Computer-Aided VLSI System Design

Final Project: Polar Decoder

lecturer: Yu-Chen Lo

Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE

Overview of Channel Coding

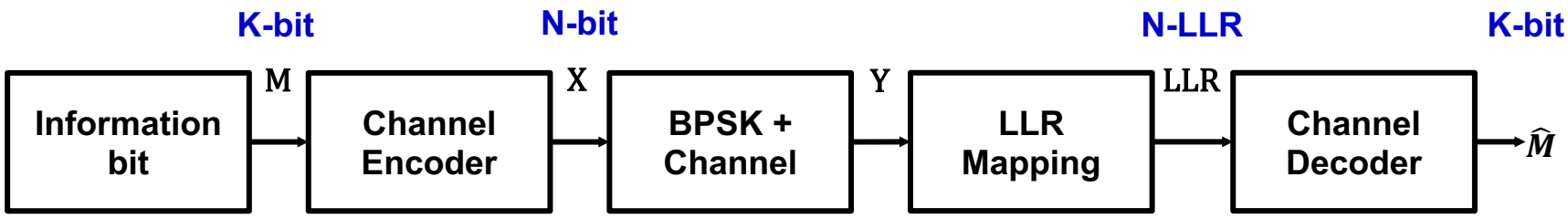
- Consider a simple binary input AWGN (additive white Gaussian noise) communication system
  - It's common to use log likelihood ratio (LLR) to represent received symbol
  - We decide $\widehat{\boldsymbol{M}} = \boldsymbol{0}$ if LLR>0, and vise versa (it's the best decision in this case)

- To overcome the imperfectness of channel
  - Channel coding: add redundancy to raw info
- The rate of a channel code is defined as R = K/N
- In this project, **polar code** is adopted

Noise: n

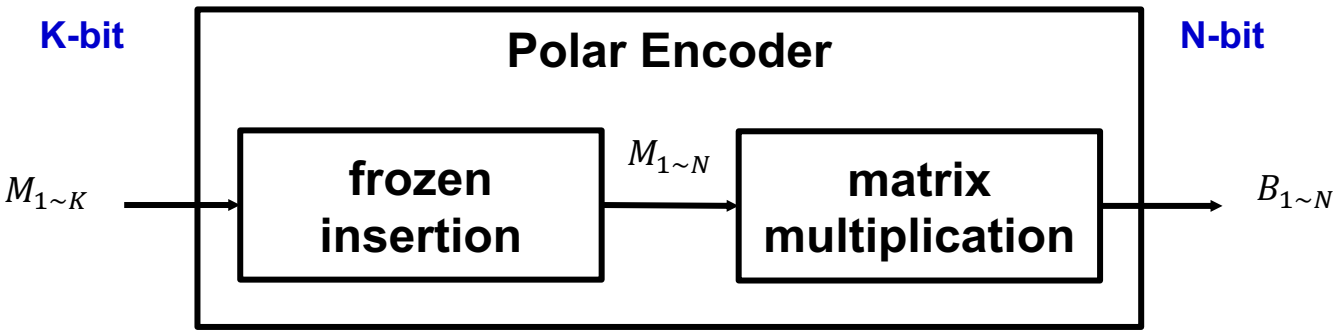
$$y_j = x_j + n_j$$

$$where \quad n_j \sim Gaussian\left(0 \quad , \frac{N_0}{2}\right), \frac{N_0}{2} = 1$$



