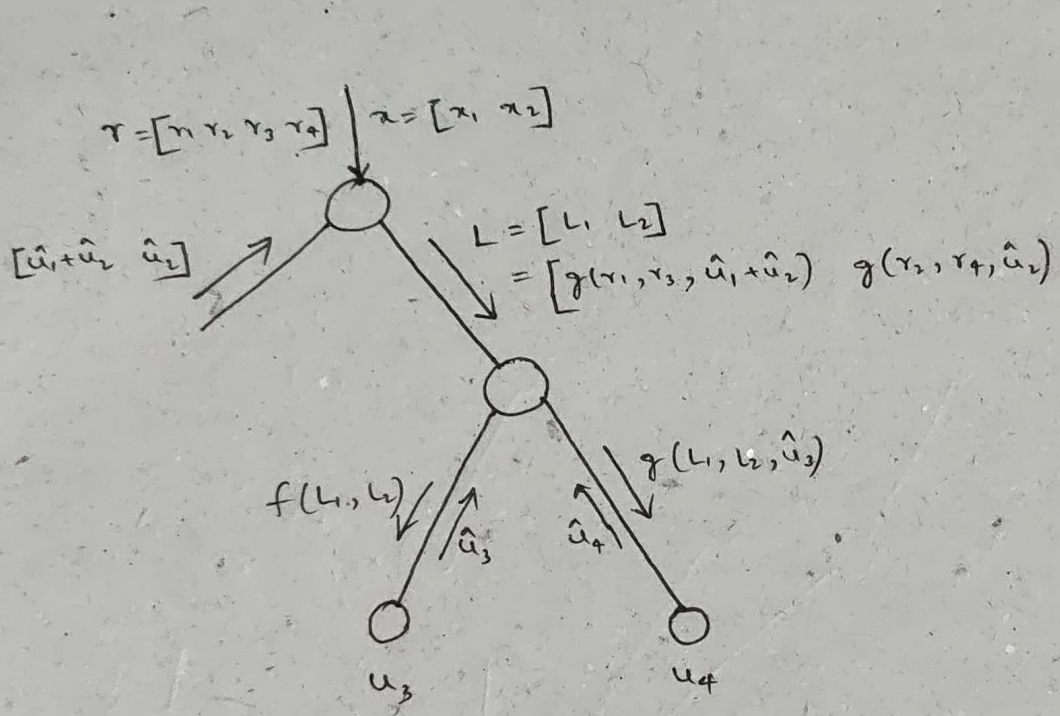


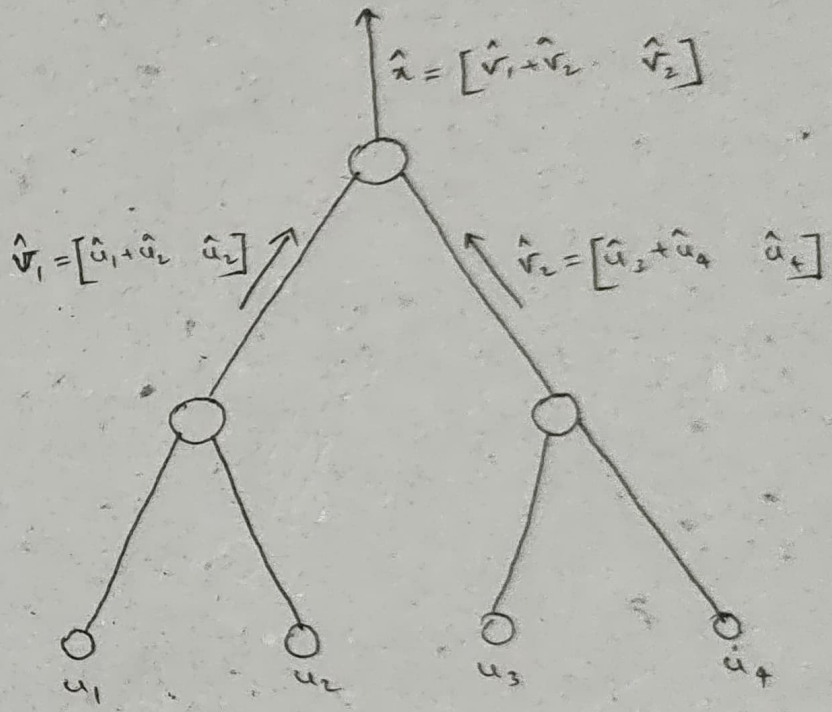


SC decoder : N = 4

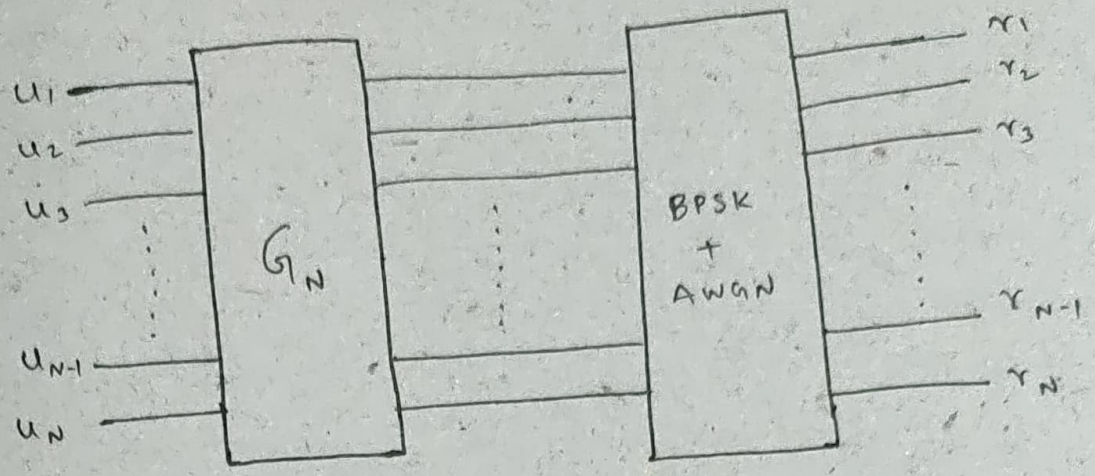

r = [r1 r2 r3 r4]
x = [x1 x2]
L = [f(r1, r3) f(r2, r4)]
v1 = x1 + x2
2 bits
u1
u2
u3
u4


r = [r1 r2 r3 r4]
[û1 + û2 û2]
v2 : 2 bits
v1 : 2 bits
