Chương 3 :
KỸ THUẬT
THÔNG TIN DỮ LIỆU

Chương 3 :
Kỹ thuật thông tin dữ liệu

- Mục đích chính của chương:
  - Các kỹ thuật và mạch điện ứng dụng để truyền các khung dữ liệu giữa 2 DTE.
  - Các hệ thống mã phát hiện lỗi mà cho phép DTE bên nhận xác định sự có mặt của bất kỳ lỗi xảy ra chuỗi bit nhận được.

Chương 3 :
Kỹ thuật thông tin dữ liệu

- Nội dung :
  - Hệ thống mã.
  - Cấu hình kết nối cơ bản.
  - Các kiểu thông tin.
  - Các kiểu truyền.
  - Truyền bất đồng bộ.
  - Truyền đồng bộ.
  - Nhiễu Gauss và tỷ lệ lỗi bit.
  - Xử lý dữ liệu số.

3.1 Hệ thống mã (coding schemes)

- Có hai hệ thống mã thường được sử dụng nhất trong hệ thống turyền số liệu :
  - Mã EBCDIC (Extended Binary Coded Decimal Interchange Code) : là bộ mã 8bit được sử dụng trong các thiết bị do hãng IBM sản xuất.
  - Mã ASCII (American Standards Committee for Information Interchange) : là bộ mã 7bit do CITT định nghĩa.

Hệ thống mã (coding schemes)

Bit 4567 / 0123	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	SOH	STX	ETX	PF	HT	LC	DEL				VT	FF	CR	SO	SI
0001	DLE	DC1	DC2	DC3	RES	NL	BS	IL	CAN	EM			IFS	IGS	IRS	IUS
0010			FS		BYP	LF	EOB	PRE			SM			ENQ	ACK	BEL
0011			SYN		PN	RS	UC	EOT					DC4	NAK		SUB
0100	SP										¢	.	<	(	+	|
0101	&										!	$	*	)	;	¬
0110	-	/									¦	,	%	-	>	?
0111										\	:	#	@	'	=	“
1000		a	b	c	d	e	f	g	h	i						
1001		j	k	l	m	n	o	p	q	r						
1010		~	s	t	u	v	w	x	y	z						
1011																
1100	{	A	B	C	D	E	F	G	H	I						
1101	}	J	K	L	M	N	O	P	Q	R						
1110			S	T	U	V	W	X	Y	Z						
1111	0	1	2	3	4	5	6	7	8	9						□

Note: To read this chart, simply find the character on the chart, then look to the left side of the row for bits 0, 1, 2, and 3, and to the top of the column for bits 4, 5, 6, and 7. This is only one of many possible implementations of EBCDIC.

EBCDIC special characters

ACK	Acknowledgement	EOT	End of Transmission	PF	Punch Off
BEL	Bell	ETX	End of Text	PN	Punch On
BS	Backspace	FF	Form Feed	PRE	Prefix
BYP	Bypass	FS	File Separator	RES	Restore
CAN	Cancel	HT	Horizontal Tab	RS	Reader Stop
CR	Carriage Return	IFS	Information File Separator	SI	Shift In
DC1	Device Control 1	IGS	Information Group Separator	SM	Start Message
DC2	Device Control 2	IL	Idle	SO	Shift Out
DC3	Device Control 3	IRS	Information Record Separator	SOH	Start of Heading
DC4	Device Control 4	IUS	Information Unit Separator	SP	Space
DEL	Delete	LC	Lower Case	STX	Start of Text
DLE	Data Link Escape	LF	Line Feed	SUB	Substitute
EM	End of Medium	NAK	Negative Acknowledgement	SYN	Synchronous Idle
ENQ	Enquiry	NL	New Line	UC	Upper Case
EOB	End of Block	NUL	Null	VT	Vertical Tab

Hệ thống mã (coding schemes)

Bits 7654321	Character	Bits 7654321	Character	Bits 7654321	Character	Bits 7654321	Character
0000000	NUL	0100000	SP	1000000	@	1100000	`
0000001	SOH	0100001	!	1000001	A	1100001	a
0000010	STX	0100010	“	1000010	B	1100010	b
0000011	ETX	0100011	#	1000011	C	1100011	c
0000100	EOT	0100100	$	1000100	D	1100100	d
0000101	ENQ	0100101	%	1000101	E	1100101	e
0000110	ACK	0100110	&	1000110	F	1100110	f
0000111	BEL	0100111	'	1000111	G	1100111	g
0001000	BS	0101000	(	1001000	H	1101000	h
0001001	HT	0101001	)	1001001	I	1101001	i
00010l0	LF	0101010	*	1001010	J	1101010	j
0001011	VT	0101011	+	1001011	K	1101011	k
0001100	FF	0101100	,	1001100	L	1101100	l
0001101	CR	0101101	-	1001101	M	1101101	m
0001110	SO	0101110	.	1001110	N	1101110	n
0001111	SI	0101111	/	1001111	O	1101111	o
0010000	DLE	0110000	0	1010000	P	1110000	p
0010001	DC1	0110001	1	1010001	Q	1110001	q
0010010	DC2	0110010	2	1010010	R	1110010	r
0010011	DC3	0110011	3	1010011	S	1110011	s
0010100	DC4	0110100	4	1010100	T	1110100	t
0010101	NAK	0110101	5	1010101	U	1110101	u
0010110	SYN	0110110	6	1010110	V	1110110	v
0010111	ETB	0110111	7	1010111	W	1110111	w
0011000	CAN	0111000	8	1011000	X	1111000	x
0011001	EM	0111001	9	1011001	Y	1111001	y
0011010	SUB	0111010	:	1011010	Z	1111010	z
0011011	ESC	0111011	;	1011011	[	1111011	{
0011100	FS	0111100	<	1011100	\	1111100	|
0011101	GS	0111101	=	1011101	]	1111101	}
0011110	RS	0111110	>	1011110	^	1111110	~
0011111	US	0111111	?	1011111	—	1111111	DEL

Hệ thống mã (coding schemes)

Những ký tự không in được trong mã ASCII

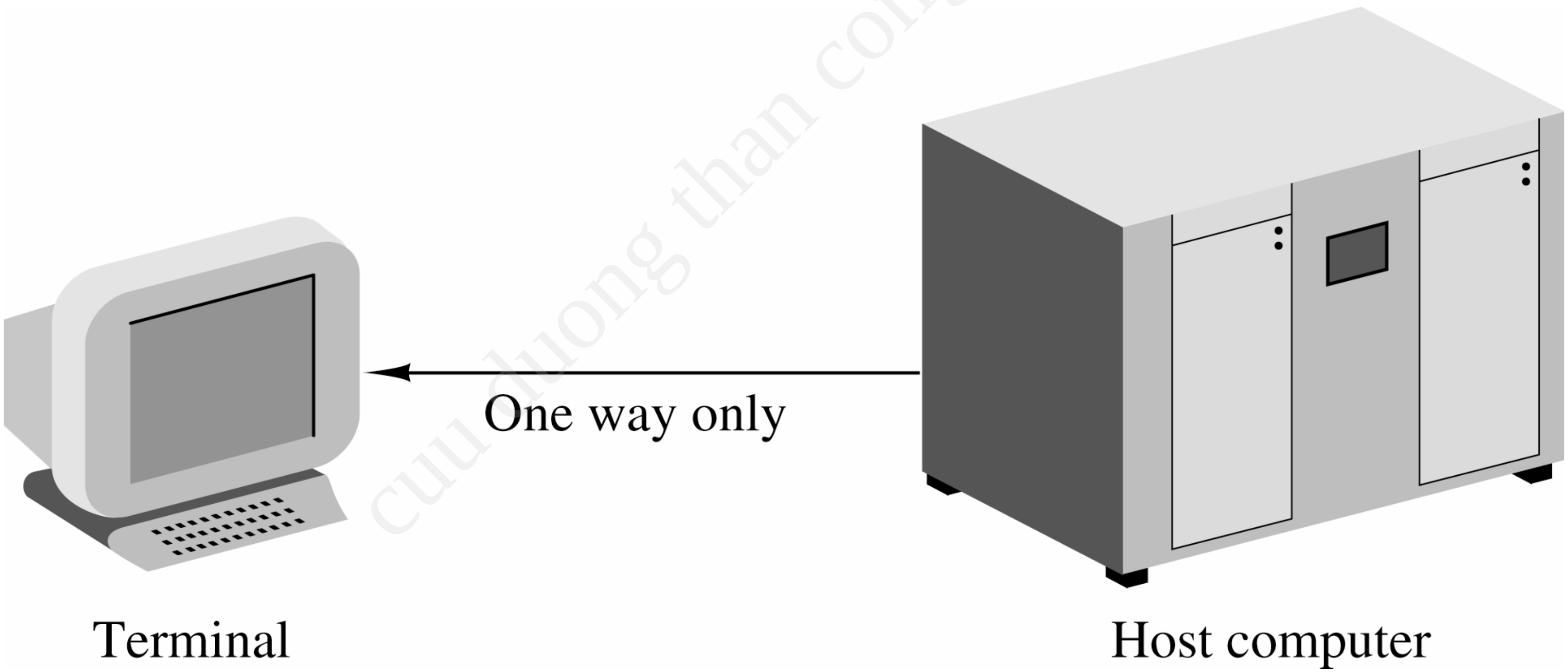
ASCII control characters			
BEL	Bell	EM	End of Medium
CAN	Cancel	ESC	Escape
DC1	Device Control 1	NUL	Null
DC2	Device Control 2	SI	Shift In
DC3	Device Control 3	SO	Shift Out
DC4	Device Control 4	SUB	Substitute
DEL	Delete		
Control codes			
ACK	Acknowledge	ETX	End of Text
DLE	Data Link Escape	NAK	Negative Acknowledge
ENQ	Enquiry	SOH	Start of Heading
EOT	End of Transmission	STX	Start of Text
ETB	End of Transmission Block	SYN	Synchronous Idle
Format effectors			
BS	Backspace	HT	Horizontal Tabulation
CR	Carriage Return	LF	Line Feed
FF	Form Feed	VT	Vertical Tabulation
Information separators			
FS	File Separator	RS	Record Separator
GS	Group Separator	US	Unit Separator

3.2 Cấu hình kết nối cơ bản

- Điểm – điểm (point - point).
- Đa điểm (Multipoint - Multidrop).
- Mắc lưới (Mesh).
- Sao (Star).
- Vòng(Ring).

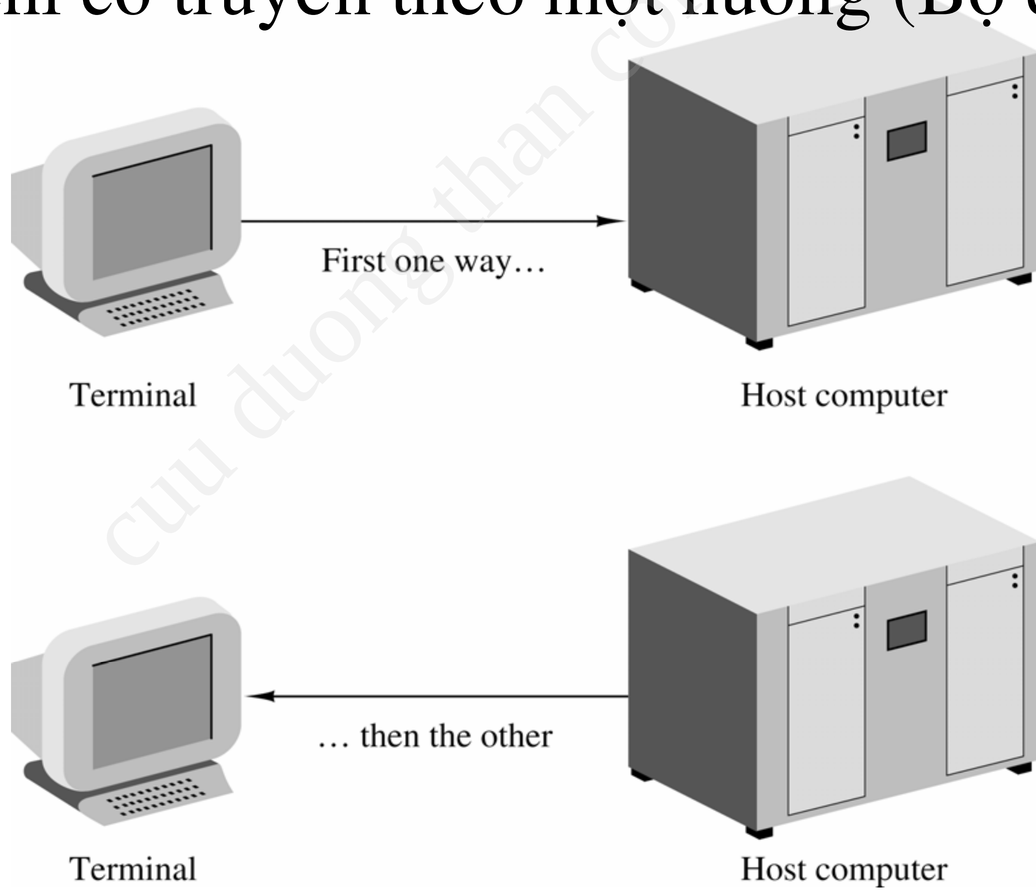
3.3 Các kiểu thông tin :

- Đơn công (Simplex): thông tin chỉ được truyền theo một hướng duy nhất (radio, tivi...)



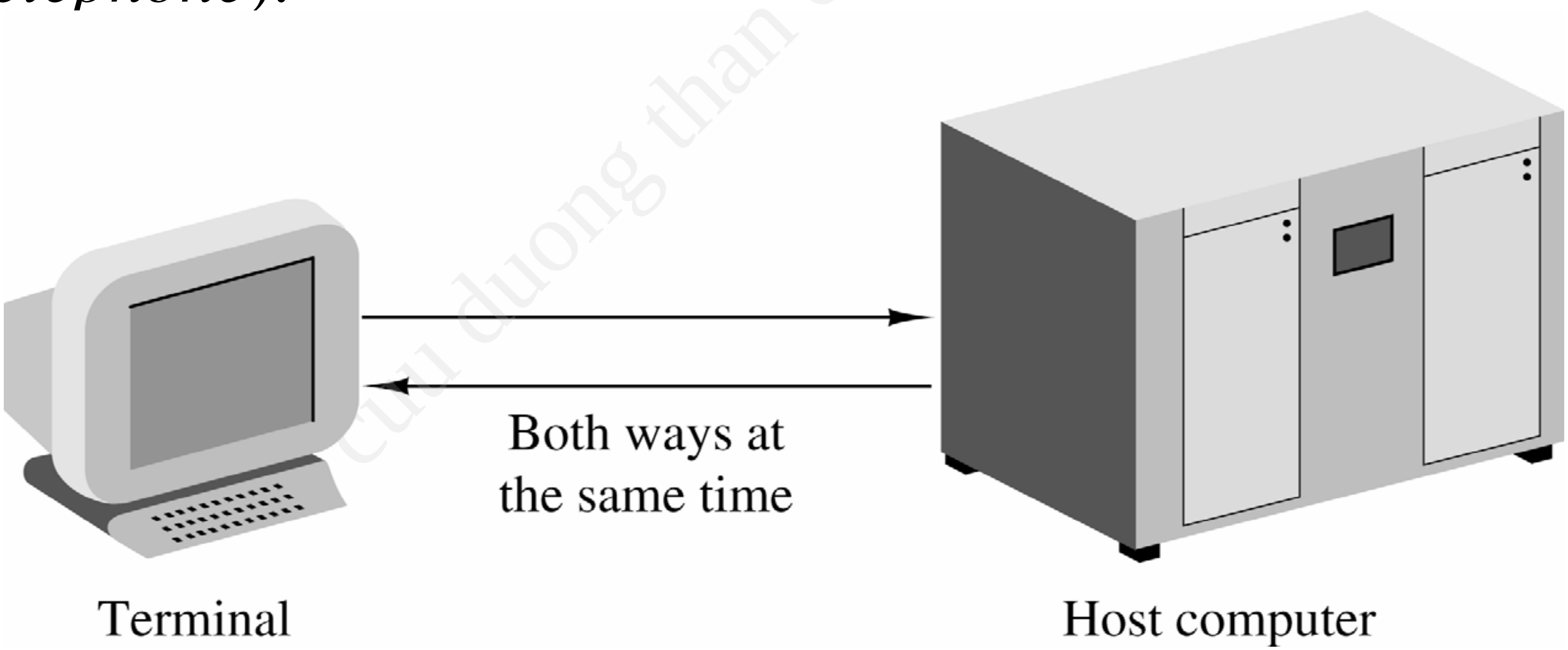
Các kiểu thông tin

- Bán song công (*half-duplex*): thông tin được truyền theo hai chiều nhưng không đồng thời, tại mỗi thời điểm thông tin chỉ có truyền theo một hướng (Bộ đàm)



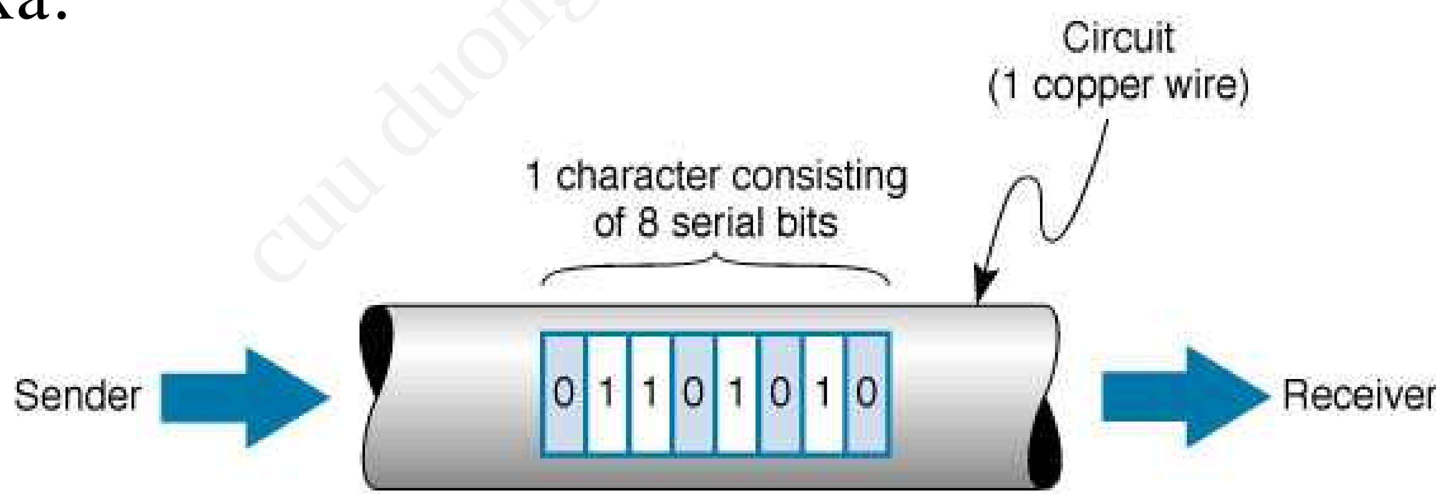
Các kiểu thông tin

- Song công (*full-duplex*): thông tin có thể được truyền 2 chiều tại cùng một thời điểm trên tuyến dữ liệu (*telephone*).