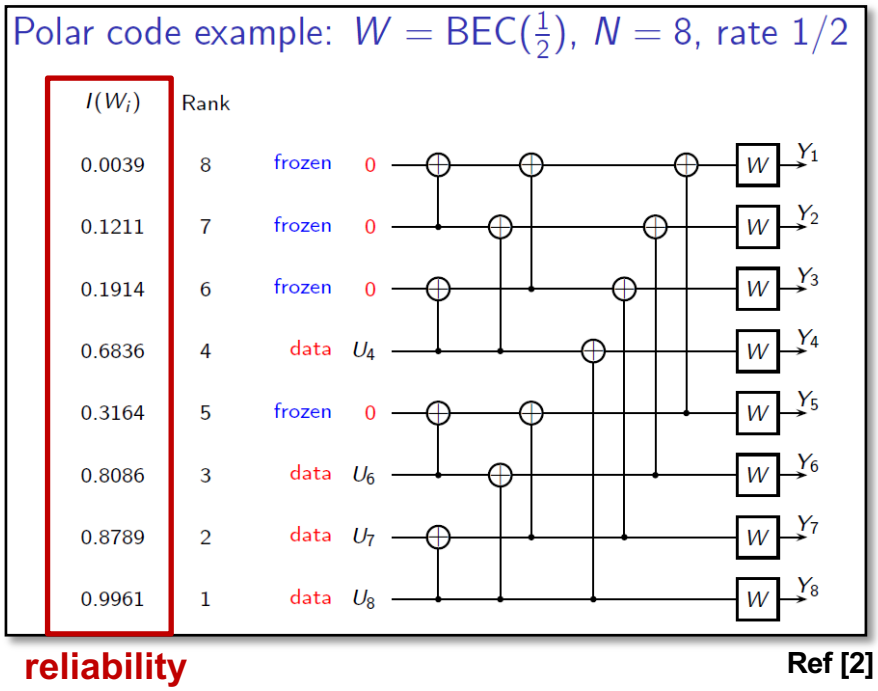
Polar Code Concept

- With N use of the same channel, polar encoder creates N logical channel with different reliability
- High reliability channel has low probability of transmit error
- Not every channel is used because channel coding needs to add redundancy
  - For rate R=K/N, we put the K information bits in the K most reliable channel
  - Transmit 0 (called frozen bits) on the rest N-K channel
- The frozen channel index is known to both transmitter and decoder



Polar Encoder

- The reliability sequence is provided
  - Reliability index of N=128, 256, 512 in the excel
- More reliable channel with larger reliability index



reliability

Ref [2]

	N=64
2	0
3	1
4	2
5	4
6	8
7	16
8	32
9	3
10	5
11	9
12	6
13	17
14	10
15	18
16	12
17	33
18	20
19	34
20	24
21	36
22	7
23	11
24	40
25	19
26	13
27	48
28	14
29	21
30	35
31	26
32	37
33	25
34	22
35	38
36	41
37	28
38	42
39	49
40	44
41	50
42	15

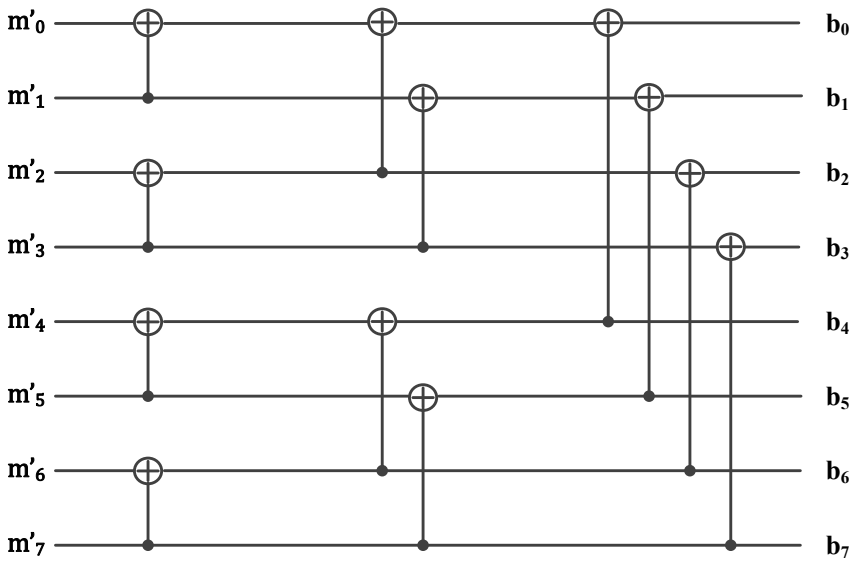
Reliability index

Polar Encoder

- Frozen insertion
  - Put the K info bits in the K most reliable channel (start from lower channel index)
  - $[\boldsymbol{m_0}\ \boldsymbol{m_1}\ \ldots\ \boldsymbol{m_K}] \rightarrow [\boldsymbol{m'_0}\ \boldsymbol{m'_1}\ \ldots\ \boldsymbol{m'_N}]$
- After frozen insertion, multiply with generator matrix $\boldsymbol{G_N}$ (mod 2)
  - $\boldsymbol{G_N} = (\boldsymbol{G_2})^{\otimes \boldsymbol{n}}$ as the n-th Kronecker power of matrix $\boldsymbol{G_2}$ [3]

$$[\boldsymbol{b_0}\quad \boldsymbol{b_1}\quad \cdots\quad \boldsymbol{b_N}] = \overrightarrow{\boldsymbol{m'}}_{1\times N}\boldsymbol{G_N}$$

$$G_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



BPSK Modulation

- BPSK modulation ($\boldsymbol{b_j \rightarrow x_j}$)
  - $\boldsymbol{E_N}$ is the energy to send one bit over channel
  - $\boldsymbol{E_b}$ is the energy per information bit
  - $\boldsymbol{R = K/N}$
  - $\boldsymbol{E_N = RE_b}$
  - Define the "design-SNR" as $\boldsymbol{2RE_b/N_0 = RE_b = E_N}$
  - If you are comparing different rate/algorithm/channel code, you should use $\boldsymbol{E_b/N_0}$ as SNR for fairness (same power for every info bit)
  - For simplicity, we use design-SNR in this project