f(L1, L2)
g(L1, L2, û1)
û1
û2
u1
u2


r = [r1 r2 r3 r4]
x = [x1 x2]
[û1+û2 û2]
L = [L1 L2]
= [g(r1, r3, û1+û2) g(r2, r4, û2)
f(L1, L2)
û3
û4
g(L1, L2, û3)
u3
u4


x̂ = [v̂1+v̂2 v̂2]
v̂1 = [û1+û2 û2]
v̂2 = [û3+û4 û4]
u1
u2
u3
u4

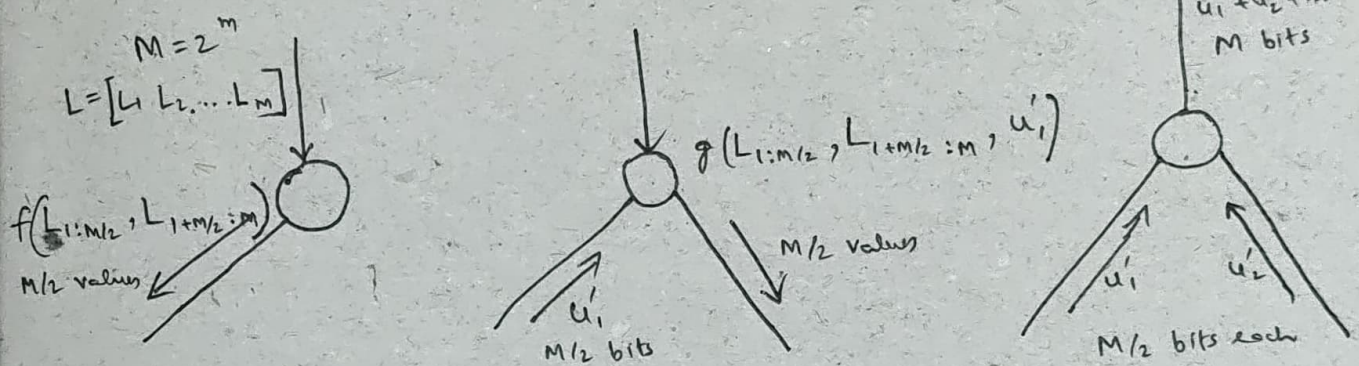
SC decoder Setup (In general)



