Computer-Aided VLSI System Design

Final Project:
BCH Decoder

Lecturer: Po-Jen Chen

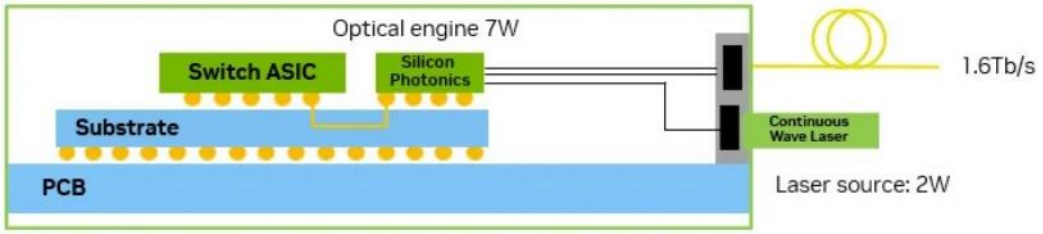
Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE

Introduction

- Advanced AI programs rely on high computing capability and a large network bandwidth
- Over the past two decades [1] M. F. Chen, *IEEE ECTC*, 2025
  - Computing capability: 60,000 times ↑
  - I/O bandwidth: 30 times ↑
- Optical-based interconnects between GPU servers have been regarded as a key to address the I/O bandwidth issue

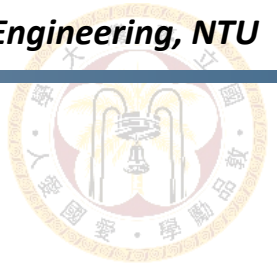
GPU server



Optical interconnect

Project Overview

- What is the importance of the BCH code?
  - Optical communications still need error correction codes
  - The BCH code is a building block of more advanced codes
- Project goal
  - Implement a BCH decoder, which can decode BCH codes of (63, 51), (255, 239), and (1023, 983)
  - The BCH decoder supports hard-decision decoding and soft-decision decoding
- In this project, you will learn:
  - Basic algebra concepts: ring, field, and polynomial
  - BCH codes: encoding and decoding
  - Algebraic decoding algorithm: Berlekamp algorithm
  - Soft-decision decoding algorithm: Chase algorithm



Outline

- Algebra Preliminaries
- BCH Code: Encoding and Decoding
- Project Specifications
- Project Submission
- Grading Policy

Finite Field

- A finite field, or called Galois field (GF), is a field that contains a finite number of elements
- A field is an algebraic structure. It is a set F with addition '+' and multiplication '$\cdot$'
- The following properties hold for all elements $a, b, c \in F$:

  (closure) $a + b \in F, a \cdot b \in F$

  (associativity) $a + (b + c) = (a + b) + c, a \cdot (b \cdot c) = (a \cdot b) \cdot c$

  (identity element) $a + 0 = 0 + a = a, a \cdot 1 = 1 \cdot a = a$

  (inverse) there exists $(-a) \in F$ such that $a + (-a) = (-a) + a = 0$,

  there exists $a^{-1} \in F$ such that $a \cdot a^{-1} = a^{-1} \cdot a = 1$ for $a \neq 0$

  (commutativity) $a + b = b + a, a \cdot b = b \cdot a$

  (distributivity) $a \cdot (b + c) = a \cdot b + a \cdot c$

- Finite fields are widely used in coding theory

$GF(2)$

- A finite field of two elements is F_2 or $GF(2)$
- $GF(2) = \{0, 1\}$ with the following operations
- Addition modulo 2

$$0 + 0 = 0,$$
$$0 + 1 = 1,$$
$$1 + 0 = 1,$$
$$1 + 1 = 0$$

- Multiplication modulo 2

$$0 \cdot 0 = 0,$$
$$0 \cdot 1 = 0,$$
$$1 \cdot 0 = 0,$$
$$1 \cdot 1 = 1$$

Polynomial (1/2)

- Only prime number p can form F_p or $GF(p)$ that has elements $\{0, 1, \ldots, p-1\}$
- To have more practical elements of the set, we construct finite fields by field extension
- We consider the set $F_p[X]$ of all polynomials

$$f(X) = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1},$$

where $a_0, \ldots, a_{n-1} \in F_p$.

- $F_p[X]$ is called the polynomial ring over F_p
- It fulfills the ring properties:
  - For addition: associativity, identity, inverse, commutativity
  - For multiplication: associativity
  - Distributivity

Polynomial (2/2)

- Two polynomials $a(X)$ and $b(X)$ can be divided as
$$a(X) = b(X)q(X) + r(X)$$
with quotient $q(X)$ and remainder $r(X)$
- The degree of $r(X)$ is less than the degree of $b(X)$
- It can also be written as $a(X) \equiv r(X) \bmod b(X)$

- A factor ring $F_p[X]/g(X)$ is the set of remainders calculated from all polynomials in $F_p[X]$ divided by $g(X)$
- This factor ring has other properties:
  - Its multiplication is commutative
  - It has multiplicative identity (unity), i.e., $f(X) = 1$

$GF(2^m)$

- Irreducible polynomial: a polynomial that cannot be factored into the product of two non-constant polynomials
- For the commutative factor ring $F_p[X]/g(X)$ with unity, this ring is a field if and only if $g(X)$ is irreducible

- Now we can construct an extension field $GF(2^m)$ with an irreducible polynomial $p(X) = p_0 + p_1X + \cdots + p_mX^m$ over $GF(2)$
- Such $p(X)$ is called a primitive polynomial
- Any primitive polynomial divides $X^{2^m-1} + 1$
- The factor ring $F_2[X]/p(X)$ is a field called $GF(2^m)$, which is also finite

$GF(2^m)$ Elements

- We define a primitive element α, for $GF(2^m)$ with primitive polynomial $p(X)$
  - Let α be a root of $p(X)$
  - Since we have $X^{2^m-1} + 1 = q(X)p(X)$,
    replacing X by α, $\alpha^{2^m-1} + 1 = q(\alpha)p(\alpha) = q(\alpha) \cdot 0 = 0$
  - So, $\alpha^{2^m-1} = 1$
- $GF(2^m)$ has the elements $\{0,\ 1,\ \alpha,\ \alpha^2,\ \dots,\ \alpha^{2^m-2}\}$
- These elements can also be expressed as polynomials $r_i(\alpha)$
  - Recall the definition of factor ring
  - $\alpha^i \equiv r_i(\alpha) \bmod p(\alpha),\ i = 0, \dots, 2^m - 2$