$$x_j = \begin{cases} -\sqrt{\frac{2\mathrm{RE_b}}{\mathrm{N_0}}} = -\sqrt{RE_b} = -\sqrt{E_N} & , b_j = 0 \\ \sqrt{\frac{2\mathrm{RE_b}}{\mathrm{N_0}}} = \sqrt{RE_b} = \sqrt{E_N} & , b_j = 1 \end{cases} \quad , \forall j$$

Ref [4]

LLR Generation

- LLR generation ($\boldsymbol{y_j \rightarrow llr_j}$)
  - LLR (floating point) is calculated using the following equation [4]

$$LR(y_j) = \frac{f(\mathrm{y}\mid 0)}{f(\mathrm{y}\mid 1)} = \frac{\exp\left(-\frac{1}{N_0}\left(y-\left(-\sqrt{\frac{2RE_b}{N_0}}\right)\right)^2\right)}{\exp\left(-\frac{1}{N_0}\left(y-\left(\sqrt{\frac{2RE_b}{N_0}}\right)\right)^2\right)} = \exp\left(-\frac{1}{N_0}\left(4y\sqrt{\frac{2RE_b}{N_0}}\right)\right) = \exp\left(-2y\sqrt{RE_b}\right)$$

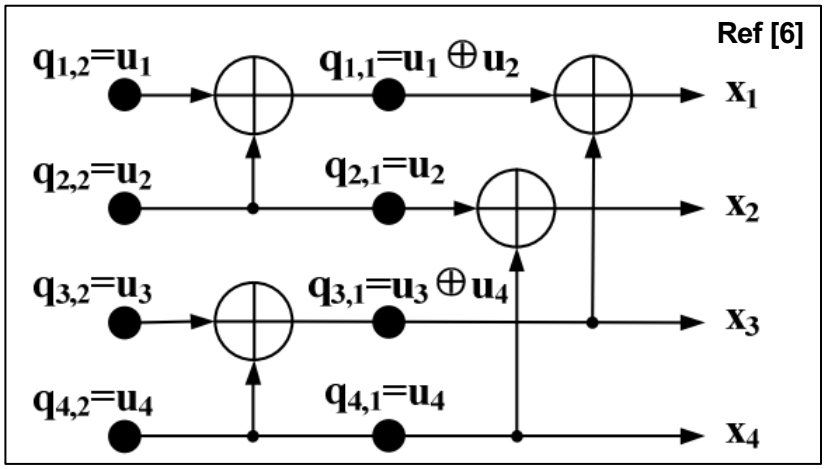
$$\boldsymbol{LLR(y_j) = \ln\left(\exp\left(-2y\sqrt{RE_b}\right)\right) = -2y\sqrt{RE_b} = -2y\sqrt{E_N}}$$

  - Then quantize to two's complement s7.4 (symmetric saturation, biased rounding)

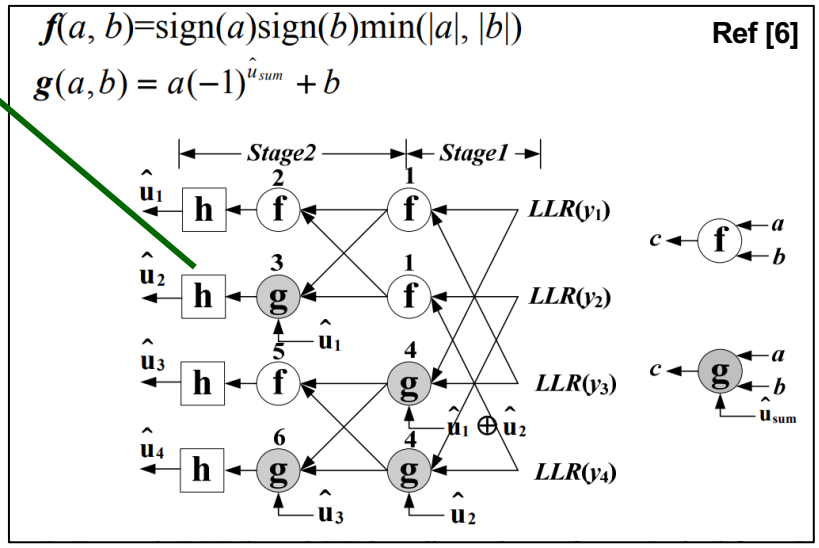
Polar Decoder

- In this final project, you can choose to implement any advanced polar decoding algorithm you find
- You can also just use the basic SC decoder (grading baseline)
  - Refer to [6] section II-A/B for a N=4 example