$$r^{(N)} = [r_1 \; r_2 \; r_3 \; \dots \; r_{N-1} \; r_N] \quad : \text{received vector}$$

SC decoder: Operations in an interior node

Step L | **Step R** | **Step U**



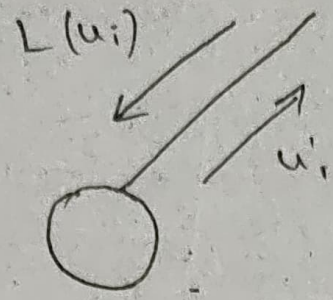
$$f_1(a_{1:p}, b_{1:p}) = [f(a_1, b_1) \; f(a_2, b_2) \dots f(a_p, b_p)]$$

$$g(r_1, r_2, b) = r_2 + (1-2b)\, r_1$$

$$f(r_1, r_2) = \text{sgn}(r_1)\, \text{sgn}(r_2)\, \min(|r_1|, |r_2|)$$

$$g(a_{1:p}, b_{1:p}, c) = [f(a_1, b_1, c) \; f(a_2, b_2, c) \dots f(a_p, b_p, c)]$$

SC decoder : decision in a leaf node



If 'i' is a frozen position : $u_i' = 0$

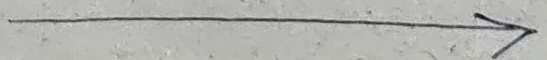
If 'i' is a message position : $u_i' = 0$, if $L(u_i) >= 0$