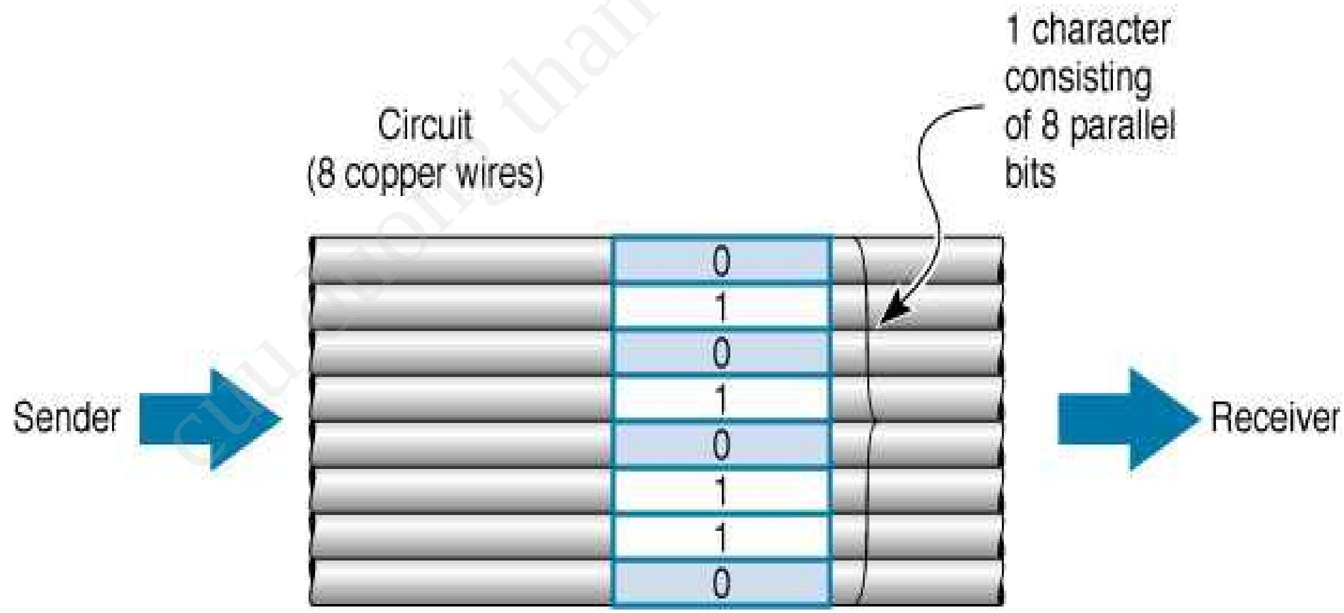
3.4 Các kiểu truyền

- **Để truyền các bit dữ liệu từ nơi phát đến nơi thu trên đường truyền vật lý ta có thể truyền theo 2 hình thức:**
- Truyền nối tiếp (Serial): Các bit được gửi lần lượt trên đường truyền. Tốc độ thấp,khoảng cách truyền xa.



Các kiểu truyền

- Truyền song song (Parallel): Các bit được gửi cùng lúc trên nhiều dây khác nhau. Tốc độ cao, khoảng cách truyền ngắn.



Các kiểu truyền

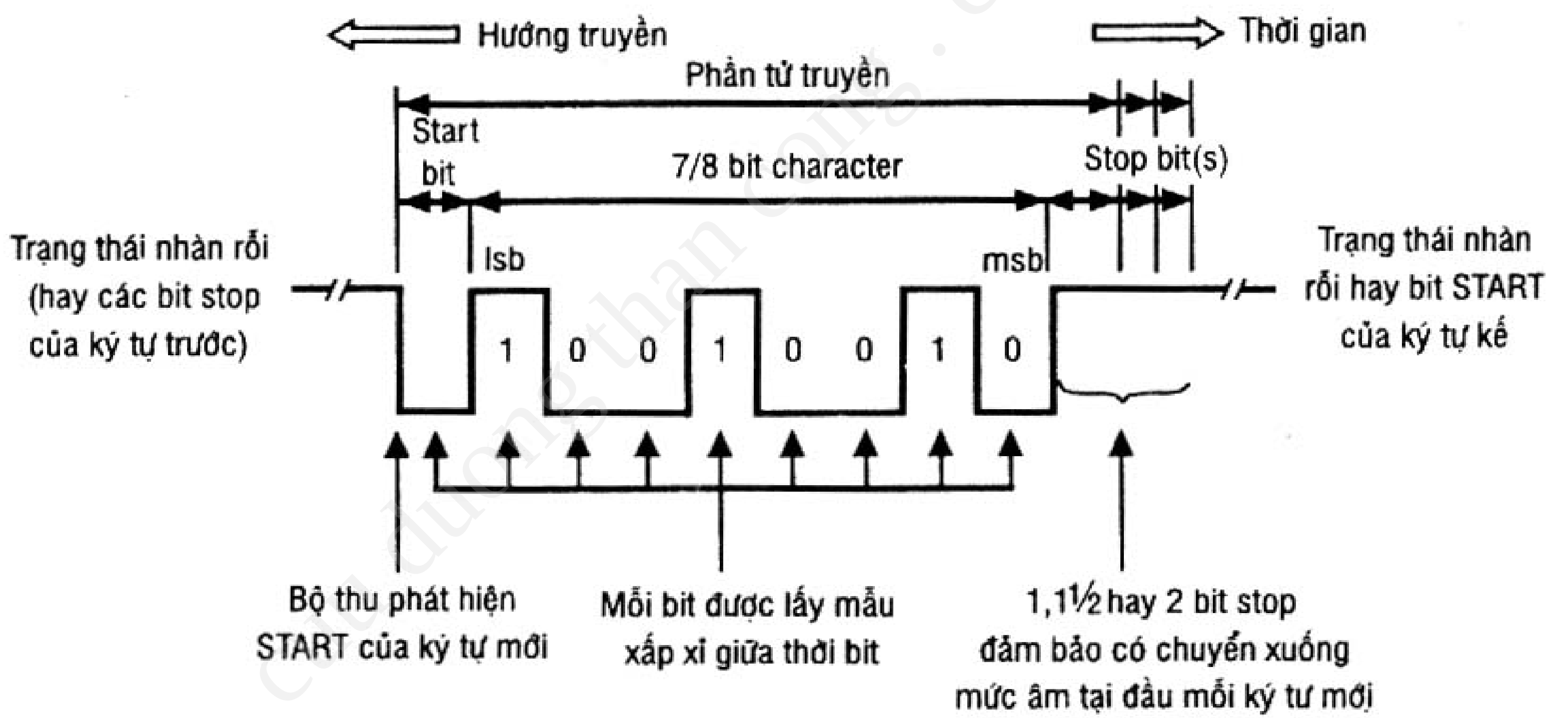
- **Để bên thu xác định và hiểu đúng các bit dữ liệu truyền đến thì phải thực hiện được những yêu cầu sau:**
  - Xác định thời điểm bắt đầu của mỗi bit trong một chu kỳ-> bit / clock synchronization
  - Xác định được vị trí bắt đầu và kết thúc của mỗi ký tự / byte. -> character / byte synchronization *(Có thể không cần thiết tùy theo kiểu truyền).*
  - Xác định vị trí bắt đầu và kết thúc của mỗi khung dữ liệu → frame synchronization

Có 2 kiểu truyền :

- Truền bất đồng bộ (**Asynchronous transmission**) **.**
- Truyền đồng bộ (**Synchronous transmission**) **.**

3.5 Truyền bất đồng bộ (Asynchronous transmission)

- Đặc điểm :

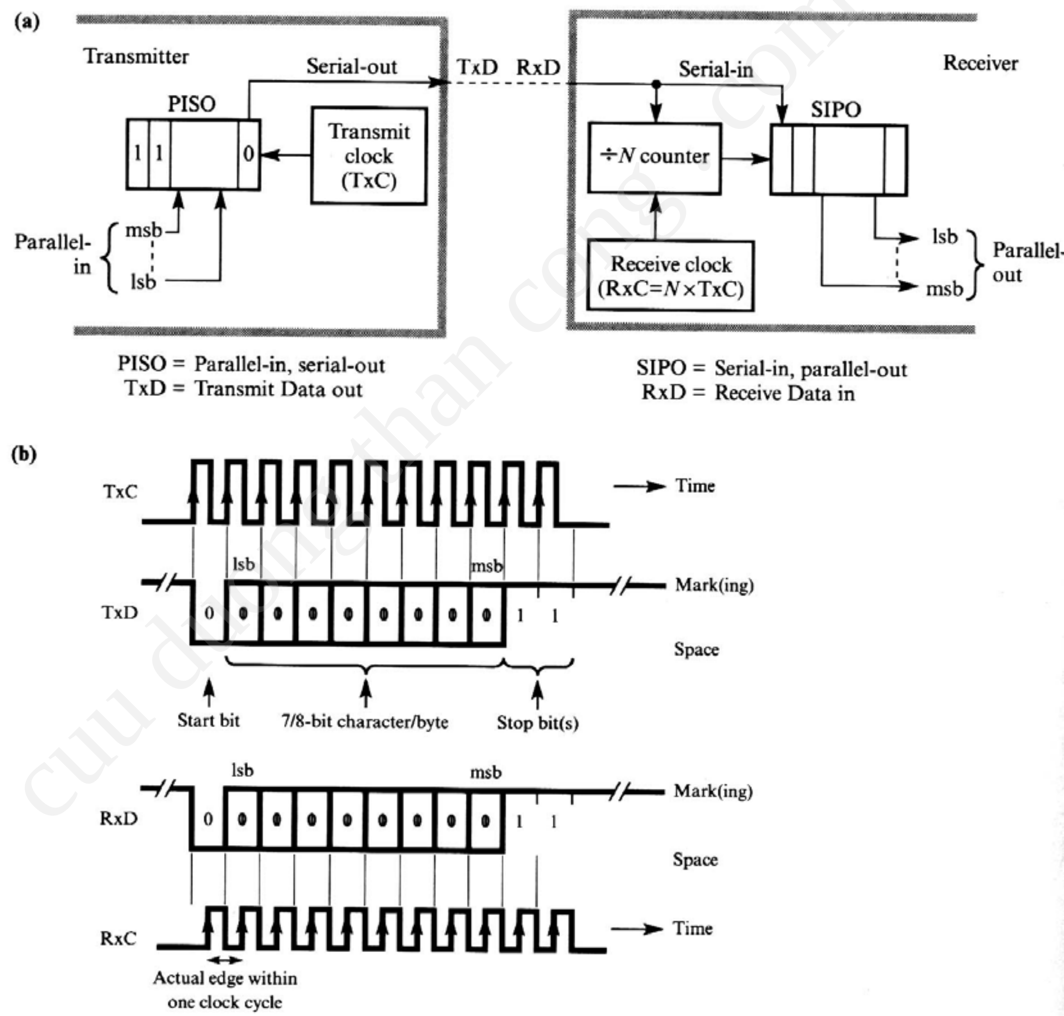


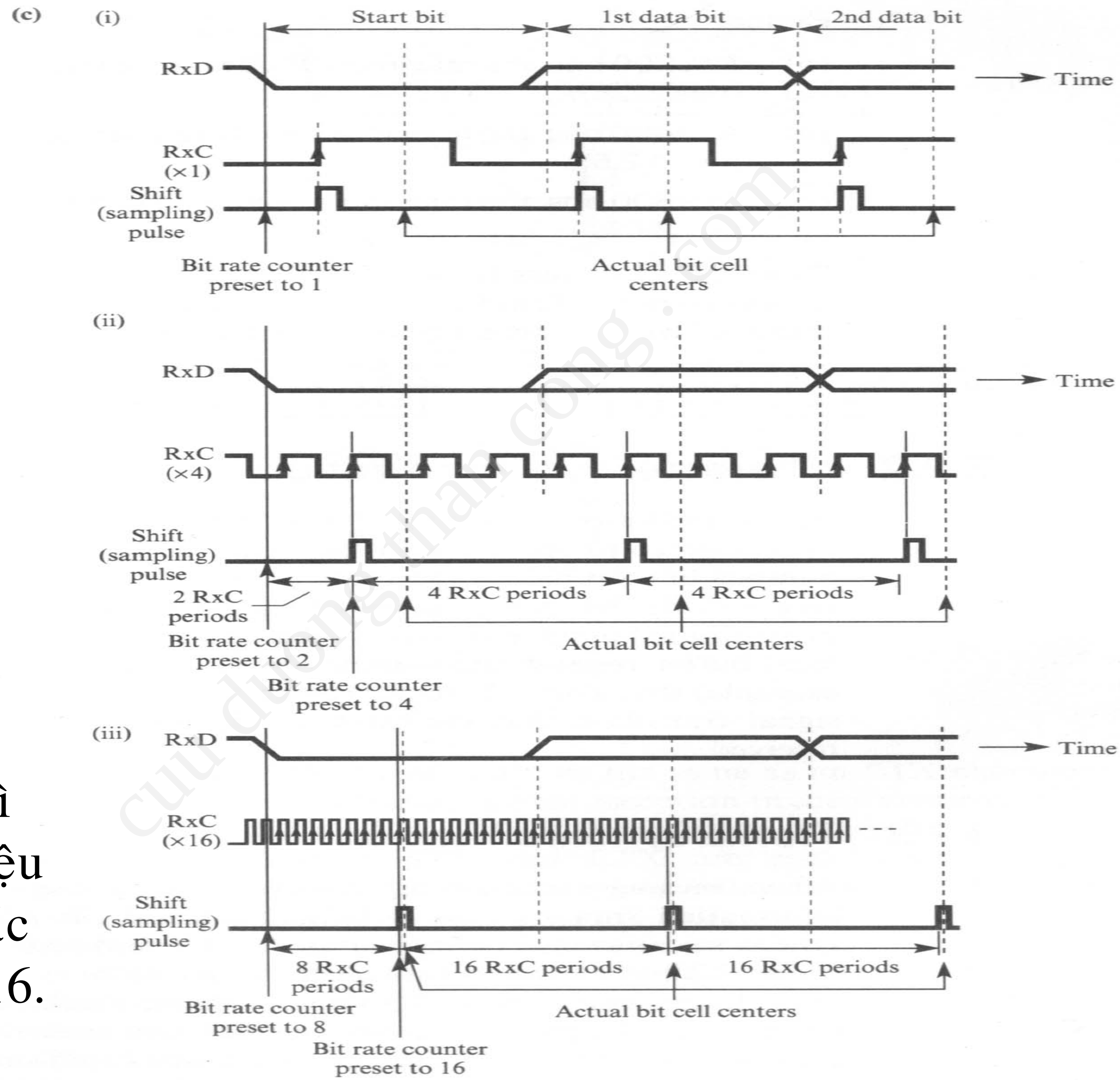
- Đồng bộ khung

Truyền bất đồng bộ (Asynchronous transmission)

- Đồng bộ bit :
  - Do đồng hồ phía thu và phát chạy độc lập nhau -> Phải làm sao lấy mẫu càng gần trung tâm bit càng tốt.
  - Nguyên lý :
    - Tần số xung clock đồng hồ thu lớn gấp N lần tần số xung clock củađồng hồ phát.
    - Khi phát hiện được trạng thái chuyển đổi mức điện áp (*vị trí bắt đầu của start bit và vị trí kết thúc của bit stop bit trước đó hay trạng thái nghĩ của đường truyền*) thì phía thu sẽ chờ sau N/2 chu kỳ xung clock thu (*vị trí giữa của start bit*) để lấy mẫu.
    - Sau đó cứ sau mỗi N chu kỳ xung clock (*vị trí giữa mỗi bit*) thu phía thu sẽ lấy mẫu bit dữ liệu thu. Điều này được thực hiện cho đến hết ký tự.