$h(x)$: Output hard bit $\hat{u}$ = 0 if frozen
= 1 if not frozen and x<0
= 0 if not frozen and x>0



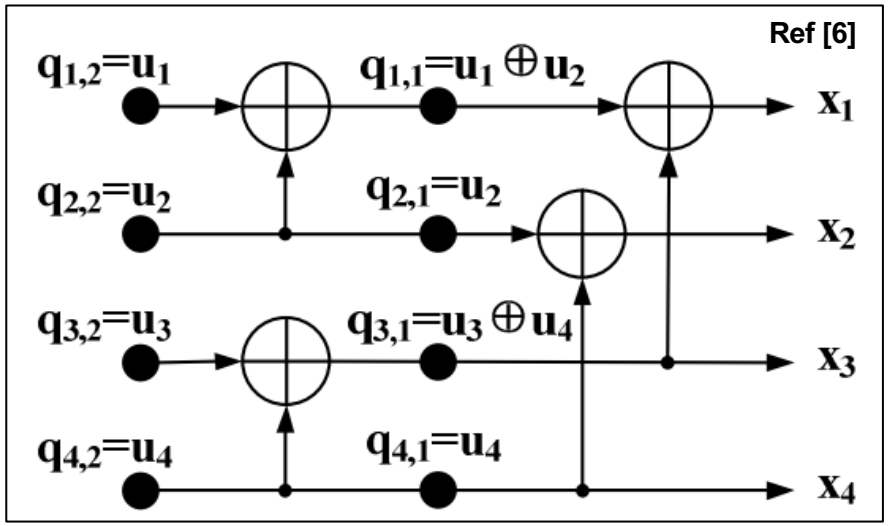
Polar encoder



SC decoder

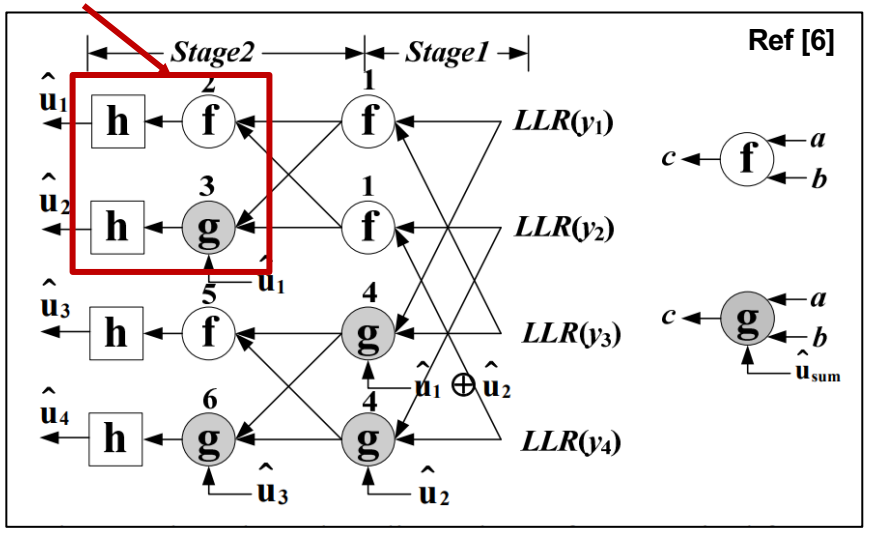
Polar Decoder Example

- Note that **N=4 decoder** is just **2f → N=2 decoder → 2g → N=2 decoder**
  - Similarly, **N=8 decoder** is just **4f → N=4 decoder → 4g → N=4 decoder**