$GF(2^m)$ Arithmetics

- $GF(2^m)$ elements can be expressed as polynomials over $GF(2)$
- Polynomials over $GF(2)$ can be added, subtracted, multiplied, and divided in the usual way
- The coefficients are modulo-2
- Multiplication example

$$(1 + X + X^2)(1 + X) = 1 + (X + X) + (X^2 + X^2) + X^3 = 1 + X^3$$

- Division example

$$X^3 + X + 1 = (X + 1)(X^2 + X) + 1$$

$$\begin{array}{r|l} & x^2 + x \\ \hline x + 1 & x^3 \quad + x + 1 \\ & x^3 + x^2 \\ \hline & x^2 + x + 1 \\ & x^2 + x \\ \hline & 1 \end{array}$$

Example: $GF(2^4)$ Elements

- $m = 4, p(X) = 1 + X + X^4$
- Calculation examples:

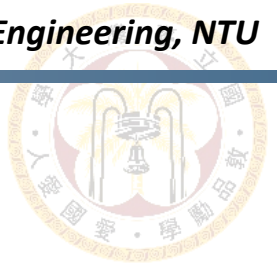
$\alpha^4 \equiv 1 + \alpha \bmod p(\alpha)$

$\alpha^7 = \alpha^6 \cdot \alpha = (\alpha^2 + \alpha^3) \cdot \alpha$
$= \alpha^3 + \alpha^4 \equiv 1 + \alpha + \alpha^3 \bmod p(\alpha)$

You can check $\alpha^{15} = 1$
$\alpha^{15} = \alpha^{14} \cdot \alpha = (1 + \alpha^3) \cdot \alpha$
$= \alpha + \alpha^4 \equiv 1 \bmod p(\alpha)$

Inverse
$\alpha^{-i}\alpha^i = 1 = \alpha^{15}$
$\Rightarrow \alpha^{-i} = \alpha^{15-i}$

Power representation	Polynomial representation	4-tuple representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
α	α	(0 1 0 0)
α^2	α^2	(0 0 1 0)
α^3	α^3	(0 0 0 1)
α^4	$1+\alpha$	(1 1 0 0)
α^5	$\alpha+\alpha^2$	(0 1 1 0)
α^6	$\alpha^2+\alpha^3$	(0 0 1 1)
α^7	$1+\alpha+\alpha^3$	(1 1 0 1)
α^8	$1+\alpha^2$	(1 0 1 0)
α^9	$\alpha+\alpha^3$	(0 1 0 1)
α^{10}	$1+\alpha+\alpha^2$	(1 1 1 0)
α^{11}	$\alpha+\alpha^2+\alpha^3$	(0 1 1 1)
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	(1 1 1 1)
α^{13}	$1+\alpha^2+\alpha^3$	(1 0 1 1)
α^{14}	$1+\alpha^3$	(1 0 0 1)



Outline

- Algebra Preliminaries
- BCH Code: Encoding and Decoding
- Project Specifications
- Project Submission
- Grading Policy

BCH Code

- The Bose, Chaudhuri, and Hocquenghem (BCH) codes form a large class of error-correcting cyclic codes
- Binary BCH codes
  - For positive integers $m \geq 3$ and $t < 2^{m-1}$, there exists a binary BCH code with the following parameters:

Block length	$n = 2^m - 1$
Number of parity-check bits	$n - k \leq mt$
Minimum distance	$d_{min} \geq 2t + 1$

  - It can correct any combination of $\boldsymbol{t}$ **or fewer errors** in a codeword
  - It is denoted by $(\boldsymbol{n},\ \boldsymbol{k})$ **BCH code**

BCH Encoding (1/3)

- The $(n = 2^m - 1,\ k)$ BCH code has codewords in $GF(2^m)$
- Let α be a primitive element of $GF(2^m)$
- Encoding is to generate a codeword polynomial that is a multiple of the generator polynomial $g(X)$
- $g(X)$ of the t-error-correcting BCH code is the lowest-degree polynomial over $GF(2)$ that has $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ as its roots
- The degree of $g(X)$ is $n - k$
- Let $\phi_i(X)$ be the minimal polynomial of α^i, i.e., $\phi_i(\alpha^i) = 0$
- Then, $g(X) = \mathrm{LCM}\{\phi_1(X), \phi_2(X), \ldots, \phi_{2t}(X)\}$, where LCM is the least common multiple

BCH Encoding (2/3)

- Suppose that the message to be encoded is $u = (u_0, u_1, \ldots, u_{k-1})$
- The corresponding message polynomial is

$$u(X) = u_0 + u_1 X + \cdots + u_{k-1} X^{k-1}$$

1. Multiply $u(X)$ by X^{n-k}

$$X^{n-k} u(X) = u_0 X^{n-k} + u_1 X^{n-k+1} + \cdots + u_{k-1} X^{n-1}$$

2. Divide $X^{n-k} u(X)$ by the generator polynomial $g(X)$

$$X^{n-k} u(X) = g(X) a(X) + b(X),$$

where $b(X)$ is the remainder and $\deg\big(b(X)\big) < n - k$

3. Add $b(X)$ and $X^{n-k} u(X)$ to obtain codeword polynomial

$$b(X) + X^{n-k} u(X) = b_0 + b_1 X + \cdots + b_{n-k-1} X^{n-k-1} + u_0 X^{n-k} + u_1 X^{n-k+1} + \cdots + u_{k-1} X^{n-1}$$

- The codeword is $(b_0, b_1, \ldots, b_{n-k-1}, u_0, u_1, \ldots, u_{k-1})$

BCH Encoding (3/3)