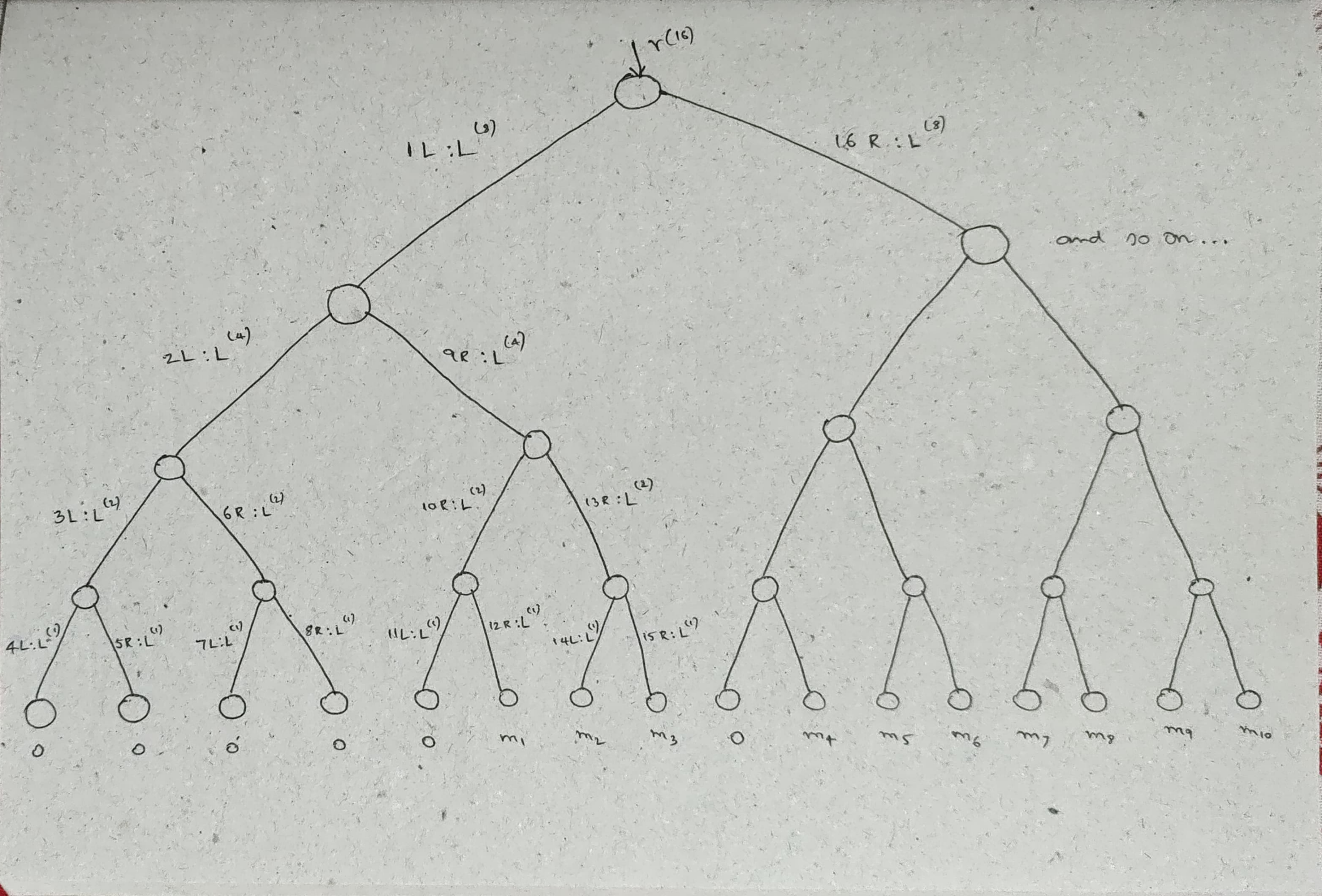
$= 1$, if $L(u_i) < 0$

SC decoder : Sequence of Operations

- Start at root
- At every node, if not leaf, do the following in sequence.
  - Do Step L and go to left child
  - When decision is received from left child, do Step R and go to right child
  - When decision is received from right child, do Step U and go to parent.
- If leaf, make decision and go to parent.

SC decoder Example : Sequence of operations




r(16)
1L : L(8)
16R : L(8)
and so on...
2L : L(4)
9R : L(4)
3L : L(2)
6R : L(2)
10R : L(2)
13R : L(2)
4L : L(1)
5R : L(1)
7L : L(1)
8R : L(1)
11L : L(1)
12R : L(1)
14L : L(1)
15R : L(1)
0
0
0'
0
0
m1
m2
m3
0
m4
m5
m6
m7
m8
m9
m10

Week 8 : Session 3

Decoding of Polar codes - 2 : Successive Cancellation List decoder

In the last section, we talked about how we can decode polar codes using successive cancellation decoder, we constructed a binary tree representation of this polar code and we went towards the left node. Once we reached the left child, we decoded it. And then we went towards the Right node, right child, decoded it and then we moved up. So, this is how we kind of decoded. we were kind of decoding them successively.

Now, the problem with the Successive Cancellation decoder is, if we make an error initially, then that error propagates, because we are making use of knowledge of decision of earlier bits, to decode the next bits.

So, how can we improve the performance of Successive Cancellation decoder? we use Successive Cancellation List decoder:

In Successive Cancellation List decoder, we are not just going to decode or take decision on just one. We are going to keep alive a list of potential candidates. So, at each time instance, we are not only keeping alive one valid path, but we would kind of keep multiple paths alive, and then based on some criteria we'll decide which is the likely path or which is the likely set of code words.

In other words, the estimates that we took in Successive Cancellation is directly $\hat{u}_i = 0$ or $\hat{u}_i = 1$. Now what we are going to say is u_i is 0 with some path metric, u_i is 1 with some path metric.