Truyền bất đồng bộ (Asynchronous transmission)





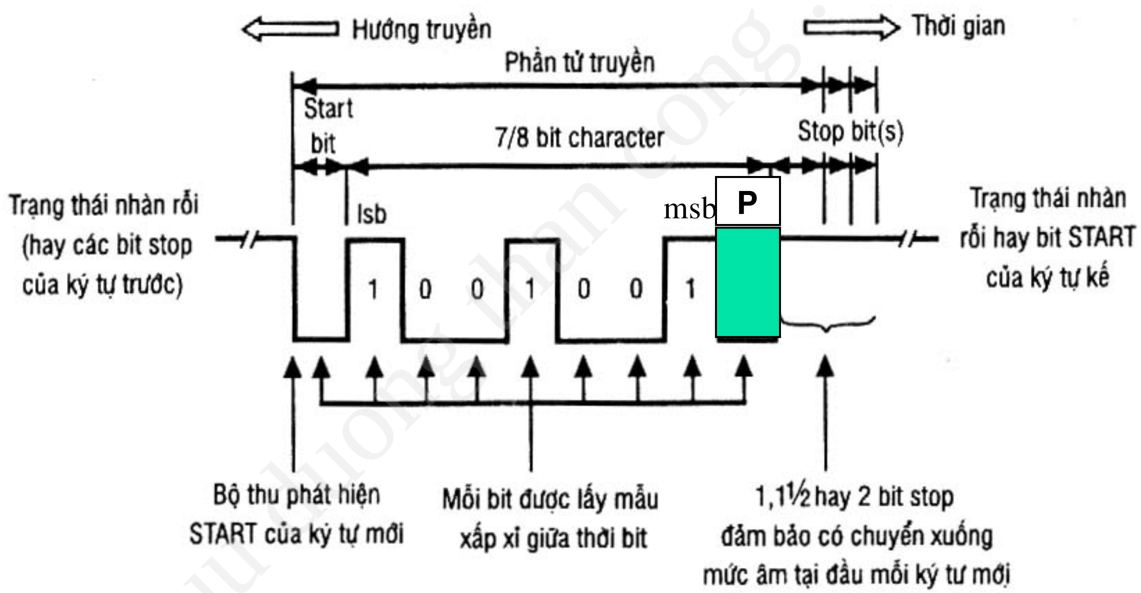
Nhận xét :

N càng lớn thì lấy mẫu dữ liệu càng chính xác

Thường N = 16.

Truyền bất đồng bộ (Asynchronous transmission)

- Đồng bộ ký tự (character)/ byte



- **Start bit** : "0" - 1bit. **Stop bits** :'1' - 1,1.5,2 bit. Data bits : 5,6,7,8.
- Parity : *Chỉ phát hiện sai khi tổng số bit lỗi là số lẻ.Vd*
  - Even : Tổng số bit 1 (Kể cả Parity) là số chẵn. Vd
  - Odd : Tổng số bit 1 (Kể cả Parity) là số lẻ. Vd

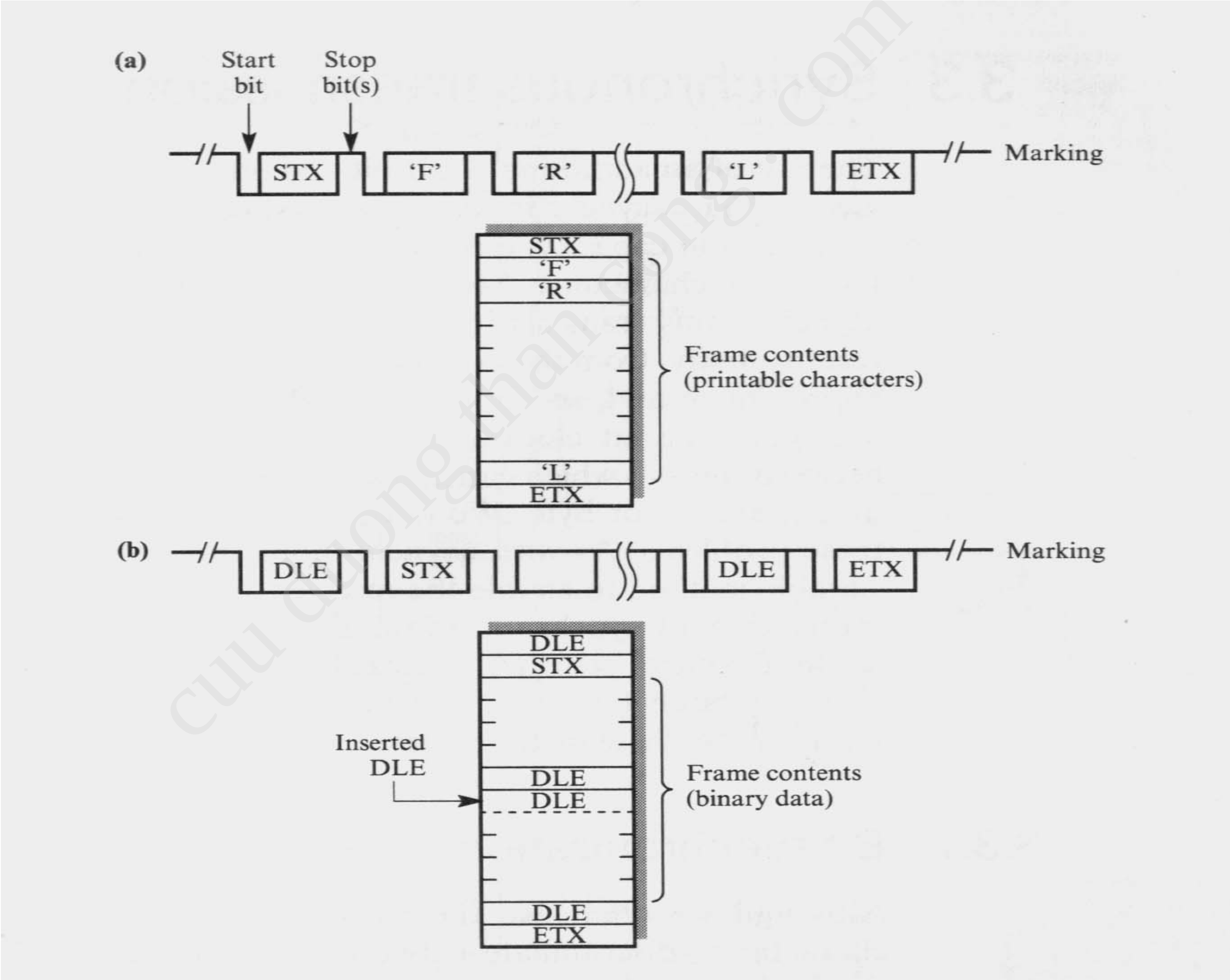
Truyền bất đồng bộ (Asynchronous transmission)

- Nguyên lý :
  - Phía phát và phía thu được lập trình để có cùng số bit trong mỗi ký tự (*start, data, parity & stop bit*).
  - Sau khi nhận được start bit, phía thu sẽ thực hiện việc đồng bộ ký tự bằng cách đếm đúng số bit đã được lập trình, sau đó chuyển nội dung ký tự vừa thu được vào bộ đệm và chờ thu ký tự mới.

Truyền bất đồng bộ (Asynchronous transmission)

- Đồng bộ khung (frame) :
  - Frame là những ký tự in được : Đóng khung toàn bộ khối bằng 2 ký tự đặc biệt :
    - STX (Start of Text) : Bắt đầu khung.
    - ETX (End of Text) : kết thúc khung.
  - Frame có những ký tự không in được :
    - Thêm ký tự DLE (Data Link Escape) trước STX và ETX .
    - Nếu dữ liệu muốn phát trùng với DLE thì áp dụng phương pháp nhồi ký tự hay nhồi byte (Charater Stuffing or Byte Stuffing).

Truyền bất đồng bộ (Asynchronous transmission)



Truyền bất đồng bộ (Asynchronous transmission)

- Hiệu suất truyền :
  - Ví dụ truyền 1 ký tự được mã hóa bằng mã ASCII, có data bit là 8 bit, 1bit star và 2 bit stop

$$\eta = \frac{\text{Số bit thông tin}}{\text{Tổng số bit truyền}} = \frac{8}{8+1+2} = 0.727 = 72.7\%$$

  - Nếu sử dụng thêm parity thì hiệu suất sẽ thấp hơn
- Tốc độ truyền dữ liệu hữu dụng : Giả sử ký tự truyền được tryền ra cổng nối tiếp với tốc độ 1200bps. Thì tốc độ truyền dữ liệu hiệu dụng là 1200x0.727= 872bps

- VD :

  DTE A cần truyền cho DTE B thông điệp như sau:TSLDLE

  Thông điệp trên được phát như khối tin lên đường truyền nối tiếp theo kiểu truyền bất đồng bộ, chuẩn RS232, mã ASCII, với cấu hình 7E1(7 bits dữ liệu, kiểm tra parity chẵn, 1 stop bit), tốc độ bit 1200bps,

a. Cho biết khung tin mà A cần truyền cho B

b. Vẽ dạng tín hiệu điện trên cho 10 bit đầu tiên của khung truyền.

c. Tính thời gian truyền của khung dữ liệu (Bỏ qua thời gian xử lý khác).

3.6 Truyền đồng bộ (Synchronous transmission)

- Đặc điểm :
  - Truyền bất đồng bộ có nhược điểm là khi truyền dữ liệu tốc độ cao thì phương pháp đồng bộ bít không đảm bảo độ tin cậy, hơn nữa hiệu suất truyền không cao. Kiểu truyền đồng bộ sẽ khắc phục những nhược điểm trên.
  - Dữ liệu sẽ được truyền liên tục thành từng khối trên đường truyền nên sẽ không có Start Bit và Stop bit.
  - Clock bên phát và bên thu phải đồng bộ nhau.

Mã đường dây (Line Codes)

- Dữ liệu số - Tín hiệu số
  - Là các xung điện áp rời rạc hoặc liên tục
  - Mỗi xung là 1 thành phần của tín hiệu
  - Dữ liệu nhị phân được mã hoá vào các thành phần tín hiệu này

Mã đường dây
(Line Codes)

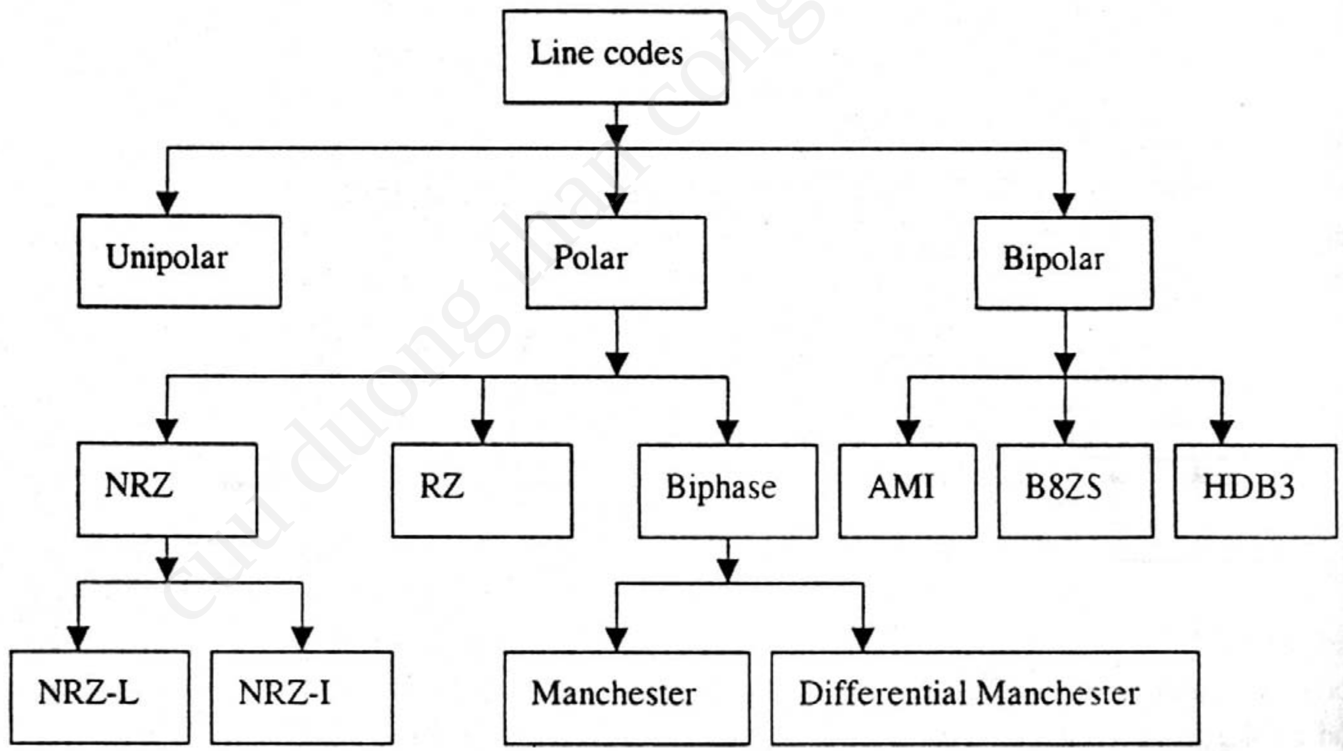
- **Các thông số cần quan tâm trong quá trình mã hoá đường dây :**
- Phổ tín hiệu
  - Không có thành phần tần số cao giảm bớt băng thông tín hiệu
  - Không có thành phần DC cho phép ghép ac bằng biến thế, tạo sự cách ly tốt
- Thông tin đồng bộ (clocking)
  - Đồng bộ giữa máy phát và máy thu
  - Dùng clock ngoài
  - Tạo cơ chế đồng bộ dựa trên tín hiệu

Mã đường dây (Line Codes)

- Phát hiện sai
  - Có thể được xây dựng dựa vào mã hoá tín hiệu
- Giao thoa tín hiệu và tính miễn nhiễu
  - Một số mã tốt hơn các mã khác
- Chi phí và độ phức tạp
  - Tốc độ càng cao thì chi phí càng cao
  - Một số mã cần tốc độ tín hiệu cao hơn tốc độ dữ liệu

Mã đường dây (Line Codes)

- Các loại mã thường dùng :



Hình 2.5.2 Các loại mã đường dây

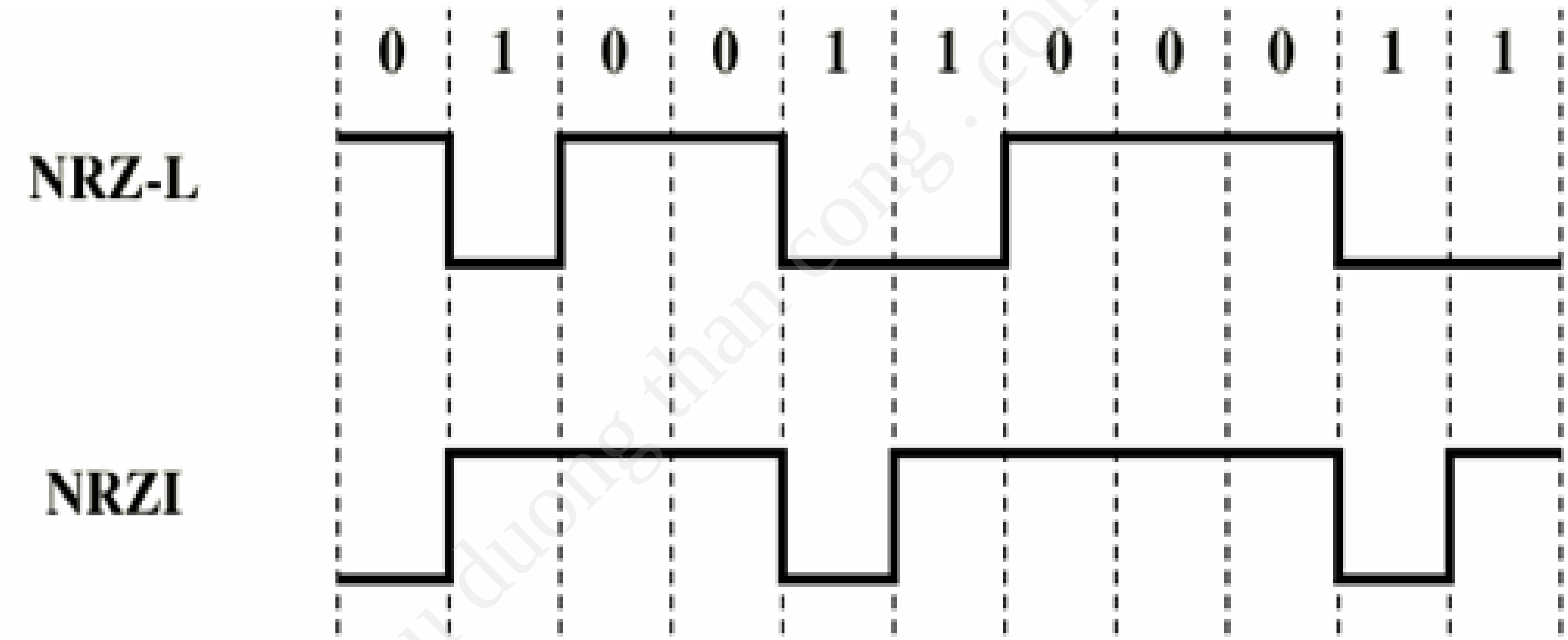
Mã đường dây (Line Codes)