- (63, 51) BCH: $m = 6, n = 63, k = 51, t = 2$

  $p(X) = 1 + X + X^6$

  $\phi_1(X) = \phi_2(X) = \phi_4(X) = 1 + X + X^6$

  $\phi_3(X) = 1 + X + X^2 + X^4 + X^6$

  $g(X) = 1 + X^3 + X^4 + X^5 + X^8 + X^{10} + X^{12}$

- (255, 239) BCH: $m = 8, n = 255, k = 239, t = 2$

  $p(X) = 1 + X^2 + X^3 + X^4 + X^8$

  $\phi_1(X) = \phi_2(X) = \phi_4(X) = 1 + X^2 + X^3 + X^4 + X^8$

  $\phi_3(X) = 1 + X + X^2 + X^4 + X^5 + X^6 + X^8$

  $g(X) = 1 + X + X^5 + X^6 + X^8 + X^9 + X^{10} + X^{11} + X^{13} + X^{14} + X^{16}$

- (1023, 983) BCH: $m = 10, n = 1023, k = 983, t = 4$

  $p(X) = 1 + X^3 + X^{10}$

  $\phi_1(X) = \phi_2(X) = \phi_4(X) = \phi_8(X) = 1 + X^3 + X^{10}$

  $\phi_3(X) = \phi_6(X) = 1 + X + X^2 + X^3 + X^{10}$

  $\phi_5(X) = 1 + X^2 + X^3 + X^8 + X^{10}$

  $\phi_7(X) = 1 + X^3 + X^4 + X^5 + X^6 + X^7 + X^8 + X^9 + X^{10}$

  $g(X) = 1 + X + X^3 + X^4 + X^7 + X^9 + X^{10} + X^{11} + X^{12} + X^{16} + X^{19} + X^{21}$
  $+X^{22} + X^{23} + X^{24} + X^{25} + X^{27} + X^{29} + X^{30} + X^{31} + X^{33} + X^{39} + X^{40}$

BCH Decoding (1/8)

- Suppose that a codeword polynomial $c(X) = c_0 + c_1X + \cdots + c_{n-1}X^{n-1}$ is transmitted
- The received polynomial is $r(X) = r_0 + r_1X + \cdots + r_{n-1}X^{n-1}$
- Let $e(X)$ be the error pattern. Then,

$$r(X) = c(X) + e(X)$$

- The algebraic (hard-decision) decoding steps:

1. Compute the syndromes from the received polynomial $r(X)$
2. Determine the error-location polynomial $\sigma(X)$ from the syndromes
3. Determine the error locations by finding the roots of $\sigma(X)$, and correct the errors in $r(X)$

BCH Decoding (2/8)

- **Step 1: Syndrome calculation**, $S = (S_1, S_2, \ldots, S_{2t})$
- For $1 \le i \le 2t$,

$$S_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + \cdots + r_{n-1}\alpha^{(n-1)i}$$

- We can also compute S_i as follows
- Since $r(X) = a_i(X)\phi_i(X) + b_i(X)$,

$$S_i = r(\alpha^i) = b_i(\alpha^i),$$

where $\phi_i(X)$ is the minimal polynomial of α^i

- Error pattern $e(X)$ is not divisible by $\phi_i(X)$
- Therefore, $e(\alpha^i) = S_i$, which is the relationship between the syndrome and the error pattern

BCH Decoding (3/8)

- The syndromes depend on the error pattern $e(X)$ only
- Suppose that $e(X)$ has v errors at locations $X^{j_1}, X^{j_2}, \ldots, X^{j_v}$, that is,

$$e(X) = X^{j_1} + X^{j_2} + \cdots + X^{j_v}$$

- From $e(\alpha^i) = S_i$, we obtain the following set of equations:

$$S_1 = \alpha^{j_1} + \alpha^{j_2} + \cdots + \alpha^{j_v}$$

$$S_2 = (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + \cdots + (\alpha^{j_v})^2$$

$$\vdots$$

$$S_{2t} = (\alpha^{j_1})^{2t} + (\alpha^{j_2})^{2t} + \cdots + (\alpha^{j_v})^{2t}$$

- We need to solve for the unknown $\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_v}$
- These $j_1, j_2, \ldots, j_v$ are called the error location numbers

BCH Decoding (4/8)

- Define the error-location polynomial

$$\sigma(X) = \left(1 + \alpha^{j_1} X\right)\left(1 + \alpha^{j_2} X\right) \ldots \left(1 + \alpha^{j_v} X\right)$$
$$= \sigma_0 + \sigma_1 X + \sigma_2 X^2 + \cdots + \sigma_v X^v$$

- Compare the coefficients of $\sigma(X)$ with previous set of equations
- We find that σ_i's are related to S_i's by the following:

$$S_1 + \sigma_1 = 0$$
$$S_2 + \sigma_1 S_1 + 2\sigma_2 = 0$$
$$S_3 + \sigma_1 S_2 + \sigma_2 S_1 + 3\sigma_3 = 0$$
$$\vdots$$
$$S_v + \sigma_1 S_{v-1} + \cdots + \sigma_{v-1} S_1 + v\sigma_v = 0$$
$$S_{v+1} + \sigma_1 S_v + \cdots + \sigma_{v-1} S_2 + \sigma_v S_1 = 0$$
$$\vdots$$

- The Berlekamp's iterative algorithm is used to calculate all σ_i's

BCH Decoding (5/8)

- **Step 2: Berlekamp's algorithm**
- Initialization

Error-location polynomial of the μth row

μ	$\sigma^{(\mu)}(X)$	d_μ	l_μ	$\mu - l_\mu$
-1	1	1	0	-1
0	1	S_1	0	0

- Loop: Assuming that we have filled out $-1, 0, 1, \ldots, \mu$th rows, we fill out the $(\mu+1)$th row as follows
  - If $d_\mu = 0$, then $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X)$, and $l_{\mu+1} = l_\mu$
  - If $d_\mu \neq 0$, then we find another row $\rho < \mu$ such that $d_\rho \neq 0$ and the number $\rho - l_\rho$ has the largest value. Then,

    $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d_\rho^{-1} X^{(\mu-\rho)} \sigma^{(\rho)}(X)$, and

    $l_{\mu+1} = \max(l_\mu, l_\rho + \mu - \rho)$

BCH Decoding (6/8)