Polar encoder

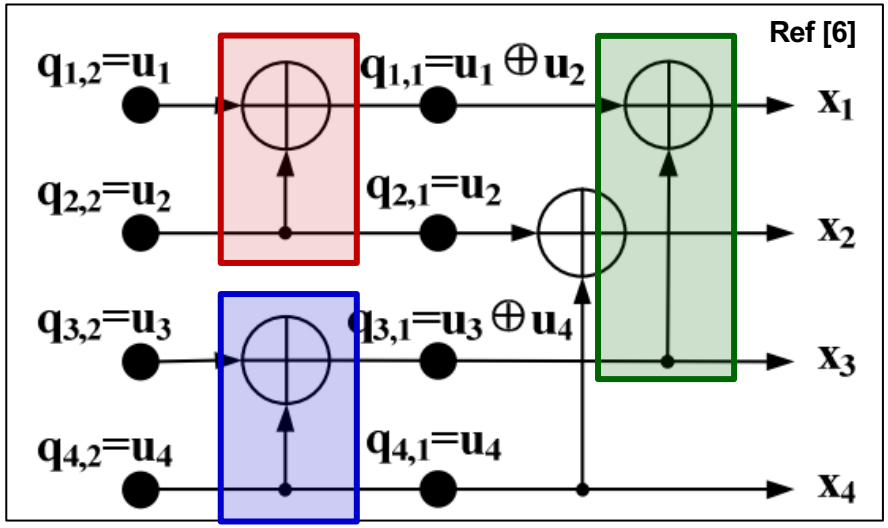
N=2 decoder



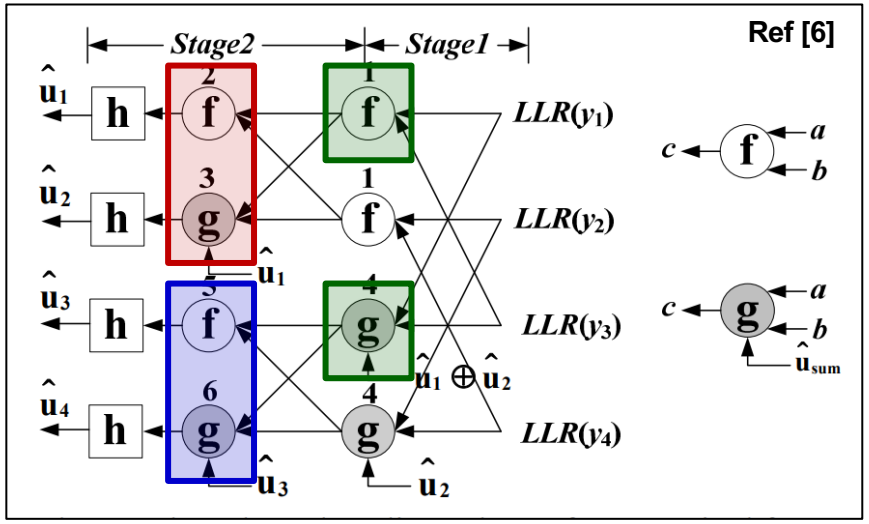
SC decoder

Polar Decoder Example

- The decoding scheme can be constructed referring to the encoder
  - Refer to [6] for more details