- NonReturn to Zero-Level (NRZ-L)
  - Có 2 mức điện áp cho bit 0 và bit 1
  - Điện áp hằng trong suốt thời gian bit, không trở về mức điện áp 0V
  - Thông thường thì điện áp âm cho bit 1 và áp dương cho bit 0

Mã đường dây
(Line Codes)

- NonReturn to Zero Inverted
  - Đảo dấu cho bit 1
  - Điện áp hằng trong suốt thời gian bit, không trở về mức điện áp 0V
  - Cạnh xung đánh dấu bit không có cạnh xung đánh dấu bit 0

Mã đường dây
(Line Codes)

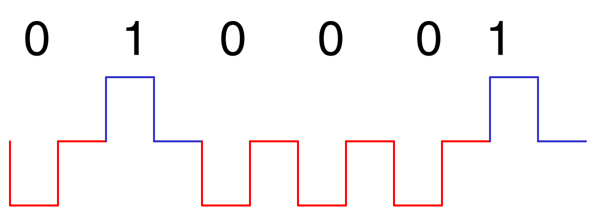


Mã đường dây (Line Codes)

- Ưu và nhược của NRZ
  - Ưu
    - Dễ dàng thực hiện
    - Sử dụng băng thông tốt
  - Nhược
    - Có thành phần DC
    - Thiếu khả năng đồng bộ
- Được sử dụng trong máy ghi từ
- Thường không được sử dụng cho truyền dẫn

Mã đường dây (Line Codes)

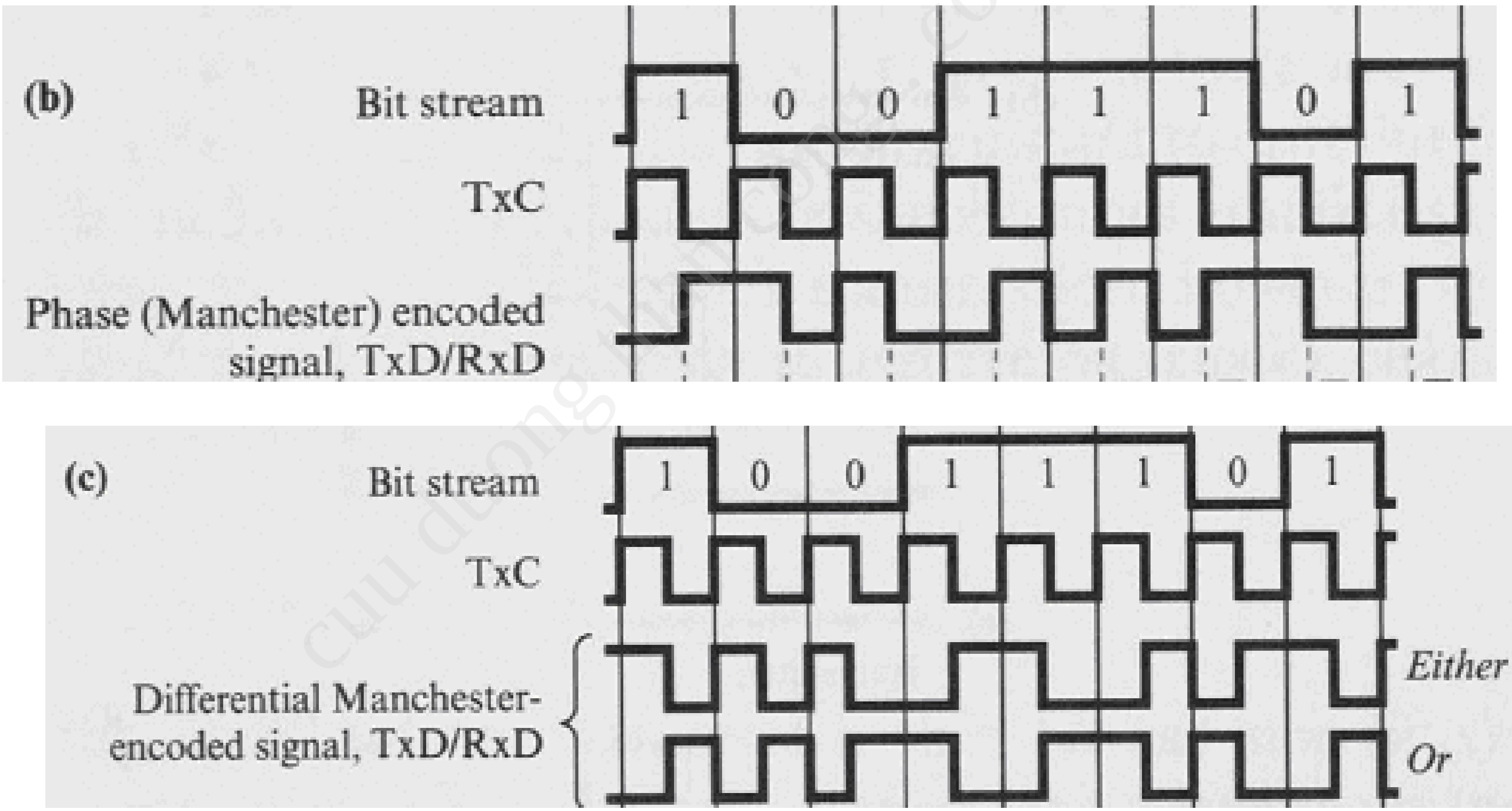
- Mã RZ :
  - Dùng 3 mức điện áp +V,0,-V.
  - Tín hiệu thay đổi trong khoảng 1 bit.
  - Bit 1 thay đổi từ +V -> 0.
  - Bit 0 thay đổi từ –V->0
- Ưu : Đảm bảo Clock để đồng bộ bit tốt.
- Khuyết : Băng thông rộng.



Mã đường dây (Line Codes)

- Mã Biphase
  - Tín hiệu thay đổi điểm giữa mỗi bit nhưng không về 0.
  - Manchester
    - Luôn có sự thay đổi trạng thái tại vị trí giữa của chu kỳ bit.
    - Bit 1 được mã hoá –V->+V
    - Bit 0 được mã hoá +V->-V
  - Manchester Vi sai
    - Tương tự như mã hoá Manchester, đảo mức tại điểm giữa của chu kỳ bit.
    - Tuy nhiên sự thay đổi mức tín hiệu tại vị trí bắt đầu của chu kỳ bit chỉ xảy ra nếu bit đó là bit 0.

Mã đường dây
(Line Codes)



Mã đường dây (Line Codes)

- Ưu, khuết điểm của mã mã Biphase
  - Ưu điểm
    - Đồng bộ ở cạnh xung giữa bit
    - Không có thành phần DC
    - Phát hiện sai : Khi có sự có mặt của cạnh xung không mong muốn
  - Nhược điểm
    - Ít nhất có 1 cạnh xung cho mỗi bit
    - Tốc độ điều chế cực đại gấp 2 lần NRZ
    - Cần băng thông rộng hơn

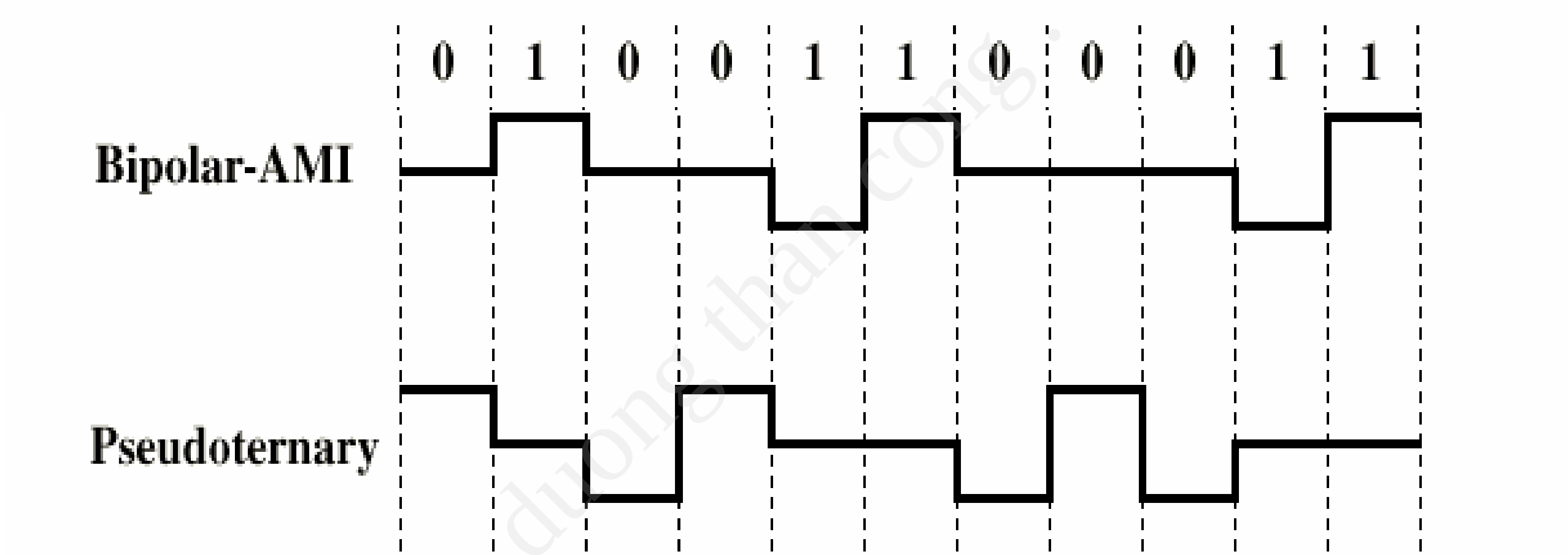
Mã đường dây (Line Codes)

- Mã Bipolar
  - Đây là mã nhị phân sử dụng nhiều hơn 2 mức
  - Mã AMI lưỡng cực
    - Bit 0 được biểu diễn bởi mức 0V
    - Bit 1 được biểu diễn bởi mức +V hoặc −V sao cho cực tính của các bit 1 gần nhau nhất luôn phiên thay đổi.
    - Ưu điểm
      - Không mất đồng bộ nếu có 1 chuỗi bit 1
      - Không có tích luỹ thành phần DC
      - Băng tần thấp
      - Dễ phát hiện sai
    - Khuyết điểm
      - Không đảm bảo đồng bộ bit nếu chuỗi bit 0 kéo dài.

Mã đường dây (Line Codes)

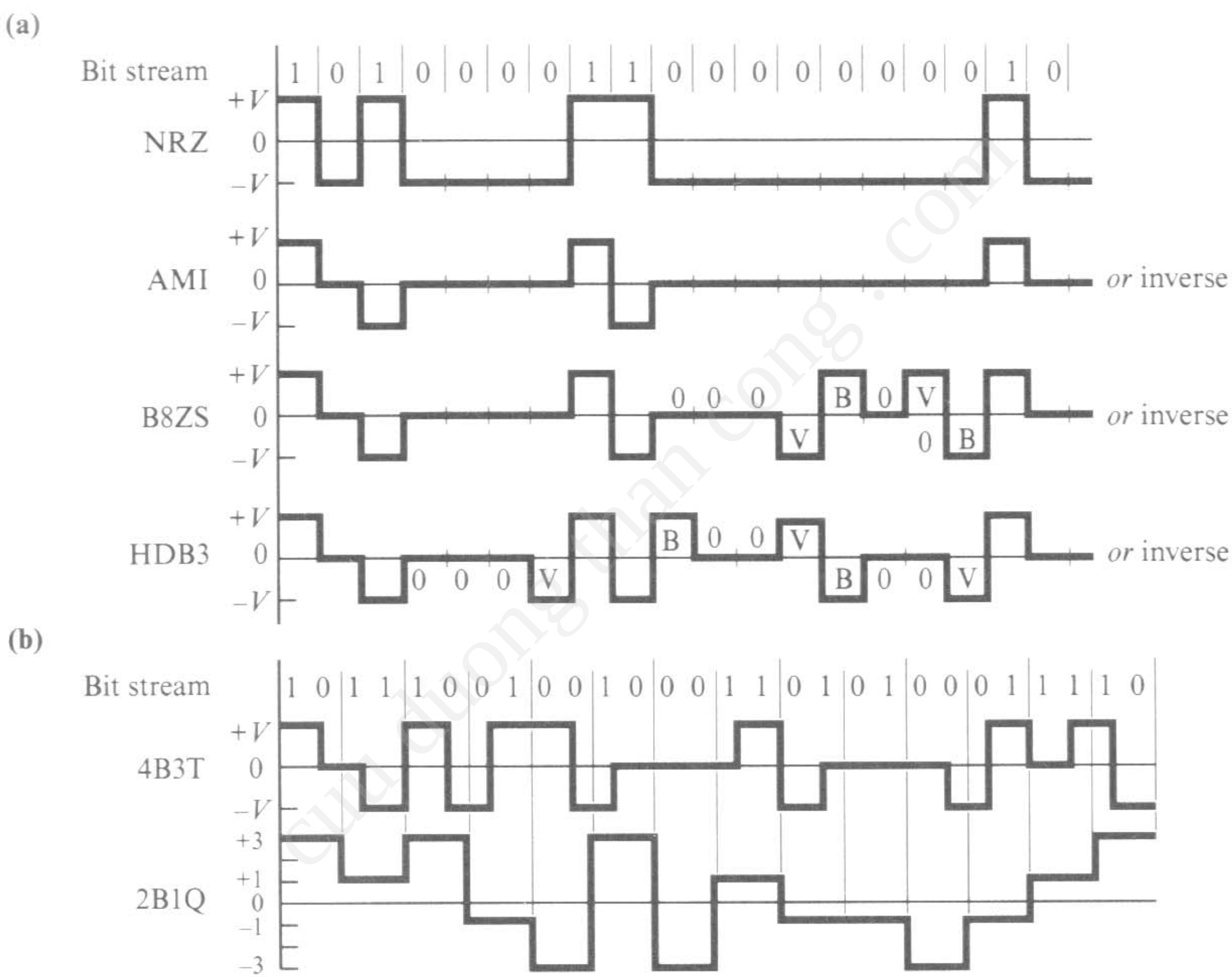
- Pseudoternary
  - Bit 1 được biểu diễn bởi không có tín hiệu trên đường truyền
  - Bit 0 được biểu diễn bằng các thay đổi luân phiên xung dương và xung âm
  - Không có ưu hay nhược so với AMI

Mã đường dây (Line Codes)



Mã đường dây (Line Codes)

- Mã HDB3
  - Kiểu mã hoá này giống với kiểu mã hoá AMI ngoại trừ một đặc điểm là nếu trong chuỗi dữ liệu phát có **4 bit 0 liên tiếp** thì sẽ được mã hoá thành **x00V.** Với
    - X = 0 Nếu tổng số bit 1 giữa 2 mã V gần nhau nhất là số lẻ.
    - X= B Nếu tổng số bit 1 giữa 2 mã V gần nhau nhất là số chẵn.
      - **'B'**đảo cực so với bit 1 gần nhất trước nó (*đúng luật mã AMI*).
      - '**V' (***violation***)** được mã hoá cùng cực tính so với bit 1 gần nhất trước (*vi phạm luật mã AMI*)
- Mã B8ZS
  - Nếu trong chuỗi dữ liệu phát có **8 bit 0 liên tiếp** thì sẽ được mã hoá thành chuỗi bit là **000VB0VB.** Trong đó:
    - *Lưu ý:* Trong chuỗi bit phát sử dụng kiểu mã hoá này chỉ có tối đa 7 bit 0 liên tiếp.