- In either case,

$$d_{\mu+1} = S_{\mu+2} + \sigma_1^{(\mu+1)} S_{\mu+1} + \cdots + \sigma_{l_{\mu+1}}^{(\mu+1)} S_{\mu+2-l_{\mu+1}}$$

where $\sigma_i^{(\mu+1)}$'s are the coefficients of $\sigma^{(\mu+1)}(X)$

- Termination

  After we obtain $\sigma^{(2t)}(X)$, we can terminate the loop

- The error-location polynomial $\sigma(X) = \sigma^{(2t)}(X)$

BCH Decoding (7/8)

- **Step 3: Finding the roots** of $\sigma(X)$

  We find the roots by substituting α^{-j} into $\sigma(X)$

  If $\sigma(\alpha^{-j}) = 0$ for $j = j_i$, then α^{-j_i} is a root
- All the error location numbers $j_1, j_2, \ldots, j_v$ can be found
- So, we obtain the error pattern $e(X) = X^{j_1} + X^{j_2} + \cdots + X^{j_v}$
- If the number of roots found is less than the degree of $\sigma(X)$, the decoding is failed
- If the number of roots found is equal to the degree of $\sigma(X)$, the decoding is successful

BCH Decoding (8/8)

- Finally, the error correction is simply adding $e(X)$

$$\hat{c}(X) = r(X) + e(X)$$
$$= \hat{b}_0 + \hat{b}_1 X + \cdots + \hat{b}_{n-k-1} X^{n-k-1} + \hat{u}_0 X^{n-k} + \cdots + \hat{u}_{k-1} X^{n-1}$$

- The decoded message is the k highest-degree terms of $\hat{c}(X)$

$$\hat{u} = (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{k-1})$$

Soft-Decision Decoding (1/3)

- The soft-decision decoding algorithms utilize channel measurement information to enhance performance
- Chase proposed a soft-decision decoding algorithm, which tests multiple $r(X)$ candidates to find the most probable result
- Chase algorithm is outlined as follows:

1. Determine the p least reliable bits of the received polynomial $r(X)$
2. Generate 2^p test patterns by enumerating all combinations of the p least reliable bits
3. Decode all test patterns by using the hard-decision decoding algorithm
4. Evaluate each successfully decoded result, and choose the most probable result

Soft-Decision Decoding (2/3)

- **Step 1: Finding p least reliable bits**, p = 2 in this project
- Received polynomial $r(X) = r_0 + r_1X + \cdots + r_{n-1}X^{n-1}$
- The log-likelihood ratio (LLR) of each received bit r_i indicates the coefficient of $r(X)$ and the reliability of that bit
  - LLR ≥ 0: the bit tends to be 0, i.e., $r_i = 0$
  - LLR < 0: the bit tends to be 1, i.e., $r_i = 1$
  - When the absolute value of LLR is smaller, it means the bit is less reliable
- We find the bits that have p smallest absolute values of LLRs
- **Step 2: Test pattern generation**
- Each of the p least reliable bits can be 0 or 1
- We generate all 2^p test patterns from $r(X)$

Soft-Decision Decoding (3/3)

- **Step 3: Test pattern decoding**
- We decode all test patterns by using the hard-decision decoding algorithm
- **Step 4: Decoding result evaluation**
- We only consider successfully decoded results
- Let $\hat{c}(X) = \hat{c}_0 + \hat{c}_1 X + \cdots + \hat{c}_{n-1} X^{n-1}$ be a decoded result
- The correlation value is calculated as

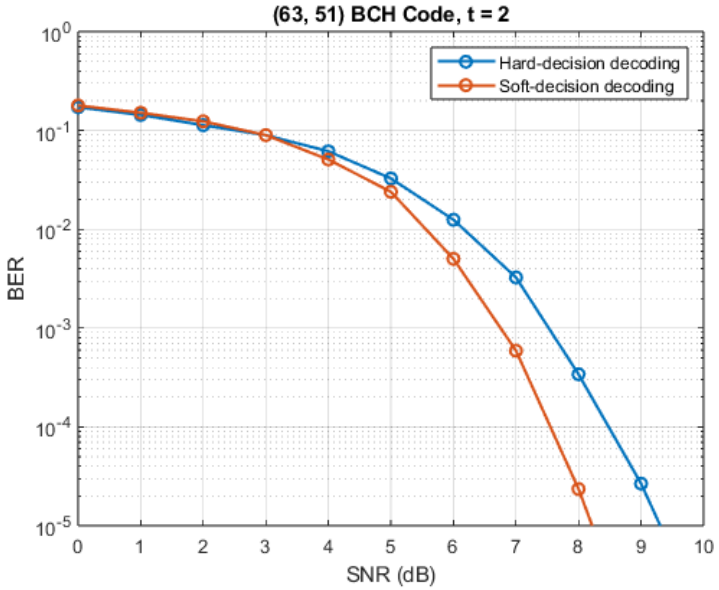
$$\sum_{i=0}^{n-1} l_i (1 - 2\hat{c}_i),$$