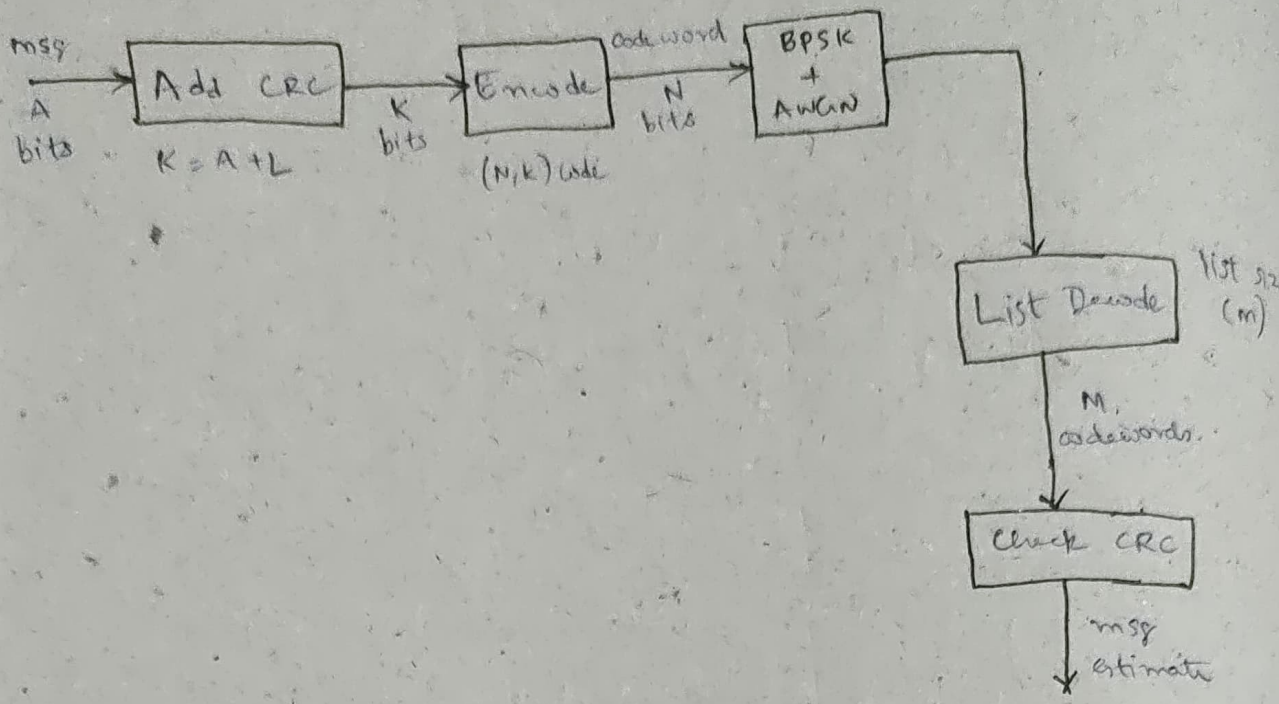
So we are also keeping track of some metric associated with our decision and we are keeping alive this metric. So, at the end of decoding, we will have multiple options available and from we will evaluate them. to see which one is most likely set of sequences. So, we are here maintaining a list. unlike earlier.

The advantage of maintaining list is that the performance improves significantly. The drawback is, at each time instance, instead of having a single decision, you've to maintain this list all the time until we are done with the decoding.

SC list decoding

- SC decoding : Needs improvement in performance
- List decoding : Produce a list of possible codewords
  - Instead of producing a single codeword estimate
  - 4 or 8
- How to choose from list ?
  - Use cyclic Redundancy checks (CRCs)
- Adds complexity but provides vital improvement in performance.
- Good candidate for implementation.

List decoding, decision metric, path metric



⊙ How to produce multiple codewords at decoder?

Polar sc decoder : consider both decisions for each bit; assign Decision Metric (DM).

⊙ If $L_{u_i} \geq 0$, $u_i' = 0$ then $DM_i = 0$.