(a)
Bit stream
1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0
NRZ
AMI
B8ZS
HDB3
+V
0
−V
or inverse
(b)
Bit stream
1 0 1 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 1 0
4B3T
2B1Q
+3
+1
0
−1
−3

Mã đường dây (Line Codes)

- Các hạn chế của mã nhị phân đa mức
  - Không hiệu quả bằng NRZ
    - Mỗi thành phần tín hiệu biểu diễn chỉ 1 bit
    - Trong hệ thống 3 mức có thể biểu diễn $\log_2 3 = 1.58$bits
    - Máy thu phải phân biệt được 3 mức tín hiệu
    - Cần công suất cao hơn 3dB với cùng xác suất lỗi bit

Mã đường dây (Line Codes)

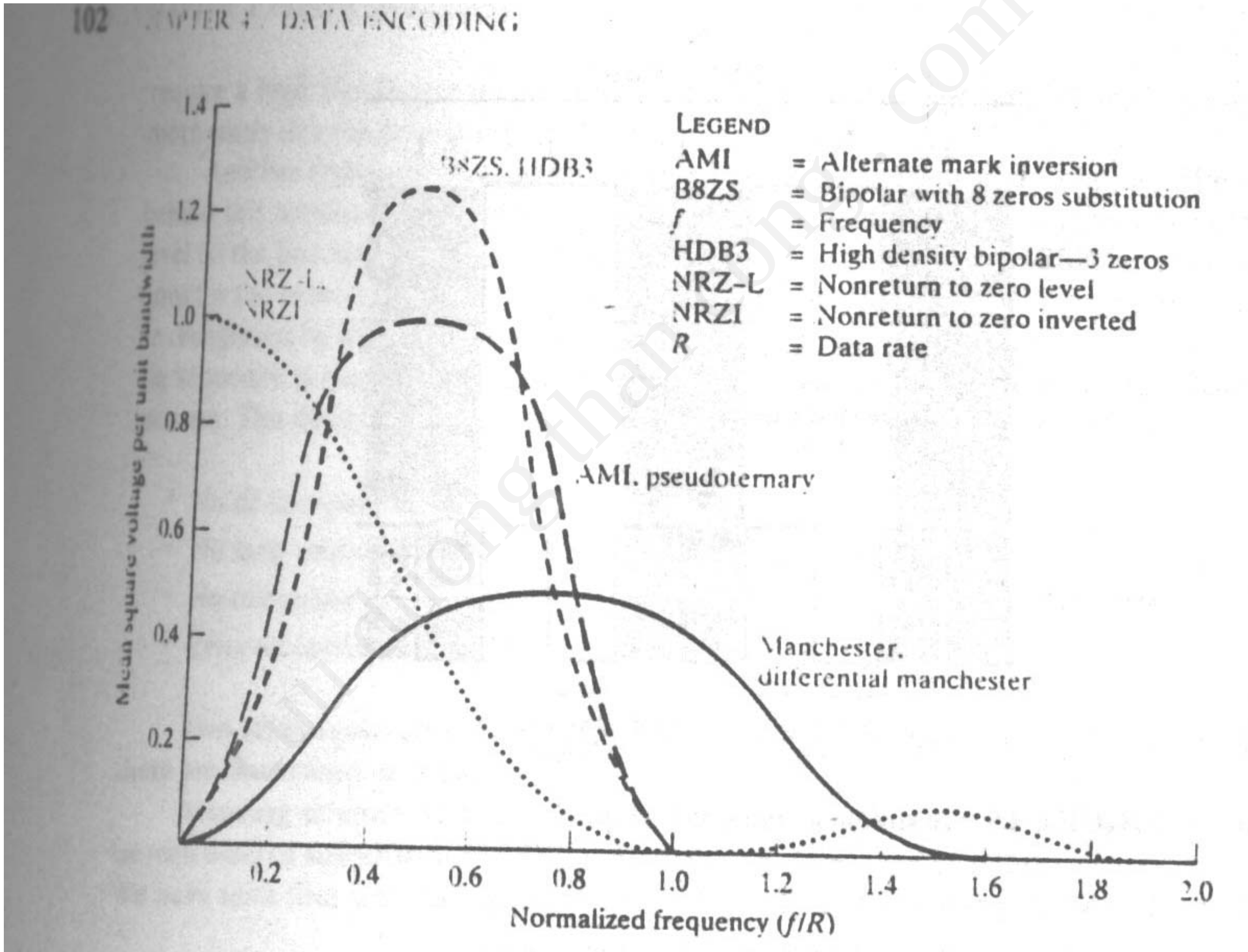


FIGURE 4.3 Spectral density of various signal encoding schemes.

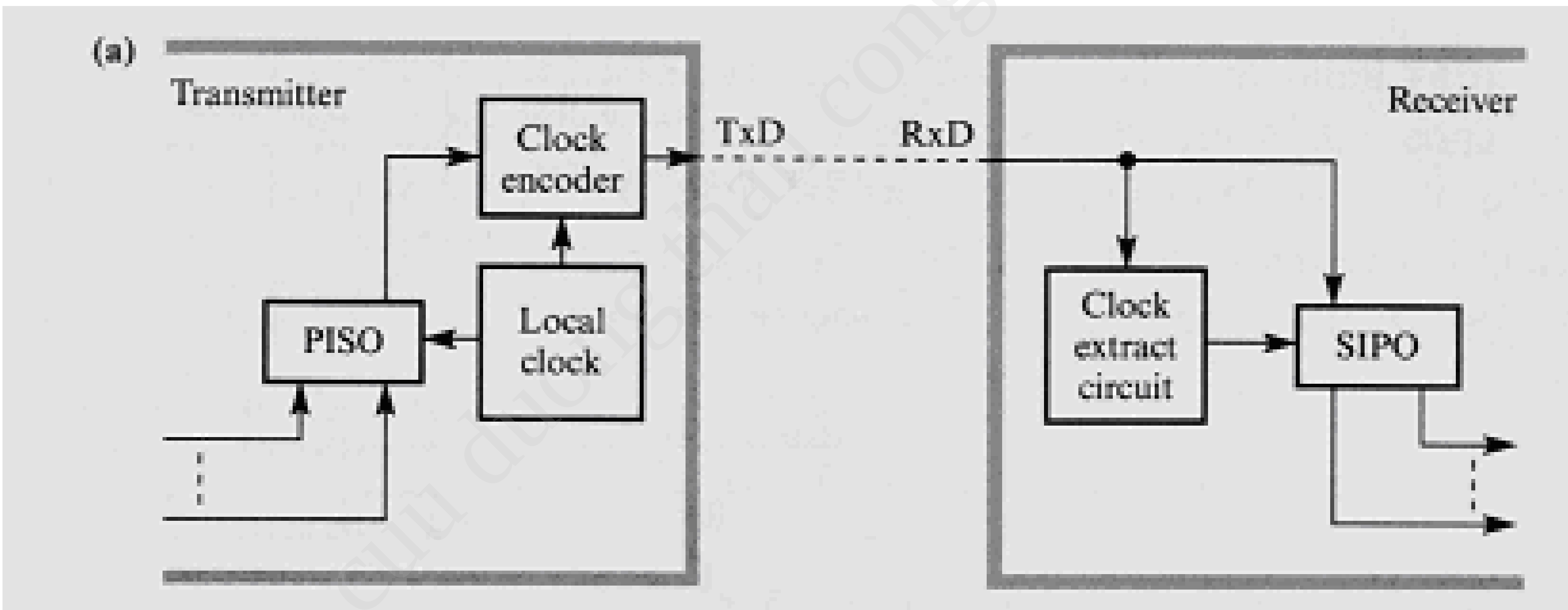
Truyền đồng bộ (Synchronous transmission)

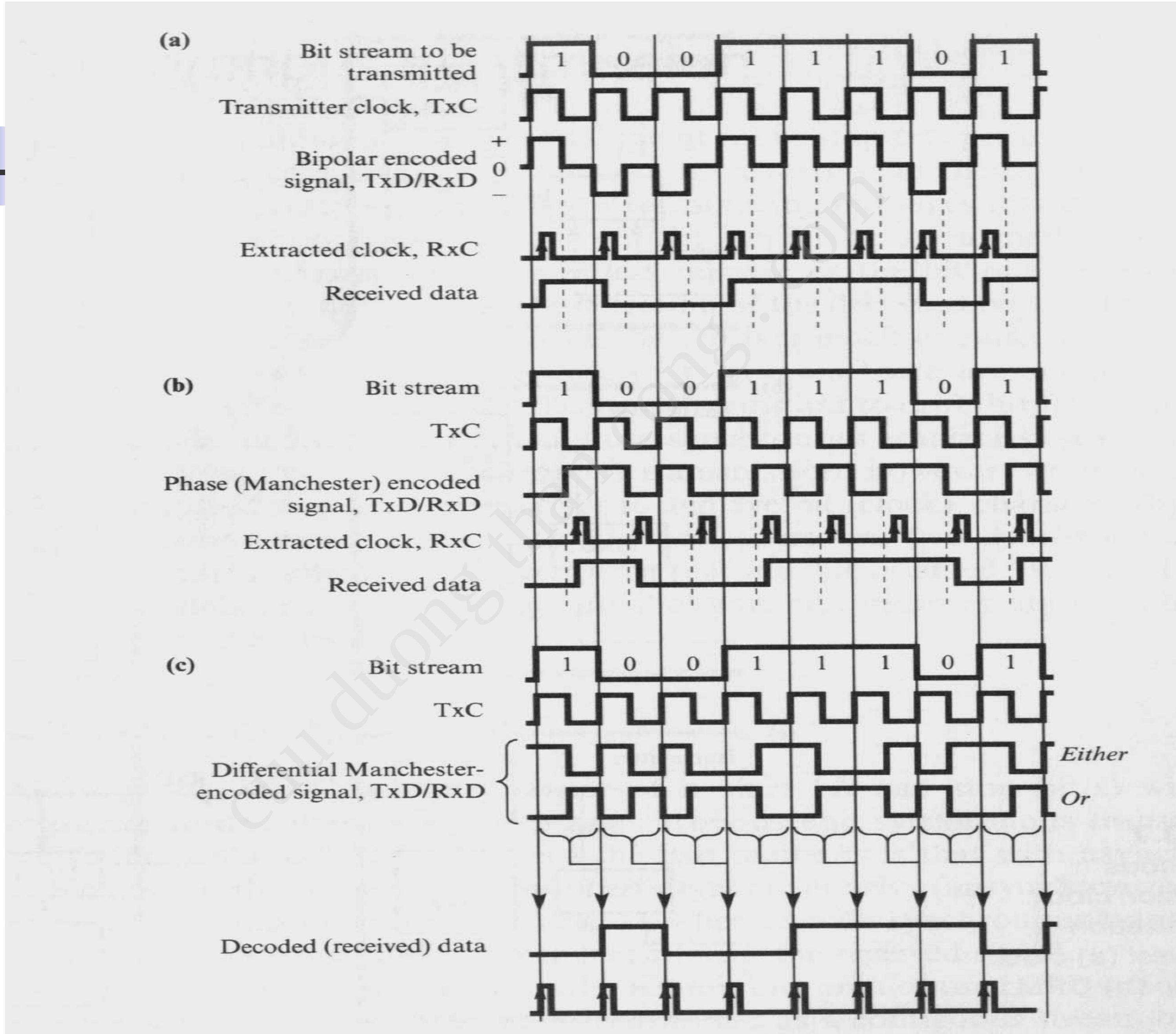
- Kỹ thuật đồng bộ trong kiểu truyền đồng bộ
  - Đồng bộ bit :
    - Clock encoding and extraction
    - Digital Phase-lock-loop (DPLL)
    - Hybrid
  - Đồng bộ khung :
    - Character-oriented
    - Bit-oriented

Truyền đồng bộ (Synchronous transmission)

- Đồng bộ bit :
  - Clock encoding and extraction
    - Phía phát gửi xung clock vào tín hiệu phát bằng cách mã hóa dữ liệu trước khi phát thông qua mạch Clocl Encoder. Phía thu sẽ trích tín hiệu clock từ tín hiệu nhận được nhờ mạch Clock Extract Circuit.
    - Mã đường dây : RZ, Manchester, differential Manchester

Truyền đồng bộ (Synchronous transmission)



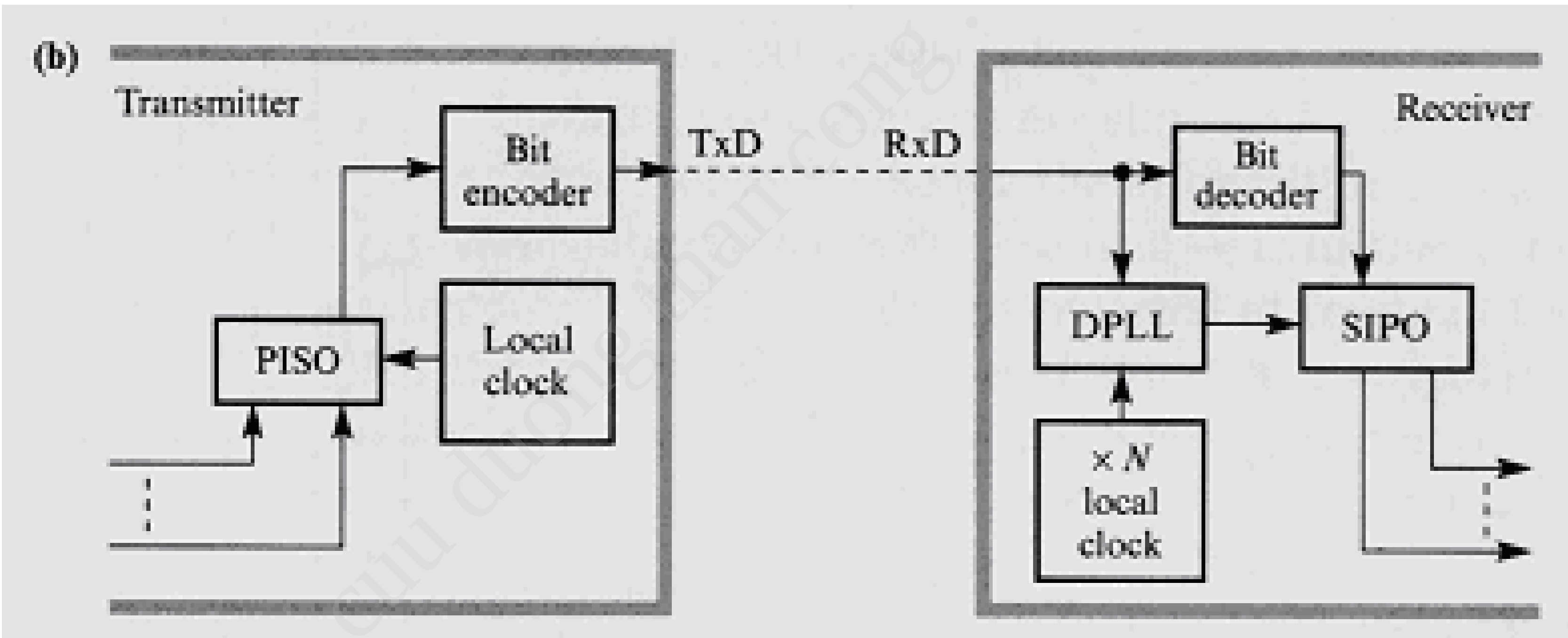


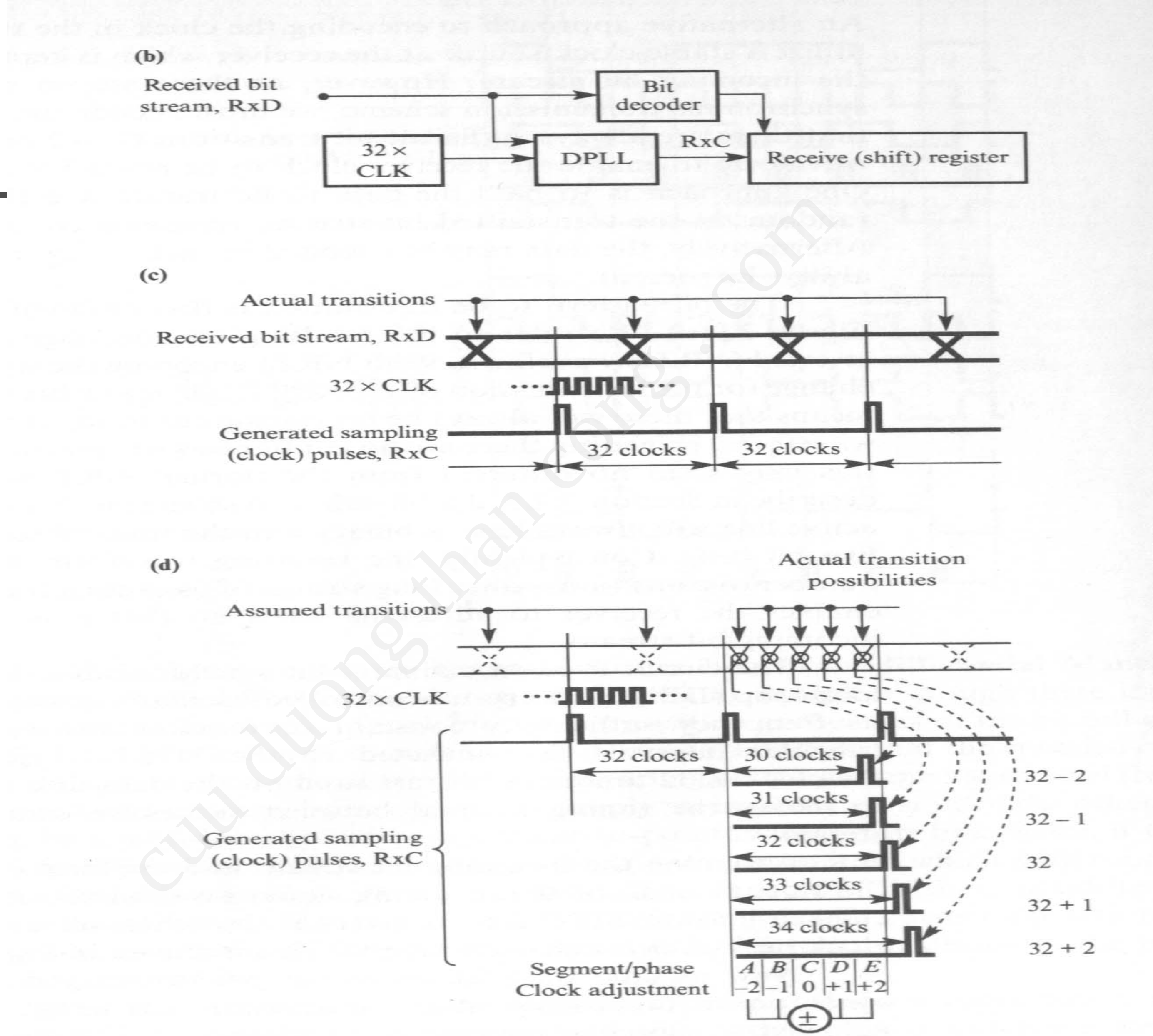
(a)
Bit stream to be transmitted
1 0 0 1 1 1 0 1
Transmitter clock, TxC
Bipolar encoded signal, TxD/RxD
+
0
–
Extracted clock, RxC
Received data
(b)
Bit stream
1 0 0 1 1 1 0 1
TxC
Phase (Manchester) encoded signal, TxD/RxD
Extracted clock, RxC
Received data
(c)
Bit stream
1 0 0 1 1 1 0 1
TxC
Differential Manchester-encoded signal, TxD/RxD
Either
Or
Decoded (received) data