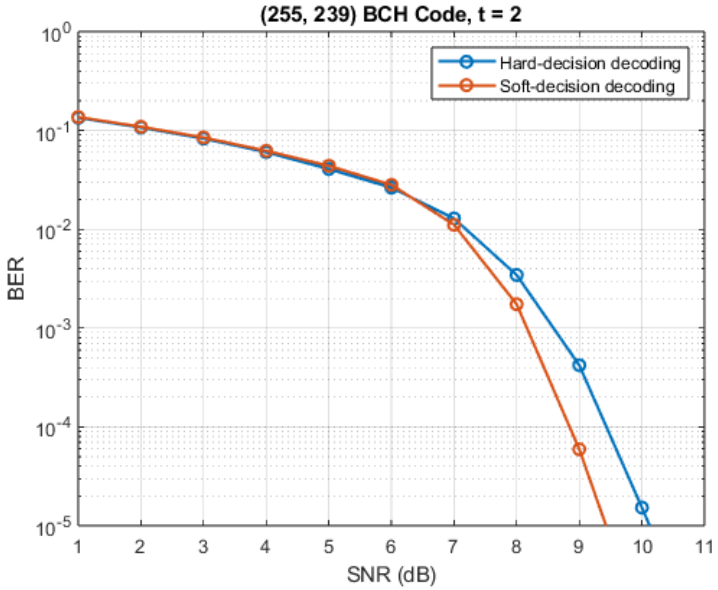
where l_i is the LLR of r_i

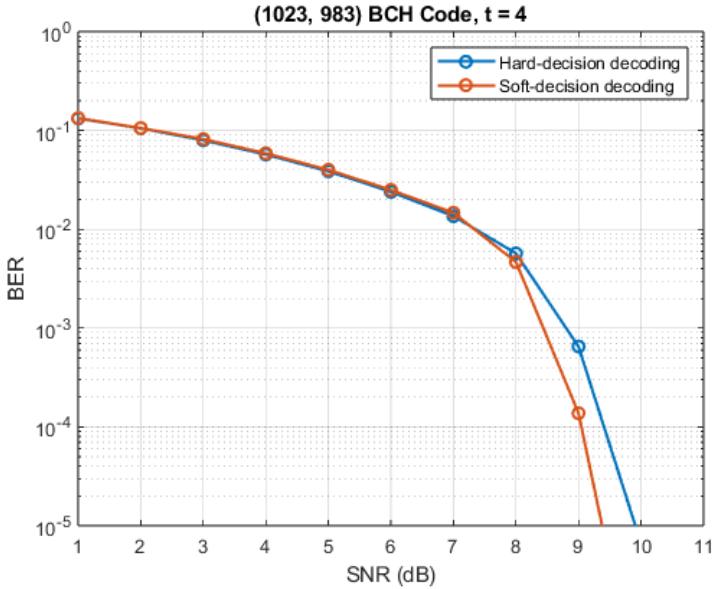
$\hat{c}_i = 0 \Rightarrow 1 - 2\hat{c}_i = 1$
$\hat{c}_i = 1 \Rightarrow 1 - 2\hat{c}_i = -1$

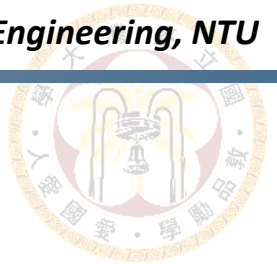
  - Treat the coefficients of $\hat{c}(X)$ as real-valued numbers
- The decoded result of maximal correlation is the most probable result

Bit Error Rate of BCH Codes





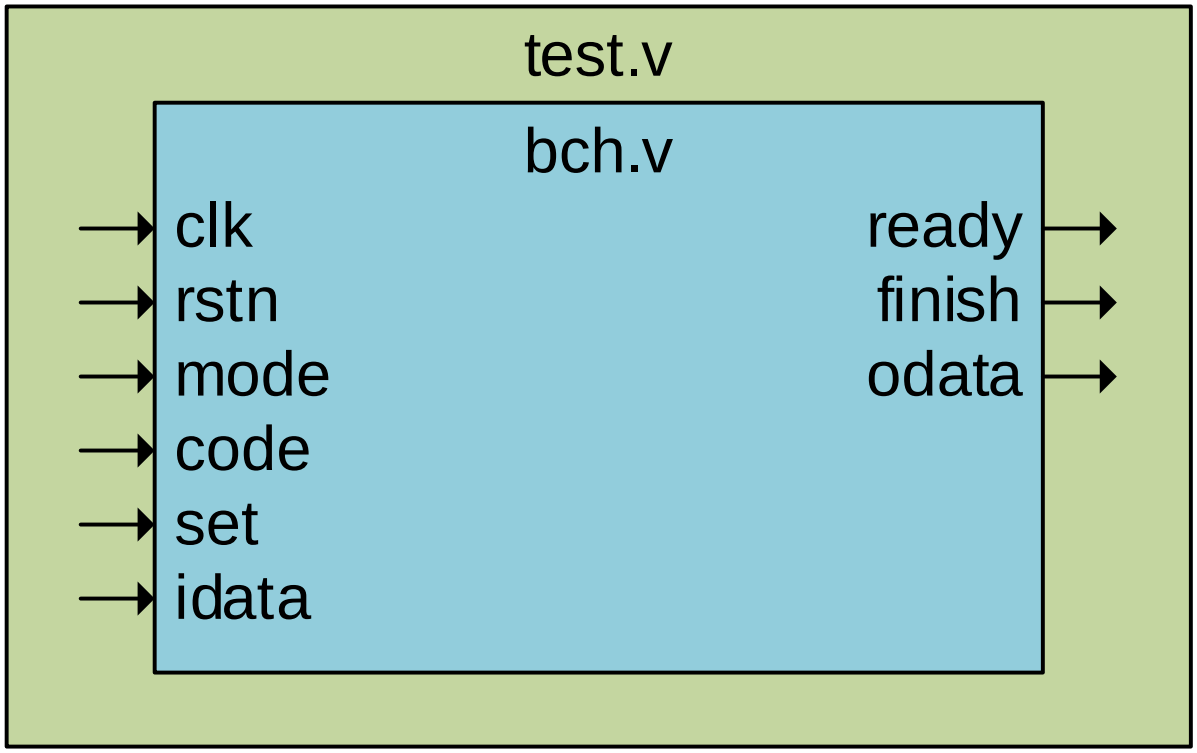


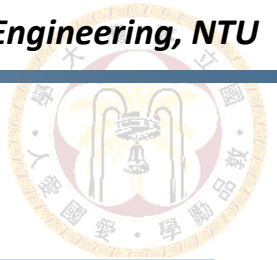


Outline

- Algebra Preliminaries
- BCH Code: Encoding and Decoding
- Project Specifications
- Project Submission
- Grading Policy