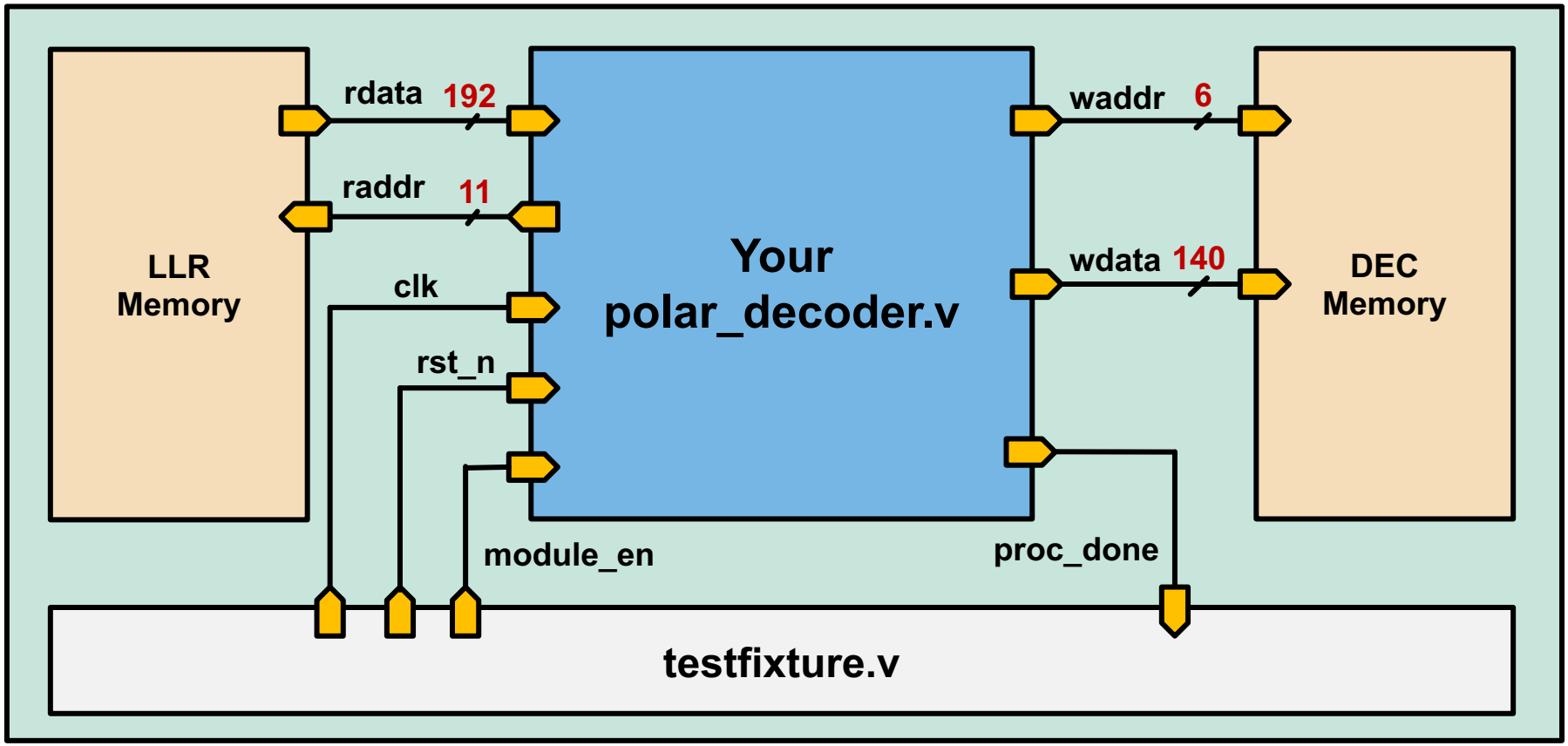
Polar encoder



SC decoder

Block Diagram

- In this final project, you need to implement a polar decoder to decode information bits from log likelihood ratio (LLR)

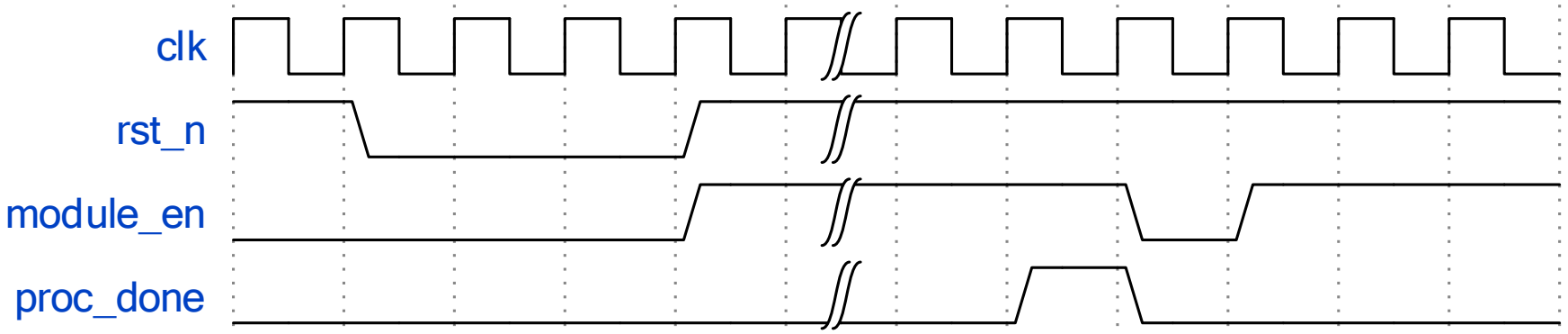


Input/Output

Signal Name	I/O	Width	Simple Description
clk	I	1	Clock signal in the system.
rst_n	I	1	Active **low** asynchronous reset.
module_en	I	1	Active **high** polar decoder enable signal
rdata	I	192	Read data from LLR memory
raddr	O	11	Read address send to LLR memory
wdata	O	140	Write data send to decode memory
waddr	O	6	Write address send to decode memory
proc_done	O	1	Active high signal, indicate the all the packet are decoded