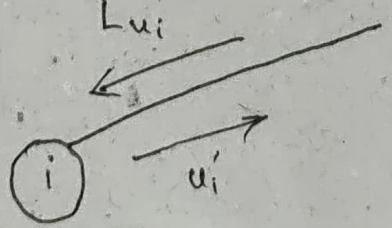
$u_i' = 1$ then $DM_i = |L(u_i)|$

⊙ If $L_{u_i} < 0$, $u_i' = 1$ then $DM_i = 0$

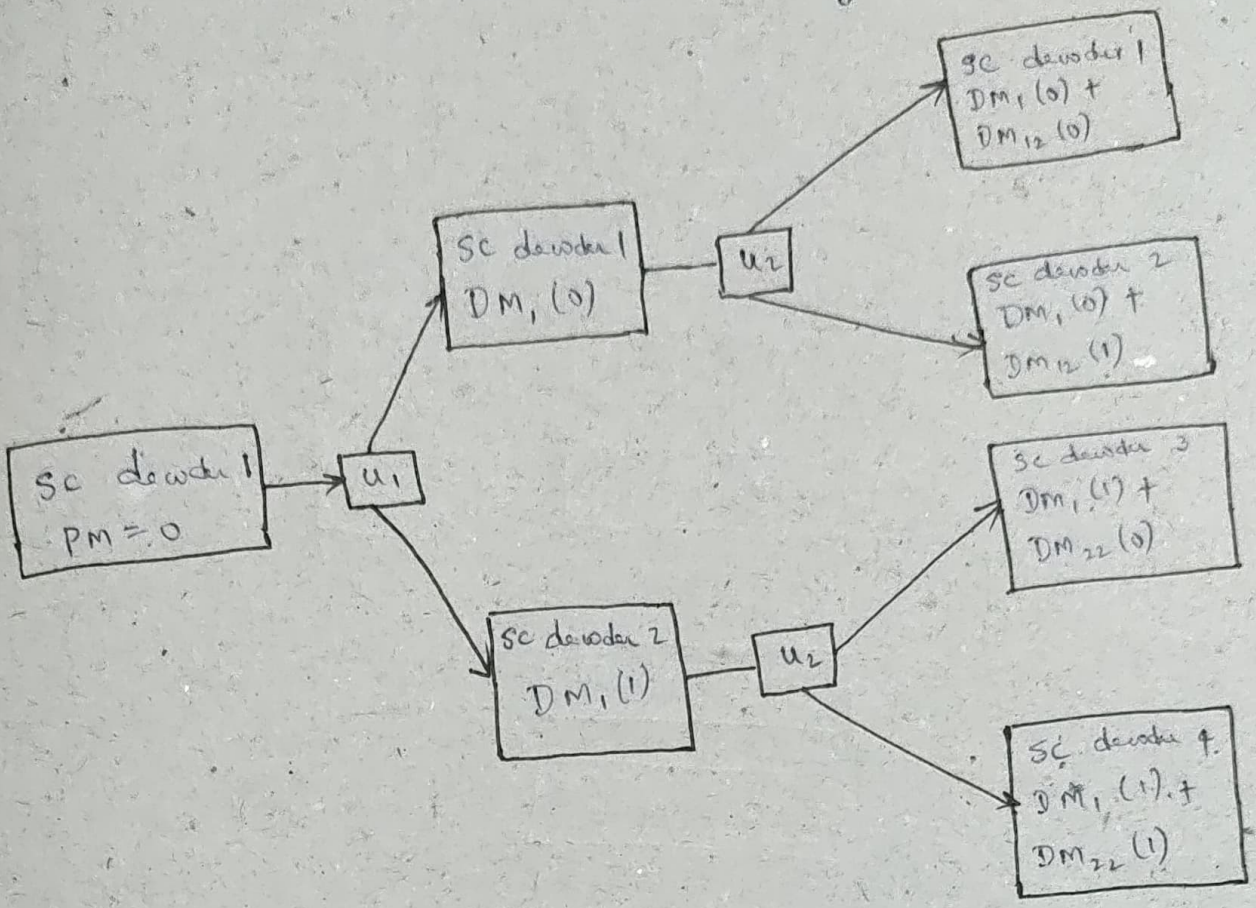
$u_i' = 0$ then $DM_i = |L(u_i)|$

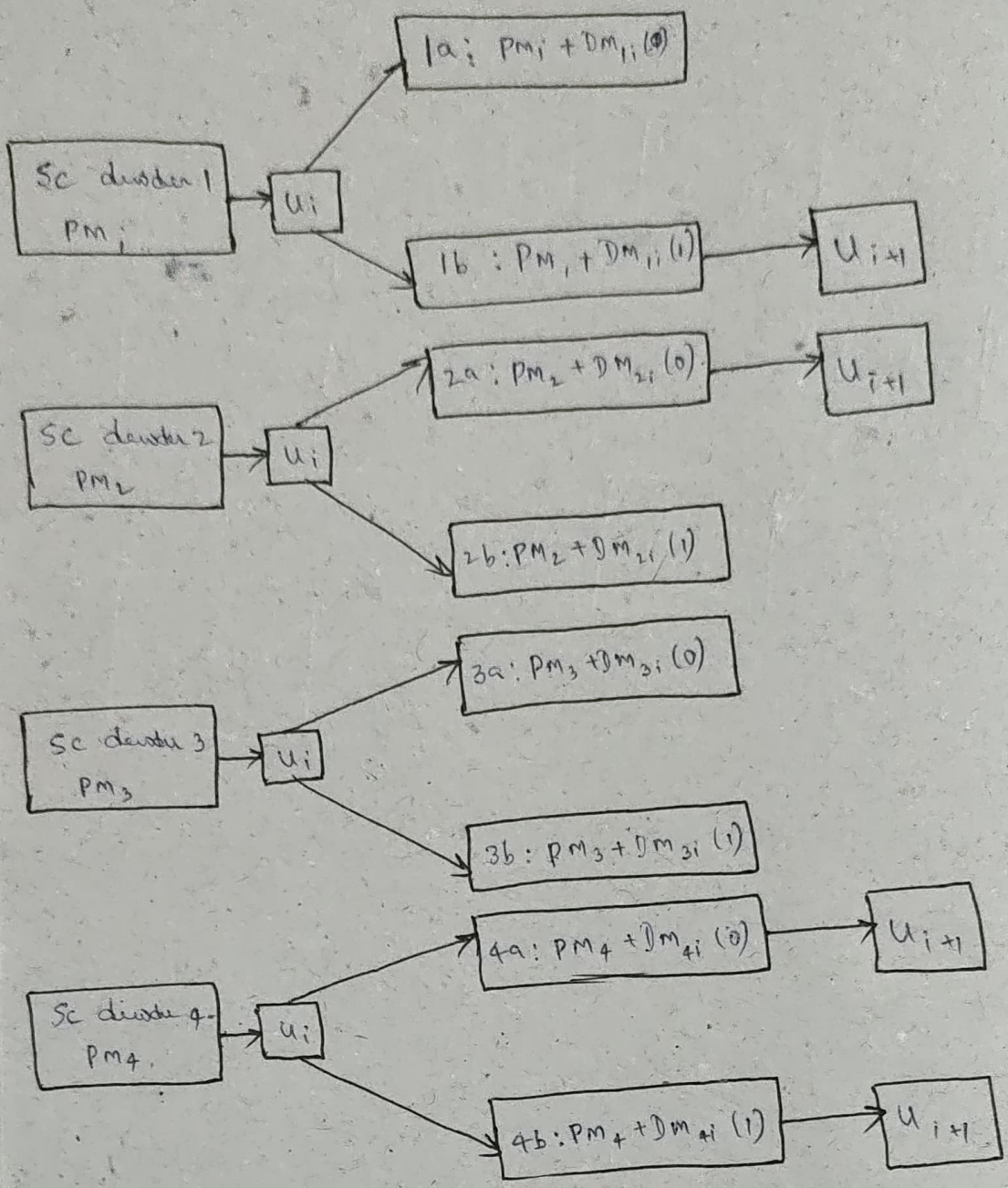
⊙ IMPORTANT : DM assigned even if 'i' is frozen.

⊙ Path Metric : Sum of decision metrics on a path of choice



Successive cancellation. List decoding : Size = 4





- Sort and select best 4
- Continue with 4 surviving decoders
- Decoded codewords on 4 surviving paths at the end.