Block Diagram

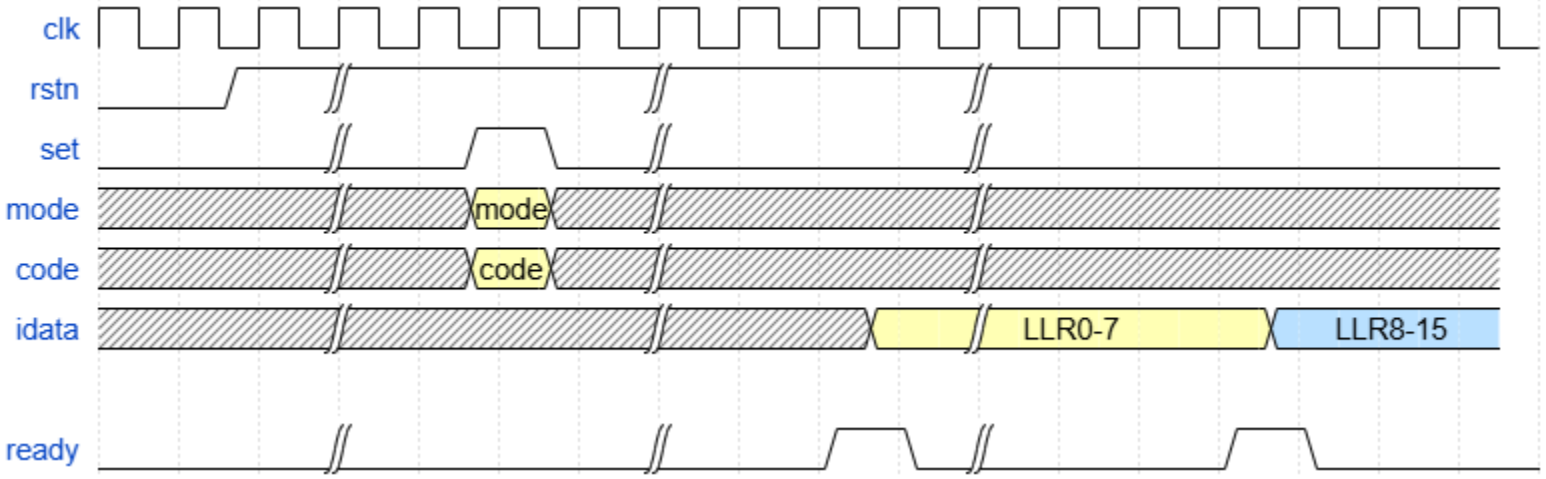




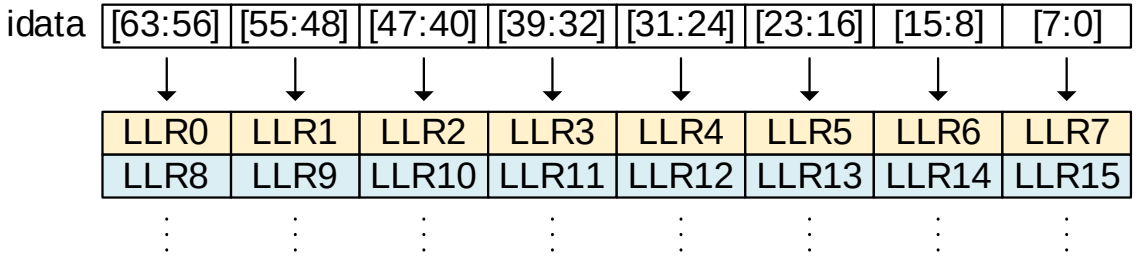
Input/Output

Signal name	I/O	Width	Description
clk	I	1	System clock signal
rstn	I	1	Synchronous active low reset signal
mode	I	1	0: hard-decision decoding 1: soft-decision decoding
code	I	2	1: (63, 51) BCH code 2: (255, 239) BCH code 3: (1023, 983) BCH code
set	I	1	Set the decoding scheme
idata	I	64	Eight LLR signals, each of which is 8 bits
ready	O	1	Indicates the state ready to get input LLR data
finish	O	1	Indicates the end of decoding
odata	O	10	Output error location number

Input Waveform (1/2)



- **mode** and **code** are valid when **set** is HIGH
  - Your design should decode codewords under these settings
- After the design is set, LLR data are given through **idata** port
  - **idata** is valid right after **ready** is HIGH



Input Waveform (2/2)

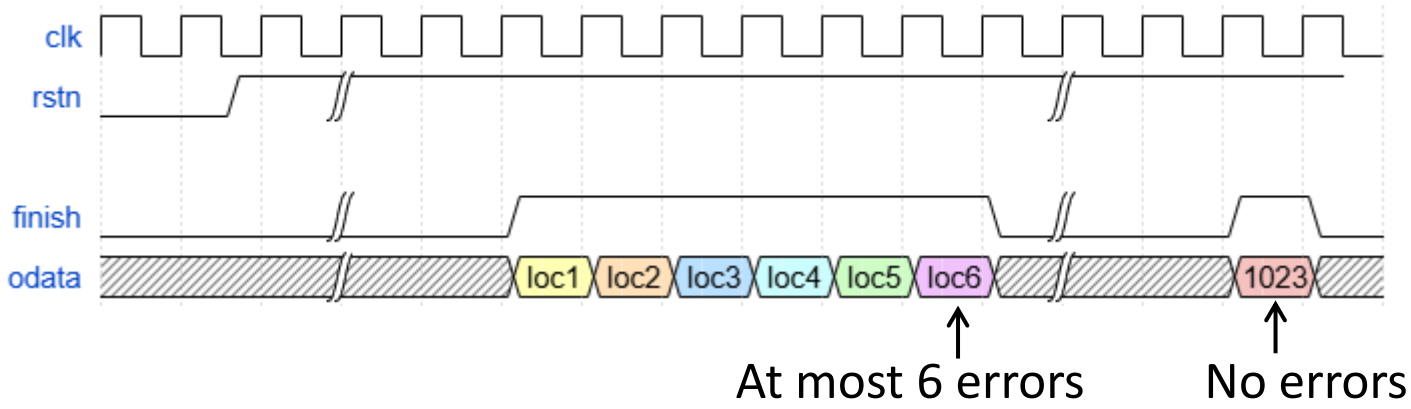
- The testbench streams out LLR data
- LLR data arrangement
  - Each LLR value is 8-bit signed integer
  - The order is in **descending power**:

  LLR0 = don't care,

  LLR1 corresponds to X^{n-1} term of $r(X)$,

  LLR2 corresponds to X^{n-2} term of $r(X)$, …
- The testbench generates all signals at the **negative edge** of the clock

Output Waveform (1/2)