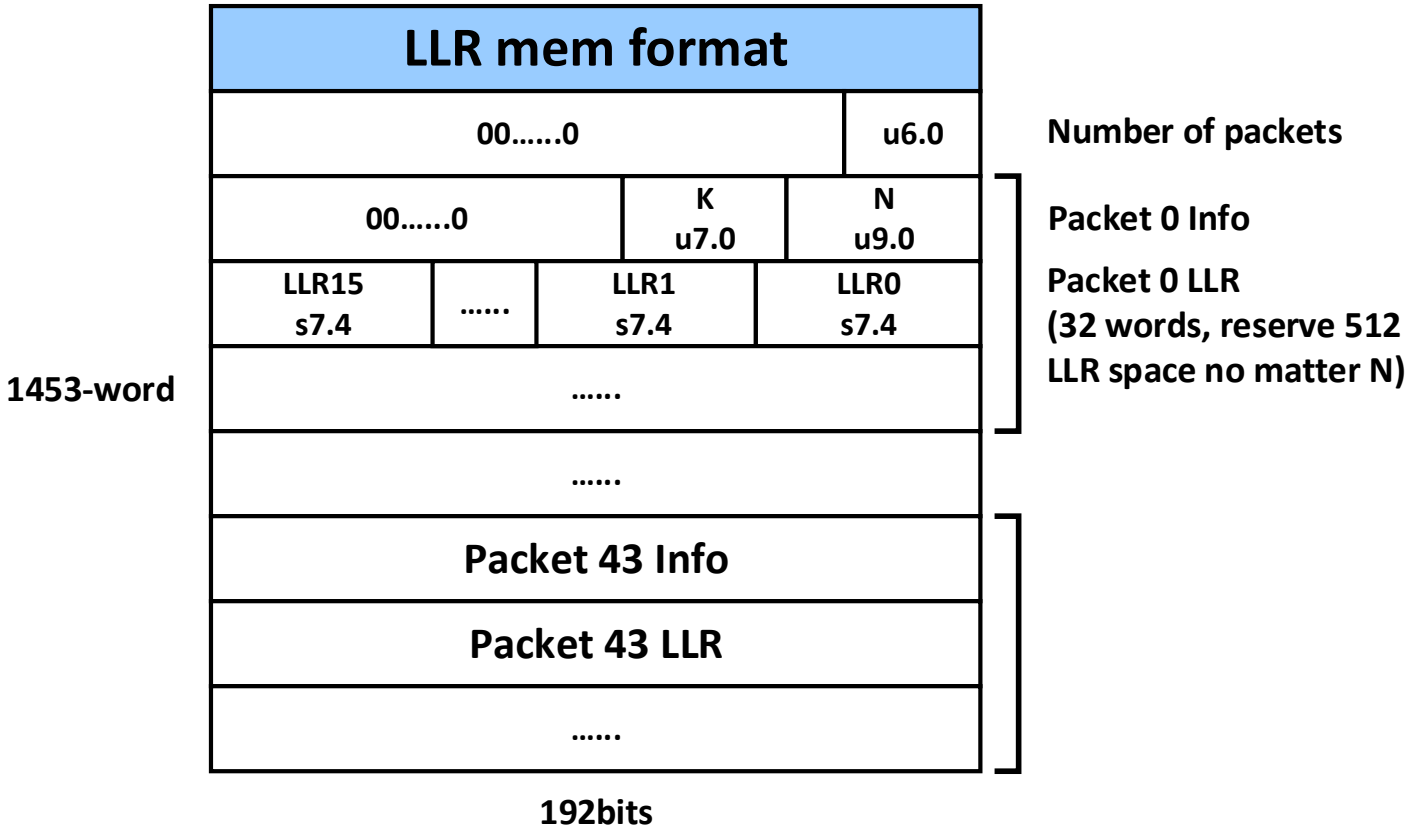
Specifications (1)

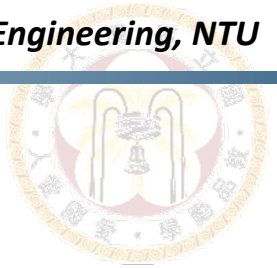
- LLR mem contains 1~44 polar encoded packets, each with different N, K
- LLR mem contents will be ready before **module_en** rise
- After **module_en** rise, DUT start decoding and write the decoded bits of each packet to Decoded mem
- DUT should pull high **proc_done 1T** when all the packets are decoded, and be prepared for next module enable (no reset)
- All the clk-to-input delay are set 1.0ns



LLR Memory Format

- Each LLR: 12bit (1 signed bit, 7 bits integer and 4 bits fractions)
- K: unsigned 8 bits, N: unsigned 10 bits





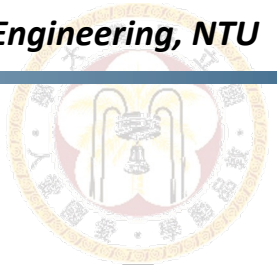
Decoded Memory Format

- Zero extension when decoded K is less than 140

Decoded mem format
Packet 0 decoded bit
Packet 0 decoded bit
(LSB align, pad 0 if K < 140)
……
Packet 43 decoded bit