Truyền đồng bộ (Synchronous transmission)

- Digital Phase-lock-loop (DPLL):
  - Bộ thu đồng bộ với bộ phát nhờ vào vòng khóa pha số. Phía thu sử dụng đồng hồ có tần số gấp N lần phía phát cấp cho PLL. PLL có nhiệm vụ tạo tín hiệu clock cho thanh ghi SIPO từ tín hiệu đồng hồ và tín hiệu nhận được sao cho đúng giữa chu kỳ bit.
  - Để clock thu duy trì được sự đồng bộ với clock phát thì chuỗi dữ liệu phát phải được ***mã hoá*** để có đủ sự thay đổi trạng thái (*1 → 0 hay 0 →1*).
  - Mã đường dây : NRZ, AMI, HDB3, B3ZS, B6ZS, B8ZS, 4B3T, 2B1Q

Truyền đồng bộ (Synchronous transmission)




(b)
Received bit stream, RxD
Bit decoder
32 × CLK
DPLL
RxC
Receive (shift) register
(c)
Actual transitions
Received bit stream, RxD
32 × CLK
Generated sampling (clock) pulses, RxC
32 clocks
32 clocks
(d)
Actual transition possibilities
Assumed transitions
32 × CLK
32 clocks
30 clocks
31 clocks
32 clocks
33 clocks
34 clocks
32 – 2
32 – 1
32
32 + 1
32 + 2
Generated sampling (clock) pulses, RxC
Segment/phase
Clock adjustment
A B C D E
–2 –1 0 +1 +2
±

Truyền đồng bộ (Synchronous transmission)

- Thông thường clock thu có tần số gấp *N=32* lần tần số clock phát. Bộ tạo dao động này được nối tới DPLL nhằm duy trì sự đồng bộ.
- DPLL là một bộ phận được sử dụng để duy trì sự đồng bộ bit giữa bộ tạo xung clock thu với chuỗi dữ liệu thu vào. Việc duy trì sự đồng bộ này được dựa trên sự thay đổi trạng thái trong chuỗi dữ liệu thu được.
- Trong trường hợp clock thu và chuỗi dữ liệu thu duy trì được sự đồng bộ với chuỗi dữ liệu thu vào (hình 3.3.3 c), bit dữ liệu thu sẽ được lấy mẫu ngay tại vị trí giữa chu kỳ bit sau mỗi 32 xung clock.

Truyền đồng bộ (Synchronous transmission)

- Trong trường hợp Clock thu và chuỗi dữ liệu thu không đồng bộ thì xung lấy mẫu sẽ được hiệu chỉnh trong vòng từ 30 đến 34 xung Clock
- Nếu trong một khoảng thời gian dài không có trạng thái chuyển đổi, DPLL sẽ phát ra một xung lấy mẫu sau mỗi 32 chu kỳ clock. Khi đó, phía thu có thể không duy trì được sự đồng bộ với chuỗi dữ liệu thu vào (*hình 3.3.3 d*). Khi phát hiện được trạng thái chuyển đổi trên đường dây, DPLL sẽ so sánh nó với thời điểm dịch chuyển giả sử, vàhiệu chỉnh xung lấy mẫu tương ứng với sự chênh lệch này. Qúa trình này được thực hiện như sau :

Truyền đồng bộ (Synchronous transmission)

- 1 chu kỳ bit chia thành 5 đoạn A, B,C,D,E nhu hình vẽ.
- Sự chênh lệch vị trí chuyển mức có thể xảy ra trong các đoạn A, B, C, D, E (*hình 3.3.3 d*).
- Nếu vị trí chuyển đổi xảy ra trong đoạn A, thì vị trí xung lấy mẫu cuối cùng trước đó rất gần với sự chuyển đổi trạng thái kế nó, nghĩa là vị trí lấy mẫu bị trễ (tốc độ lấy mẫu chậm). Do đó DPLL sẽ hiệu chỉnh bằng cách rút ngắn khoảng thời gian lấy mẫu xuống còn $32 - 2 = 30$ clock.
- Ngược lại, nếu vị trí chuyển đổi xảy ra như trong trường hợp E, thì vị trí xung lấy mẫu cuối cùng trước đó rất xa với sự chuyển đổi trạng thái kế nó, nghĩa là vị trí lấy mẫu bị sớm (tốc độ lấy mẫu nhanh). Do đó DPLL sẽ hiệu chỉnh bằng cách kéo dài khoảng thời gian lấy mẫu lên $32+2 =34$ clock.

Truyền đồng bộ (Synchronous transmission)

- Tương tự trong trường hợp B hoặc D, vị trí lấy mẫu trễ và sớm ít hơn so với A hoặc E, do đó DPLL sẽ hiệu chỉnh khoảng thời gian lấy mẫu tương ứng sẽ là 32-1=31 hoặc 32+1 = 33 clock.
- Trong trường hợp C, vị trí xung clock DPLL giả sử nó xảy ra trùng với vị trí chuyển đổi trạng thái thực, sự đồng bộ được duy trì, do đó không cần phải hiệu chỉnh. (*khoảng lấy mẫu vẫn là 32 xung clock*).

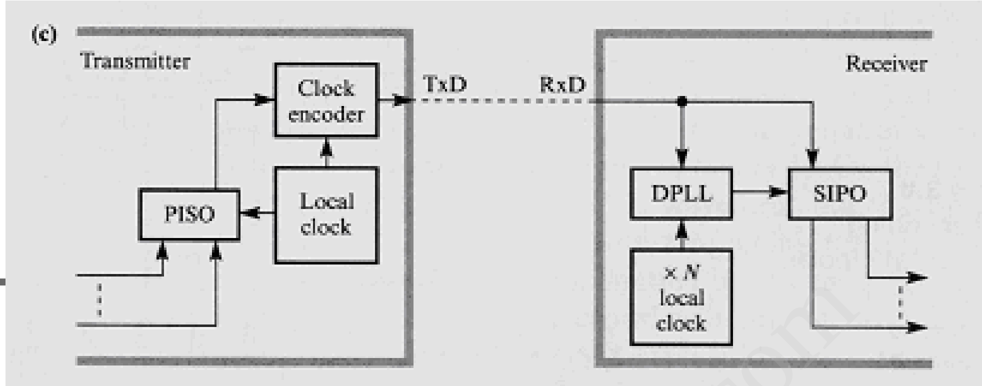
Truyền đồng bộ (Synchronous transmission)

- Bằng cách hiệu chỉnh như trên, sẽ tạo ra xung lấy mẫu càng lúc càng gần trung tâm của bit dữ liệu.
- Như vậy độ rộng của 1 bit tương đương với 32 xung clock. Vùng A.B xa nhất nên gán A=E=10 clock, B=C=D= 4 clock.
- Số bit dữ liệu tối đa để xung lấy mẫu bộ thu đồng bộ với dữ liệu nhận được được tính như sau: Khảng A hoặc E mỗi lần hiệu chỉnh 2 nhịp clock (± 2) nên để thoát ra khỏi vùng A hoặc E cần 5 bit dữ liệu cho sự hiệu chỉnh thô (vì đoạn này chiếm 10 xung clock) và tương tự để thoát ra vùng B hoặc D thì cần 4 bit dữ liệu, và cuối cùng để đảm bảo lấy chính xác tại trung tâm mỗi bít thì cần 1 bit dữ liệu nữa. Vậy tổng cộng cần 10 bit dữ liệu.

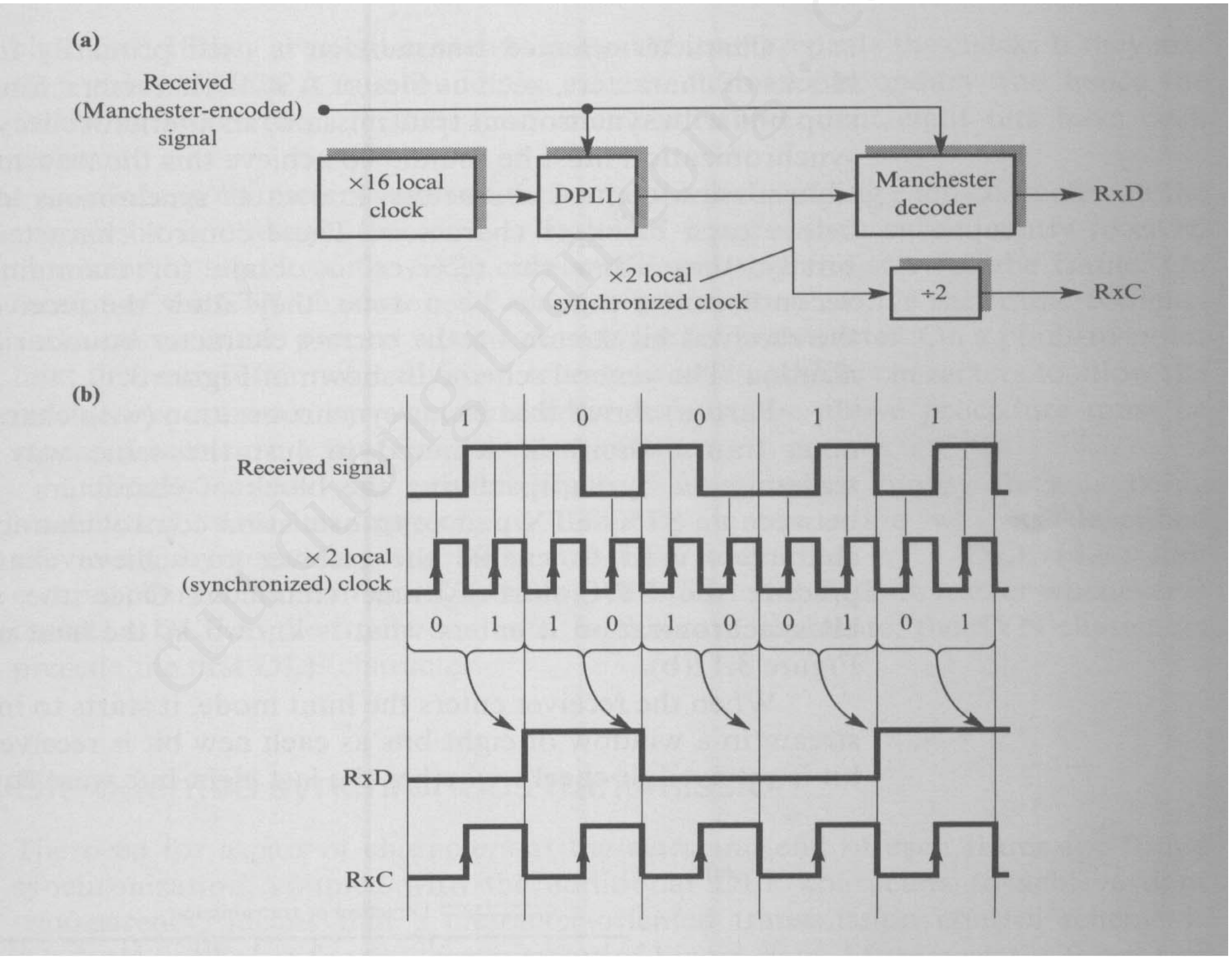
->Do đó thường trong kỹ thuật truyền đồng bộ thì các bít đồng bộ thường được phát trước khi tuyền dữ liệu thực sự, để đảm bảo đồng bộ bên phát và bên thu không ảnh hưởng đến thông tin cần truyền.

Truyền đồng bộ (Synchronous transmission)

- Hybrid :
  - Khi tốc độ bit tăng thì các phương pháp trên rất khó thực hiện đồng bộ. Để giải quyết vấn đề này ta sử dụng phương pháp Hybrid.
  - Đây là phương pháp kết hợp 2 phương pháp Clock encoder và DPLL. Clock encoder đảm bảo các bit khi nhận được có ít nhất 1 sự xáo trộn trong chu kỳ 1 bit, trong khi đó DPLL được dùng để giữ nhịp nội tại đồng bộ với dữ liệu nhận.
  - Khuyết điểm : Sử dụng băng thông lớn.
  - Mã đường dây : Manscheter



(c)
Transmitter
Clock encoder
TxD
RxD
Receiver
PISO
Local clock
DPLL
SIPO
× N local clock



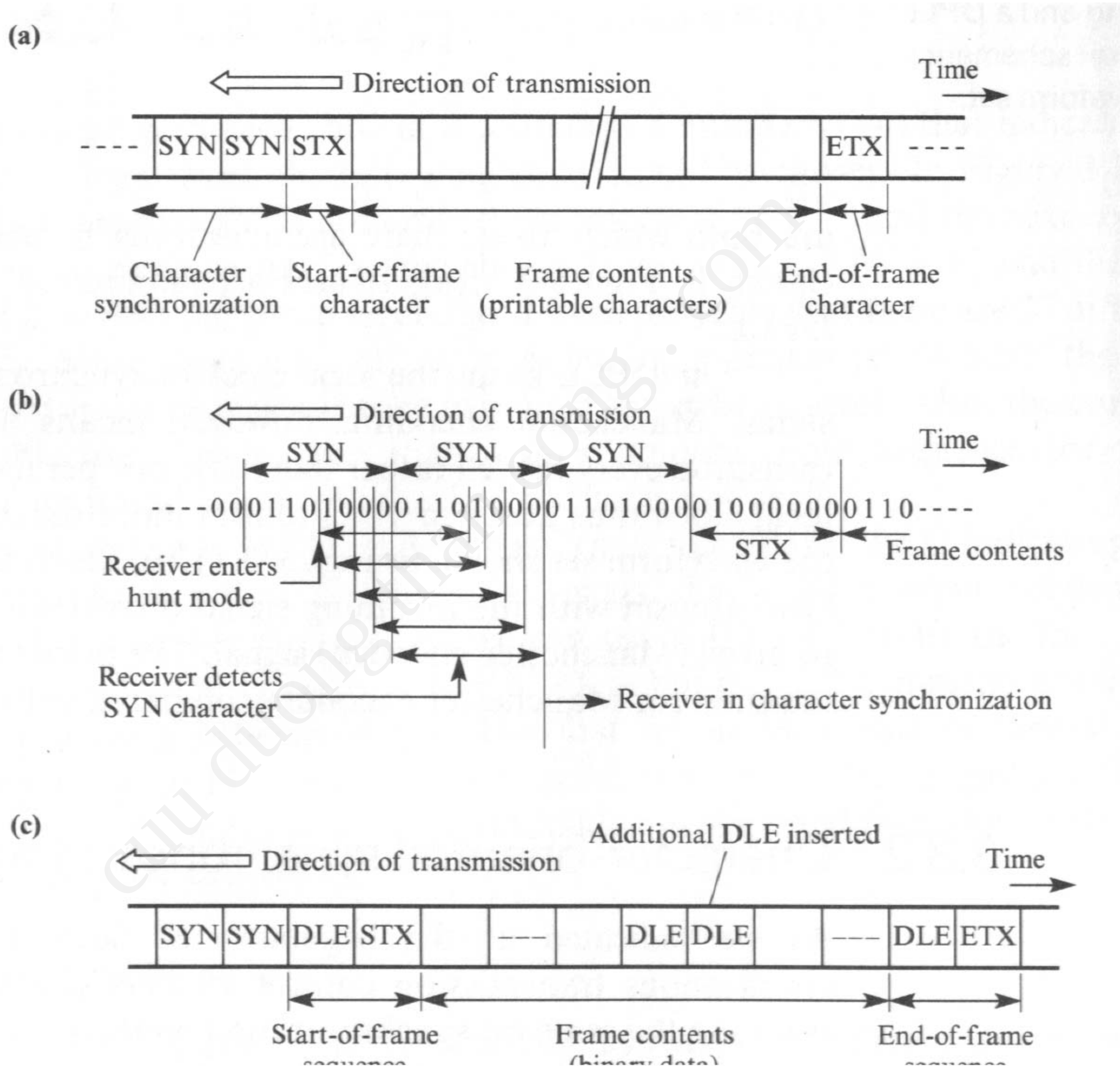
(a)
Received (Manchester-encoded) signal
×16 local clock
DPLL
Manchester decoder
RxD
×2 local synchronized clock
÷2
RxC
(b)
1 0 0 1 1
Received signal
× 2 local (synchronized) clock
0 1 1 0 1 0 0 1 0 1
RxD
RxC

Truyền đồng bộ (Synchronous transmission)

- Đồng bộ khung :
  - Character-oriented :
    - Thường sử dụng khi truyền các khối ký tự.
    - Để thực hiện việc đồng bộ ký tự, bộ phát sẽ truyền trước ít nhất là 2 **ký tự điều khiển** (*control characters*) còn gọi là **ký tự đồng bộ** *SYN* trước khi truyền khối ký tự. Điều này sẽ thực hiện 2 chức năng:
      - Đồng bộ bit: tạo ra các trạng thái chuyển đổi mức tín hiệu trên đường truyền để DPLL thiết lập được sự đồng bộ.
      - Đồng bộ ký tự: cho phép phía thu xác định chính xác vị trí bắt đầu và kết thúc của mỗi ký tự.