- Your design should continuously output error location numbers through **odata** port, along with **finish** raised to HIGH
  - Ascending error location numbers: loc1 < loc2 < …
  - The number of output cycles is the number of errors
  - If a codeword has no errors, then **odata** = 'd1023 for a cycle
- Note: $e(X) = X^{j_1} + \cdots + X^{j_v}$. Then $j_1, \ldots, j_v$ are the error location numbers

Output Waveform (2/2)

- The testbench will not raise **set** to HIGH until your design completes its output
  - Example sequence:

  Testbench raises **set** → your design raises **finish** → … →

  your design raises **finish** → testbench raises **set** → …
- The testbench checks all signals at the **negative edge** of the clock

Simulation Settings

- RTL simulation: In **01_RTL/**

```
$ bash rtlsim.sh
```

Pattern number

rtlsim.sh

```
vcs -f rtl.f -pvalue+CYCLE=10 -pvalue+PATTERN=100 -full64 -R
-debug_access+all +v2k +notimingcheck -sverilog
vcs -f rtl.f -pvalue+CYCLE=10 -pvalue+PATTERN=200 -full64 -R
-debug_access+all +v2k +notimingcheck -sverilog
vcs -f rtl.f -pvalue+CYCLE=10 -pvalue+PATTERN=300 -full64 -R
-debug_access+all +v2k +notimingcheck -sverilog
```

- Gate-level simulation: In **03_GATE/**

Write your cycle time in **cycle.txt**

```
$ bash gatesim.sh
```

cycle.txt

```
5.0 ns (write decimal point)
```

Clock period

- Post-layout simulation: In **05_POST/**

Write your cycle time in **cycle.txt**

```
$ bash postsim.sh
```

cycle.txt

```
6.0 ns (write decimal point)
```

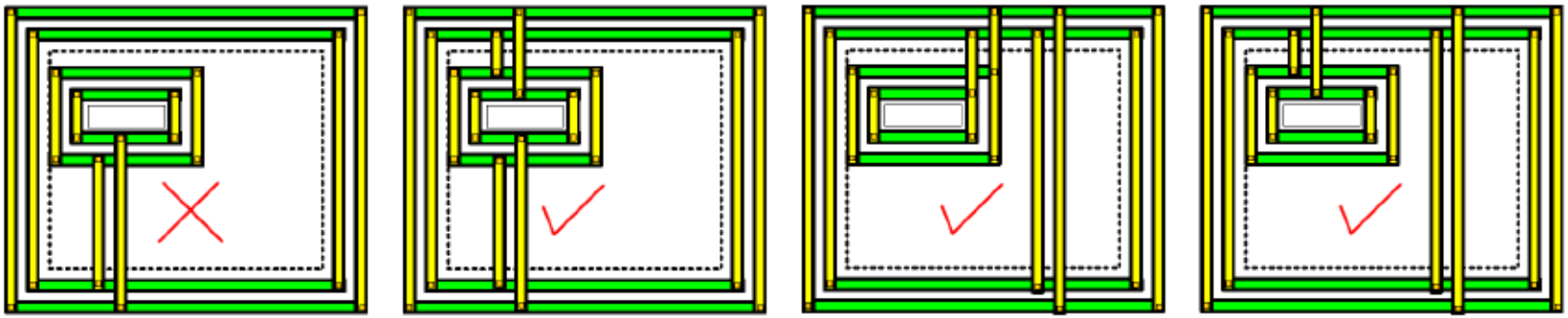
Clock period

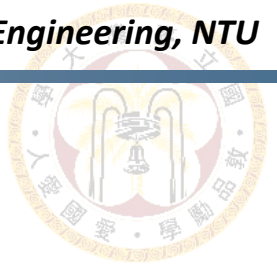
Timing Specifications

- Only the worst-case library is used for synthesis
- The slack for setup time should be nonnegative for synthesis
- The slack for setup time and hold time should be nonnegative for APR
- **No timing violation** for the gate-level simulation and post-layout simulation
  - The timing violation due to reset during the first 10 cycles can be ignored
- Your design should not exceed the maximum cycle of 1000000 for each pattern

APR Specifications (1/2)

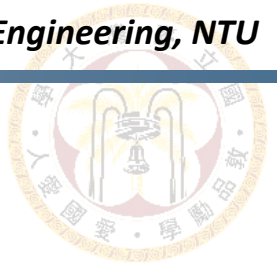
- Only the **macro layout** is needed
  - IO pads and bonding pads are not required
  - The core area is used for grading
- VDD and VSS power rings should each be 2 μm wide, with only one ring required for each
- Power stripes
  - At least one set, with VDD and VSS stripes each 2 μm wide
  - Vertical power stripes require at least one set





APR Specifications (2/2)

- Remember to add the power rail (follow pin)
- Dummy metal layers are not needed
- Core filler must be added
- The GDSII file after APR must be generated
- Ensure that the APR DRC/LVS is completely **error-free**



Outline

- Algebra Preliminaries
- BCH Code: Encoding and Decoding
- Project Specifications
- Project Submission
- Grading Policy

Submission (1/3)