44-word

140bits

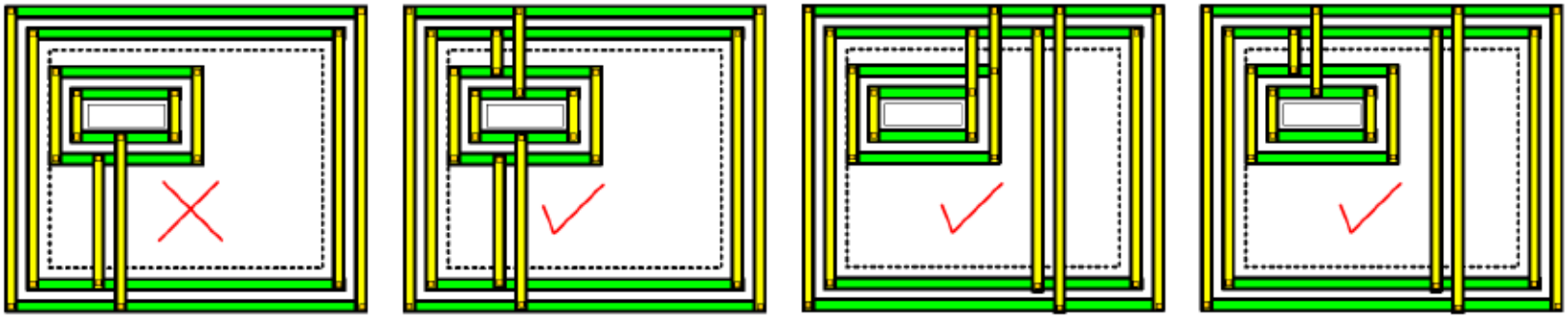


Specifications (2)

- Only worst corners are needed for synthesis and APR.
- The slack for setup-time should be non-negative.
- **No any timing violation and glitches** for the gate level simulation and post-layout simulation.

Specifications (3)

- 只需做 Marco layout 即不用包含 IO Pad 、 Bonding Pad)
- VDD 與 VSS Power Ring 寬度請各設定為 2um 只須做一組
- 不需加 Dummy Metal
- Power Stripe 務必至少加一組 ，其 VDD 、 VSS 寬度各設定為 2um
  - Power Stripe 垂直方向至少一組，水平方向可不加



Specifications (4)

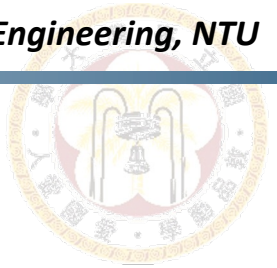
- 務必要加 Power Rail (follow pin)
- Core Filler 務必要加
- APR 後之 GDSII 檔案務必產生
- 完成 APR DRC/LVS 完全無誤
- 記得先產生polar_decoder.ioc，再重新讀取該檔來設定 pin position

Grading Policy

- **Baseline** 50% + **Performance** 40% + **Report** 10%

Item		%	Description
RTL Simulation		20	1. Pass baseline and full pattern simulation 2. If DRC/LVS is not clean, Post-sim is treated as failed
Gate-level Simulation		10	
Post Layout Simulation		20	
Performance	Area, time	20	1. Only baseline pattern is used. 2. You can hand in two version designs
	Power	20	
Report		10	1. Block diagram 2. Optimization technique

Violation	Penalty
G-Sim is pass but Post-Sim is failed	**Performance*0.5**
Only RTL pass	**Performance不評分**
Submitted file name	**總分-3**



Grading Policy (2)

- Baseline fix pattern
  - N(128, 256, 512) × K(12, 35, 60, 97, 110) × SNR(mid, high)
  - Total 30 packets
  - SNR mid: 7dB, SNR high: 8dB

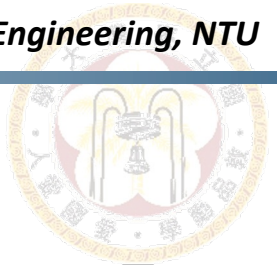
- Full pattern
  - N(128, 256, 512) × K(12~108/140) × SNR(mid, high)
  - Total 710 packets

Submission

- **Due Tuesday, Dec. 20th, 23:59**
- **Require data (with the required directory hierachy):**

teamXX_final.tar	
01_RTL	1. All design verilog files 2. rtl.f
02_SYN	1. Area/timing reports
03_GATE	1. polar_decoder_syn.v/sdf 2. rtl.f
04_APR	1. All design database 2. polar_decoder.gds
05_POST	1. polar_decoder_pr.v/sdf 2. rtl.f
06_POWER	1. try_active.power
reports	1. design.spec 2. teamXX_report.pdf

- **Final project presentation (MTK experience sharing)**
  - **Date: Dec. 27th, 2022**



Reference

[1] http://dde.binghamton.edu/filler/mct/lectures/25/mct-lect25-bawgnc.pdf

[2] https://simons.berkeley.edu/sites/default/files/docs/2689/slidesarikan.pdf

[3] https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214

[4] https://arxiv.org/abs/1501.02473

[5] https://iopscience.iop.org/article/10.1088/1757-899X/768/7/072075/meta

[6] https://arxiv.org/ftp/arxiv/papers/1603/1603.07055.pdf