Truyền đồng bộ (Synchronous transmission)

- Khi phía thu thực hiện được việc đồng bộ bit, nó sẽ bắt đầu chế độ dò tìm (hunt mode). Trong mode này phía thu sẽ thu và kiểm tra mỗi nhóm 8 bit xem có phải là ký tự đồng bộ (SYN) hay không. Nếu không phải là ký tự SYN, phía thu sẽ thu bit kế tiếp và kiểm tra. Ngược lại nếu đúng là SYN thì phía thu xem như đã thực hiện xong việc đồng bộ ký tự, và sau đó nhận vào 8 bit xem như 1 ký tự.
- Sau khi đã thực hiện xong vấn đề đồng bộ như trên Việc đồng bộ khung được thực hiện giống như kỹ thuật truyền bất đồng bộ.



(a)
Direction of transmission
Time
SYN SYN STX
ETX
Character synchronization
Start-of-frame character
Frame contents (printable characters)
End-of-frame character
(b)
Direction of transmission
Time
SYN
SYN
SYN
----000110100000110100001101000010000000110----
STX
Frame contents
Receiver enters hunt mode
Receiver detects SYN character
Receiver in character synchronization
(c)
Additional DLE inserted
Direction of transmission
Time
SYN SYN DLE STX
DLE DLE
DLE ETX
Start-of-frame sequence
Frame contents (binary data)
End-of-frame sequence

Truyền đồng bộ (Synchronous transmission)

- Nhận xét :
  - Kiểu truyền định hướng ký tự này sử dụng nhiều ký tự điều khiển (STX, ETX, DLE), do đó hiệu suất truyền thấp.
  - Trong kiểu truyền này yêu cầu khối dữ liệu phát phải có chiều dài là bội số của 8 đảm bảo hệ thống xử lý theo từng ký tự (định hướng ký tự). Điều này có thể không được đảm bảo nếu khối ký tự phát là dữ liệu nhị phân bất kỳ.

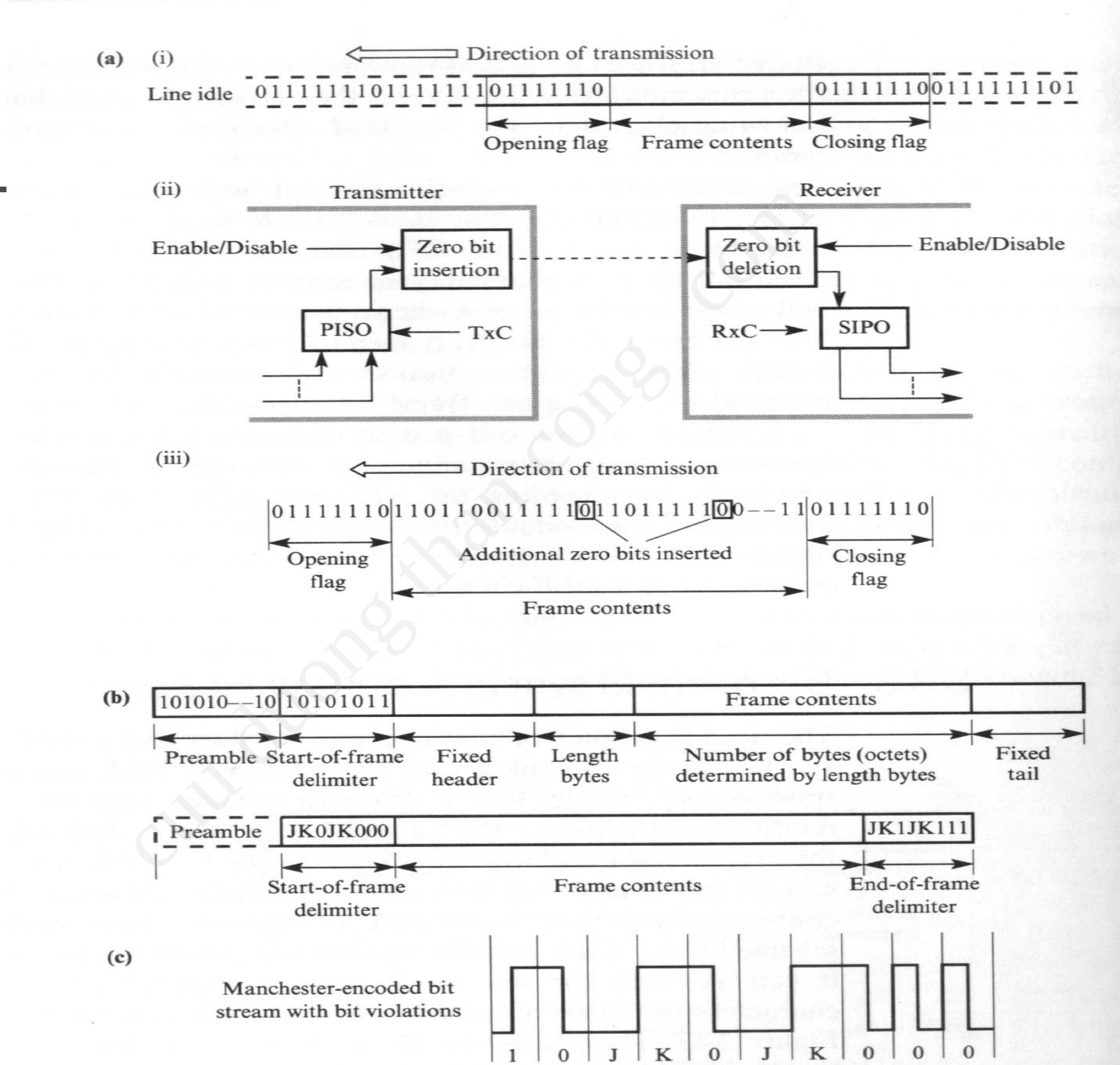
Truyền đồng bộ (Synchronous transmission)

- Bit-oriented : Điểm - điểm
  - Bắt đầu và kết thúc khung (start and end of frame) truyền chuỗi 8 bit 01111110 gọi là cờ (flag pattern).
  - Phía thu sẽthực hiện việc đồng bộ khung bằng cách tìm ký tự (chuỗi bit) cờ này theo nguyên tắc tìm từng bit (bit by bit basic).
  - Khi phía thu nhận được **opening flag** (*cờ bắt đầu*), phía thu sẽ bắt đầu nhận khung dữ liệu cho tới khi phát hiện được **closing flag** (*cờ kết thúc*), khi đó việc thu khung dữ liệu kết thúc.
  - Để thực hiện đồng bộ bit, phía phát sẽ gửi các byte rảnh (idle bytes :11111111) trước cờ khởi đầu của khung.(*Chú ý: Các bit 1 được dùng mã đường dây nên sẽ có sự xáo trộn mức để bên thu thực hiện đồng bộ*
  - Việc trong suốt dữ liệu được thực hiện bằng cách **chèn bit 0** (zero bit insertion) thực hiện tại phía phát. Khi hoạt động, khối này sẽ kiểm tra xem trong chuỗi bit phát có chuỗi liên tiếp 5 bit 1 hay không, nếu có thì sẽ thực hiện chèn 1 bit 0 vào cuối chuỗi này. Khi đó trong chuỗi dữ liệu phát sẽ không thể có ký tự cờ 01111110. Phía thu sẽ thực hiện ngược lại, nếu thu được chuỗi dữ liệu bao gồm 5 bit 1 liên tiếp, sau đó là bit ***0*** thì nó sẽ loại bỏ bit ***0*** này bằng mạch zero bit deletion (mạch xoá bit 0).

Truyền đồng bộ (Synchronous transmission)

- Bit-oriented : Đa điểm
  - Trong cấu hình mạng đa điểm như mạng LAN thì phương pháp đồng bộ Bit Oriented có thể được thực hiện theo nhiều cách khác nhau như :
    - Để tất cả các trạm bám theo đồng bộ thì phát mẫu bit gọi là Preamble : 1010101010
    - Để xác định vị trí bắt đầu và kết thúc một frame thì dùng những mẫu bit như sau (Tùy theo cấu trúc) :
      - Start of frame delimiter : 10101011, hoặc
      - Start of frame :JK0JK000. End of frame : JK1JK100

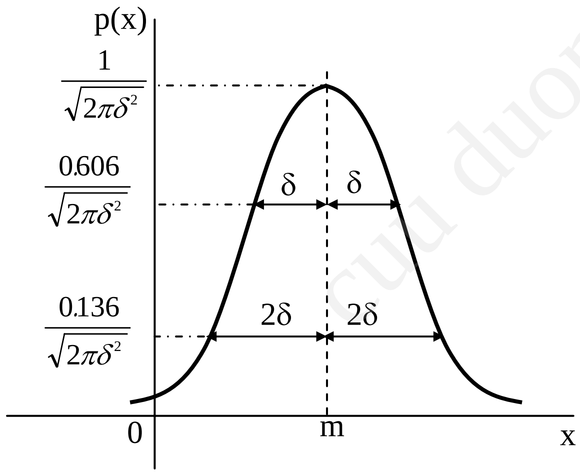
      Trong đó J,K là những mẫu bit được mã hóa không đúng chuẩn với dòng bit truyền thực sự.


(a) (i)
Direction of transmission
Line idle
0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1
Opening flag
Frame contents
Closing flag
(ii)
Transmitter
Receiver
Enable/Disable
Zero bit insertion
PISO
TxC
Zero bit deletion
Enable/Disable
RxC
SIPO
(iii)
Direction of transmission
0 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 – – 1 1 0 1 1 1 1 1 1 0
Opening flag
Additional zero bits inserted
Closing flag
Frame contents
(b)
101010––10
10101011
Frame contents
Preamble
Start-of-frame delimiter
Fixed header
Length bytes
Number of bytes (octets) determined by length bytes
Fixed tail
Preamble
JK0JK000
JK1JK111
Start-of-frame delimiter
Frame contents
End-of-frame delimiter
(c)
Manchester-encoded bit stream with bit violations
1 0 J K 0 J K 0 0 0

3.6 Nhiễu gauss và tỷ lệ lỗi bit

- Nhiễu Gauss :
  - Hàm mật độ công suất của nhiễu Gauss :

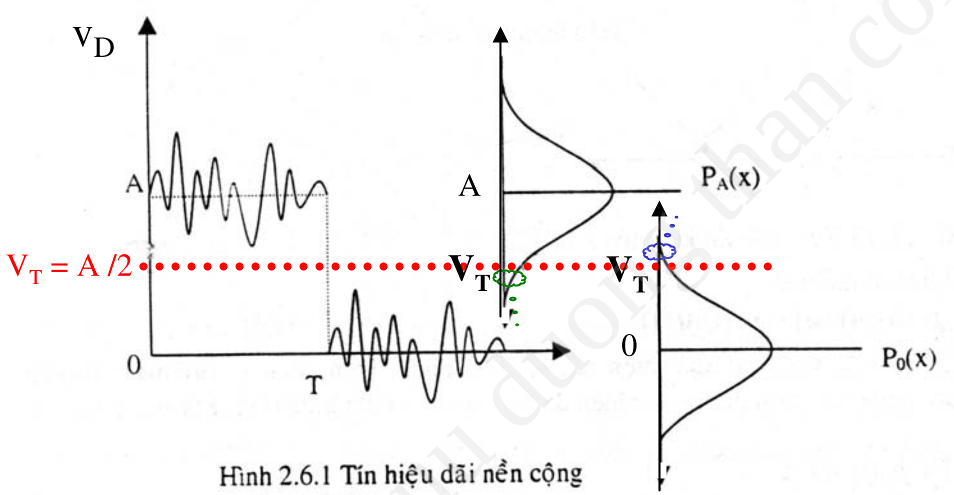
$$p(x) = \frac{1}{\sqrt{2\pi\delta^2}} e^{-(x-m)^2/2\delta^2}$$



- m: giá trị trung bình (DC).
- δ : Độ lệch chuẩn (áp hiệu dụng)
- δ^2 : gọi là phương sai (công suất nhiễu)

Truyền đồng bộ (Synchronous transmission)

- Xét tín hiệu truyền là dãi nền, và chỉ chịu tác động của nhiễu Gauss bằng cách cộng trực tiếp vào tín hiệu, có dạng như sau :



Hình 2.6.1 Tín hiệu dãi nền cộng nhiễu AWGN

Nhiễu Gauss có thể biểu diễn qua hàm mật dộ công suất pdf (power density function)

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-m)^2/2\sigma^2} \quad (2.1)$$

V_T: ngưỡng xác quyết.

$v_D > v_T$: Xác quyết mức ‘1’

$v_D < v_T$: Xác quyết mức ‘0’

Nếu δ càng lớn thì xác suất xác quyết nhầm càng cao.

Truyền đồng bộ (Synchronous transmission)

- Tín hiệu nhận được cộng luôn cả nhiễu nếu lớn hơn V_T thì xác quyết mức '1', ngược lại nhỏ hơn V_T thì xác quyết mức '0'. Do đó ta có :
  - Xác xuất lỗi khi truyền bit 1 sai là:

$$P_r(v_D < v_T) = p(0/1) = \int_{-\infty}^{v_T} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-A)^2}{2\sigma^2}} dx$$

  - Xác xuất lỗi khi truyền bit 0 sai là:

$$P_r(v_D > v_T) = p(1/0) = \int_{v_T}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} dx$$

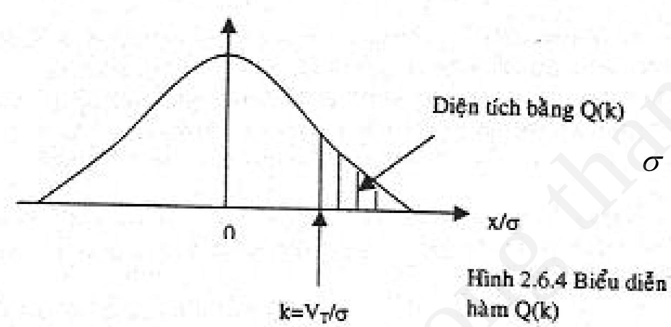
  - Giả sử xác suất xuất hiện bit 1 và 0 là $p_r(1)$ và $p_r(0)$
- Xác suất lỗi 1 bit :

$p_e = p_r(1)p(0/1) + p_r(0)p(1/0)$

Nếu xác suất xuất hiện 0 và 1 là như nhau tức $p_r(0) = p_r(1) = 0.5$, thì $p_e = 0.5\, p(0/1) + 0.5\, p(1/0) = p(0/1) = p(1/0)$.