Create a folder named **teamID_final**,

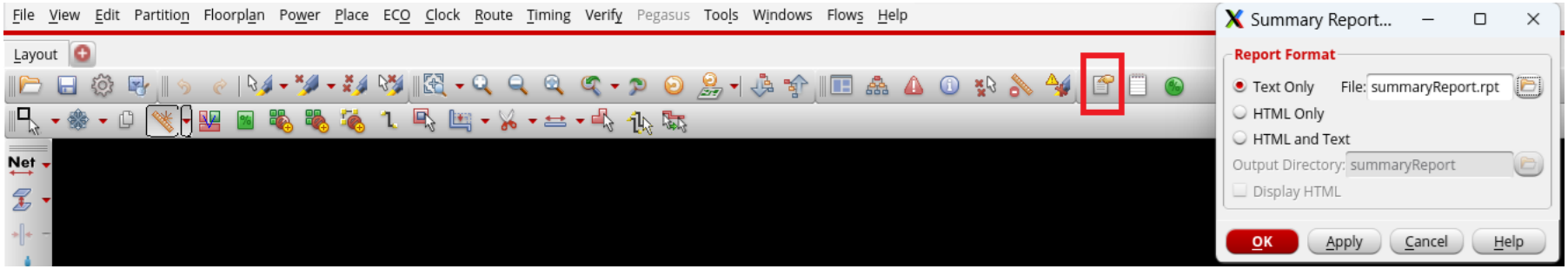
e.g., team001_final

Three digits

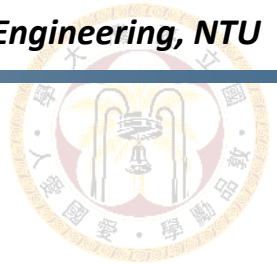
```
team00_final/
├── 01_RTL/
│   ├── bch.v (and other RTL files)
│   └── rtl.f (RTL filelist)
├── 02_SYN/
│   ├── Netlist/
│   │   ├── bch_syn.v
│   │   └── bch_syn.sdf
│   └── Report/
│       ├── bch_syn.area (area report)
│       └── bch_syn.timing_max (setup time report)
├── 03_GATE/
│   └── cycle.txt (your clock period)
├── 04_APR/
│   ├── DBS/
│   │   ├── final (saved APR design)
│   │   └── final.dat (APR database)
│   ├── Netlist/
│   │   ├── bch_apr.v
│   │   └── bch_apr.sdf
│   ├── Report/
│   │   ├── summaryReport.rpt (area report)
│   │   ├── bch_postRoute.summary (setup time report)
│   │   ├── bch_postRoute_hold.summary (hold time report)
│   │   ├── bch.geom.rpt (DRC report)
│   │   ├── bch.conn.rpt (LVS report)
│   │   └── bch.antenna.rpt (DRC report)
│   └── bch.gds
├── 05_POST/
│   └── cycle.txt (your clock period)
└── team00_report.pdf
```

Submission (2/3)

- APR area report



- APR timing reports can be found in **04_APR/timingReports/**
- Other reports can be found in **04_APR/**



Submission (3/3)

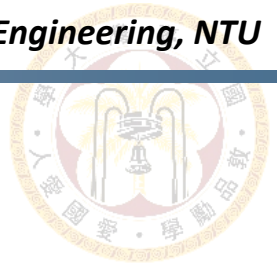
- **Deadline: 2025/12/16 13:59:59** (UTC+8)

- Compress the folder **teamID_final** (all lowercase) in a tar file named **teamID_final_vk.tar** (k: version number, e.g., 1, 2, …)
- Ensure all required files are submitted
- Submit to NTU Cool

Report Requirements

- Format: written in English, single column
- Contents:

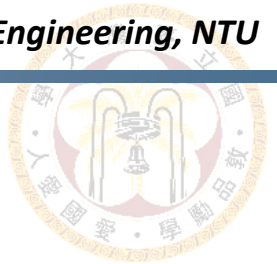
1. Algorithm design
   - Describe the analysis and improvements you have done
2. Hardware implementation
   - Provide the hardware architecture, key module implementation, and hardware optimization techniques
3. APR results
   - Show the layout photo
4. Performance evaluation
   - Assess the efficiency of your design, including metrics such as speed, power, and area



Discussion

- NTU Cool Discussion Forum
  - For any questions not related to assignment answers or privacy concerns, please use the NTU Cool discussion forum
  - TAs will prioritize answering questions on the NTU Cool discussion forum

- Email: d13943013@ntu.edu.tw
  - Title should start with [CVSD 2025 Fall Final Project]
  - Email with wrong title will be moved to trash automatically



Outline

- Algebra Preliminaries
- BCH Code: Encoding and Decoding
- Project Specifications
- Project Submission
- Grading Policy

Grading Policy Overview

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

Item	Percentage	Description
RTL	10	RTL simulation using open patterns
Synthesis	10	Gate-level simulation using open patterns
APR	10	Post-layout simulation using open patterns
APR	20	Post-layout simulation using hidden patterns

- Performance

Item	Percentage	Description
Latency per bit	30	Post-layout simulation using hidden patterns Area × Time / n (excluding data input time)
Energy per bit	10	Post-layout simulation using hidden patterns Power × Time / n

Grading Policy Details (1/3)

- The points of each item can be partially given
  - It is according to the number of patterns you pass
- Open patterns
  - Three patterns (n=63, 255, 1023) for hard-decision decoding
  - Each pattern has two codewords
- Hidden patterns
  - Mixed combinations of {n=63, 255, 1023} x {hard-decision, soft-decision}
  - Each pattern has **one** codeword for the simplicity
- A test pattern is regarded as passed if **all codewords** in that pattern are correctly decoded

Grading Policy Details (2/3)

- Performance ranking
  - Sorting order:

(1) Project completeness (no violation of specifications),
APR > Synthesis

(2) The number of passed patterns

(3) Performance metric A × T or P × T (sum of all patterns)

  - Example

Student A: APR done, pass 10 patterns, A × T = 10000

Student B: APR done, pass 8 patterns, A × T = 6000

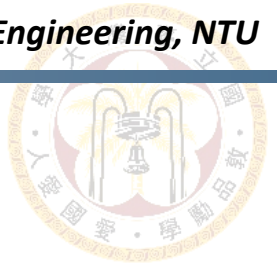
Student C: APR done, pass 8 patterns, A × T = 8000

Student D: Synthesis done, pass 10 patterns, A × T = 8000

Ranking: #1 is A, #2 is B, #3 is C, and #4 is D

Grading Policy Details (3/3)

- **Late submissions are not accepted**
  - Any submission after the deadline will receive 0 points
- File corrections after the deadline should be avoided
  - Corrections for the folder name, file name, file hierarchy, missing file cause 5-point deduction
- The TA will grade your submissions by using scripts
- **No plagiarism**
  - Plagiarism in any form, including copying from online sources, is strictly prohibited



Final Project Presentation

- Date: December 23
- Time: 14:20 – 17:20
- Online

- Top-performing teams will be invited to present their design optimization strategies
- Bonus points will be awarded for presentations

References

- Algebra textbook

John B. Fraleigh, "*A First Course in Abstract Algebra*," Pearson, 2013.

- Coding theory textbook

Shu Lin and Daniel J. Costello, "*Error Control Coding: Fundamentals and Applications*," Pearson, 2004.

- Soft-decision decoding

D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inf. Theory*, vol. 18, no. 1, pp. 170–182, Jan. 1972.