Truyền đồng bộ (Synchronous transmission)

- Để tính p(0/1) hay p(1/0) thì dựa vào hàm Q(k).
- Tính chất hàm Q(k)



Hình 2.6.4 Biểu diễn ý nghĩa hàm Q(k)

- Hàm Q(k) thực chất là hàm phân bố chuẩn với m = 0,σ = 1.

$$Q(k) \approx \frac{e^{-\frac{k^2}{2}}}{\sqrt{2\pi}k}$$

- Khi dùng hàm Q(k) để tính xác suất thì cần chuẩn hóa giá trị ngưỡng. Trong trường hợp này $v_T = A/2$ nên $k = v_T/\sigma$

  -> $p_e = p(0/1) = p(1/0) = Q(v_T/\sigma) = Q(A/2\sigma)$

Truyền đồng bộ (Synchronous transmission)

- Ngoài ra, xác suất lỗi có thể tính dựa vào $(S/N)_v$, hoặc $(S/N)_p$
  - $P_e = Q\ (A/2\sigma) = Q\ [(S/N)_V]$
  - $P_e = Q\ (A/2\sigma) = Q\ [(S/N)_P]$
- Xác suất sai k bit bất kỳ khi truyền khối n bit

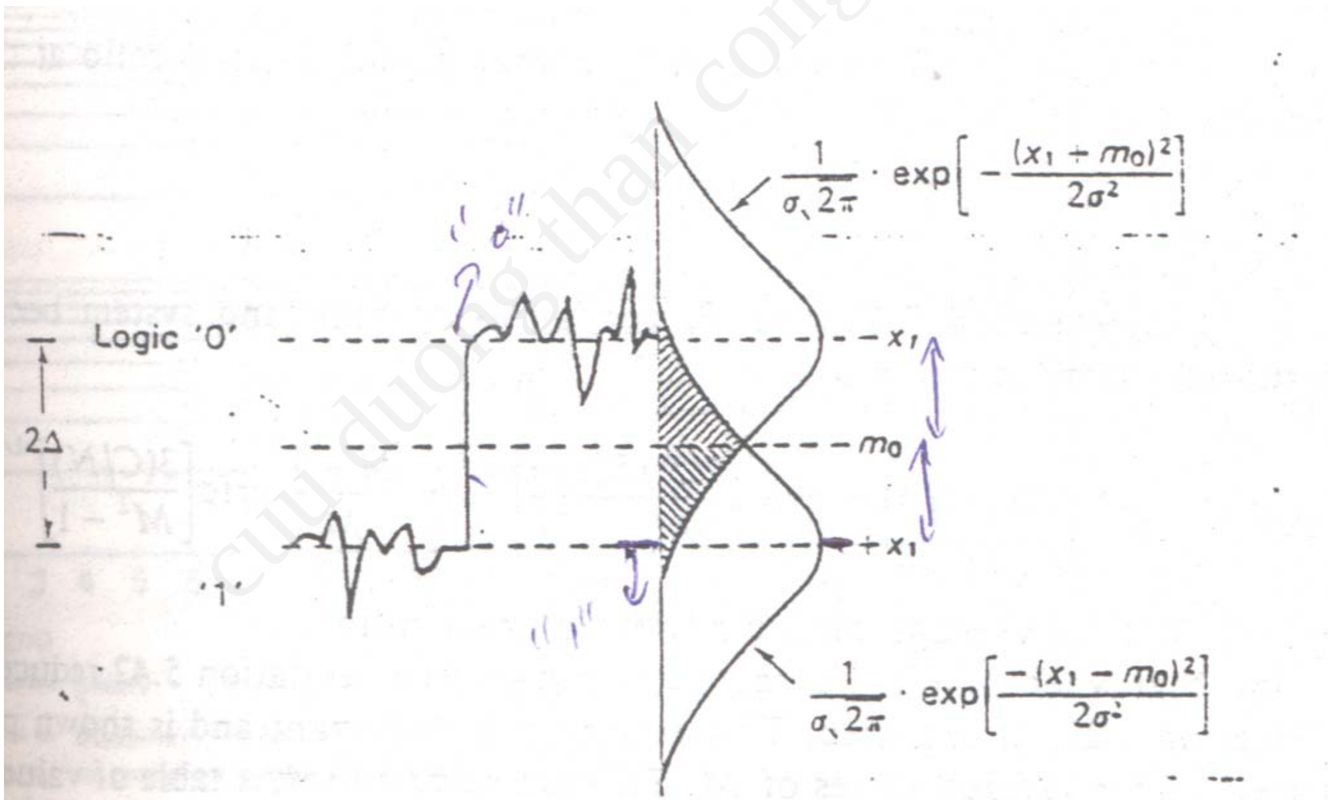
$$p_k = C_n^k p_e^k (1 - p_e^k)^{n-k}$$

- Nếu truyền n bits mà toàn bộ bị sai (k=0).

$p_r(\text{error}) = 1\text{-}p_0 = 1\text{-}(1\text{-}p_e)^n \approx np_e$. (do $p_e << 1$)

Truyền đồng bộ (Synchronous transmission)

- Trường hợp tổng quát $v_T = m_o$



Truyền đồng bộ (Synchronous transmission)

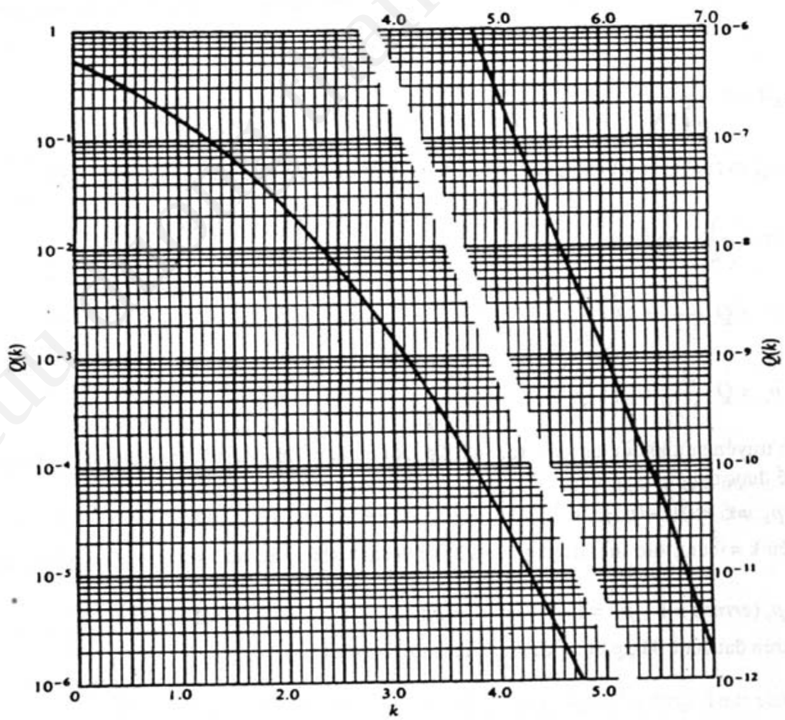
- Đồ thị tính hàm Q(k)

666 TABLES

Numerical values of $Q(k)$ are plotted below for $0 \le k \le 7.0$. For larger values of k, $Q(k)$ may be approximated by

$$Q(k) \approx \frac{1}{\sqrt{2\pi}\,k} e^{-k^2/2}$$

which is quite accurate for $k > 3$.



Hình 2.6.5 Đồ thị hàm Q(k)

3.7 Các phương pháp phát hiện lỗi :

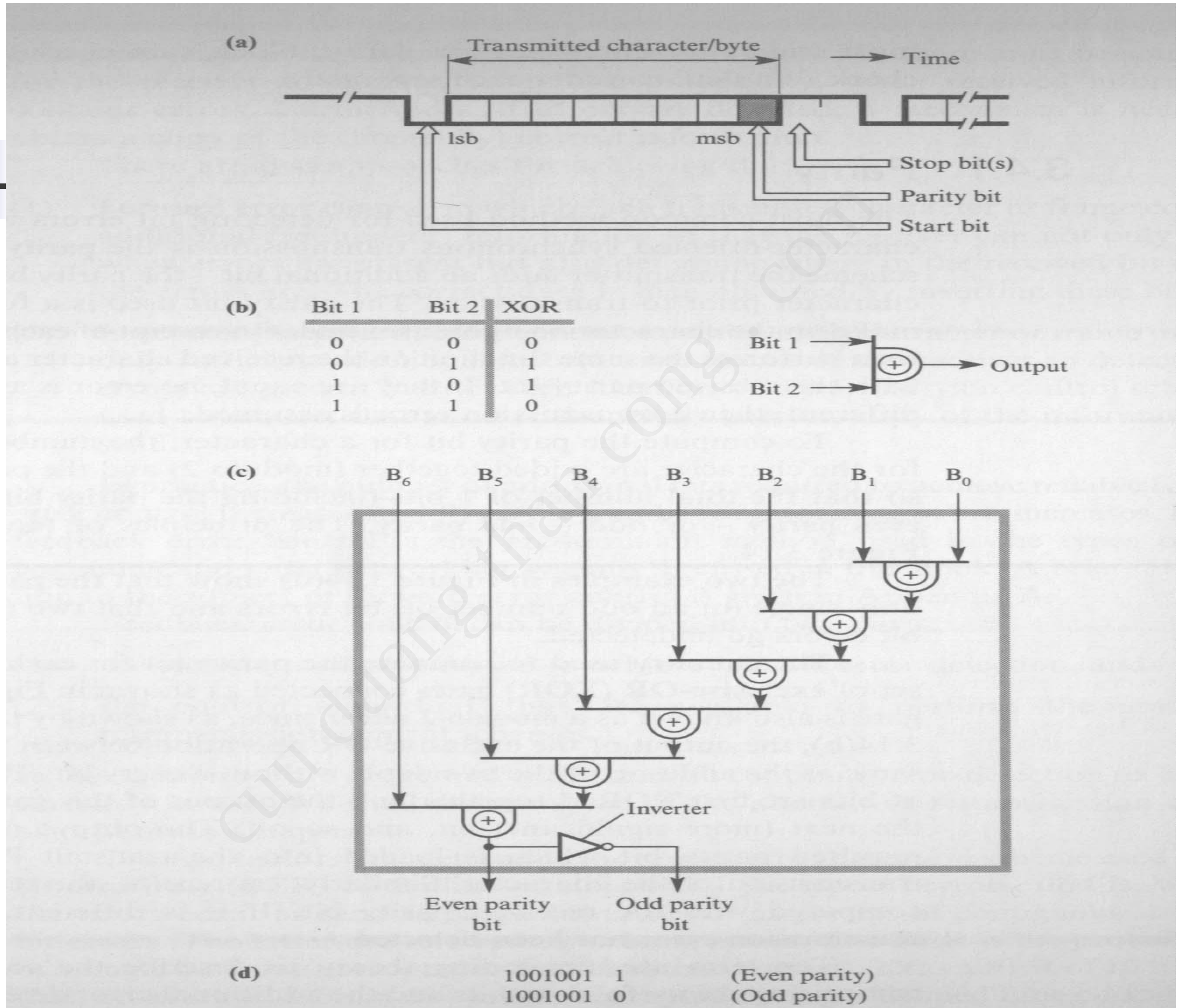
- Có 2 phương pháp phát hiện lỗi
  - **Forward error control** : Mỗi ký tự hoặc khung khi truyền chứa thêm thông tin bổ sung để phía thu khi nhận dựa vào thông tin này phát hiện dữ liệu nhận được có bị sai hay không, nếu sai thì phía thu tiến hành sửa (nếu có thể). Thường sử dụng cho các đường truyền rất xa có thời gian trễn do lan truyền lớn.
  - Feedback (backward) error control : Mỗi ký tự hoặc khung khi truyền chứa thêm thông tin bổ sung để phía thu khi nhận dựa vào thông tin này phát hiện dữ liệu nhận được có bị sai hay không, chứ không tiến hành sửa. Nếu sửa sai thì yêu cầu bên gửi phát lại, phương pháp này thường được sử dụng trong các hệ thống truyền thông.

Các phương pháp phát hiện lỗi

- **Forward error control** :
  - Parity check
  - Block sum check
  - Cyclic Redundant Check
  - Hamming

Các phương pháp phát hiện lỗi

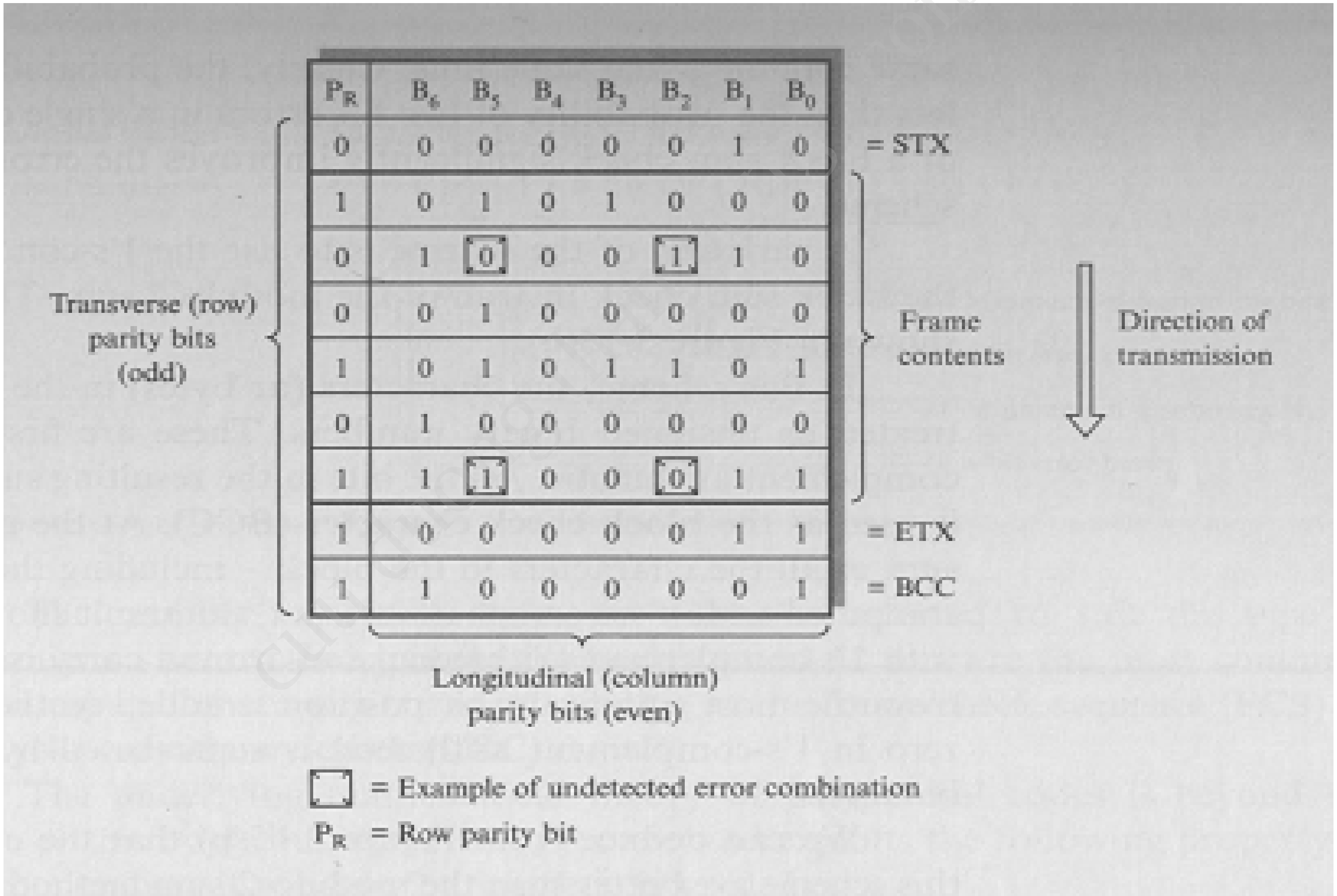
- Parity check :
  - Trong mỗi ký tự (7 hoặc 8 bit) truyền đi khối kiểm tra sẽ thực hiện việc chèn một bit (*parity bit*) vào cuối ký tự (*ngay trước stop bit*). Giá trị parity bit là 0 hay 1 tuỳ vào phương pháp kiểm tra là kiểm tra chẵn (*even parity*) hay kiểm tra lẻ (*odd parity*).
    - *Kiểm tra chẵn :* Tổng số bit 1 trong tất cả bit dữ liệu (không kể start và stop bit) và parity bit là số chẵn
    - *Kiểm tra lẻ*: Tổng số bit 1 trong tất cả bit dữ liệu (*không kể start và stop bit*) và parity bit là1 số lẻ.
  - Phía thu sẽ thực hiện việc tính lại parity bit sau đó so sánh parity bit nhận được, nếu khác nhau thì phía thu sẽ hiểu rằng đã có lỗi xảy ra trên đường truyền.
    - Phát hiện sai nếu tổng số bit sai là số lẻ
    - Không phát hiện sai nếu tổng số bit sai là số lẻ.


(a)
Transmitted character/byte
Time
lsb
msb
Stop bit(s)
Parity bit
Start bit
(b)
Bit 1
Bit 2
XOR
0 0 0
0 1 1
1 0 1
1 1 0
Bit 1
Bit 2
Output
(c)
B_6
B_5
B_4
B_3
B_2
B_1
B_0
Inverter
Even parity bit
Odd parity bit
(d)
1001001 1 (Even parity)
1001001 0 (Odd parity)

Các phương pháp phát hiện lỗi

- Block sum check :
  - Sử dụng khi truyền dữ liệu dưới dạng một khối các ký tự, trong kiểu kiểm tra này, mỗi ký tự truyền đi sẽ được phân phối 2 bit kiểm tra parity là *parity hàng* và *parity cột*. Các bit parity theo từng cột được gọi là ký tự kiểm tra khối BCC- *block check character*.
    - Phát hiện và sửa sai nếu lỗi bit đơn.
    - Không phát hiện sai nếu các bit sai kiểu chùm như : sai 4 bit, 2 bit cùng hàng và 2 bit cùng cột.
    - Các trường hợp còn lại thì phát hiện sai được
  - Thường sử dụng trong kiểu truyền bất đồng bộ.

Các phương pháp phát hiện lỗi



Các phương pháp phát hiện lỗi

- Cyclic Redundant Check :
  - Phía phát tạo ra một ký số kiểm tra khung FSC (*frame sequence check*) hay CRC, FSC được phát kèm theo phía sau của frame thông tin.
  - Phát hiện được tất cả lỗi bít đơn, bit đôi,bit lẻ hay bit chùm.
  - Thường sử dụng trong kỹ thuật truyền đồng bộ.
  - Giả sử gọi :
    - M(x) : bản tin cần truyền đi (the message to be transmitted)gồm kbit.
    - G(x) : đa thức sinh (the divisor or generator) gồm n+1bit
    - R(x) : số dư gồm n bit (k > n)
    - Q(x) : thương số của phép chia
    - T(x) : thông điệp truyền đi gồn (n+k) bit

Các phương pháp phát hiện lỗi

- Bên phát :
  - *Bước 1* **:Chuyển thông điệp nhị phân M thành đa thức M(x).**
    - **Nhân đa thức M(x) với x^n, tương đương với chuỗi bit nhị phân được dịch sang trái n bit.**
  - ***Bước 2*: Thực hiện phép chia** $$\frac{M(x).x^n}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

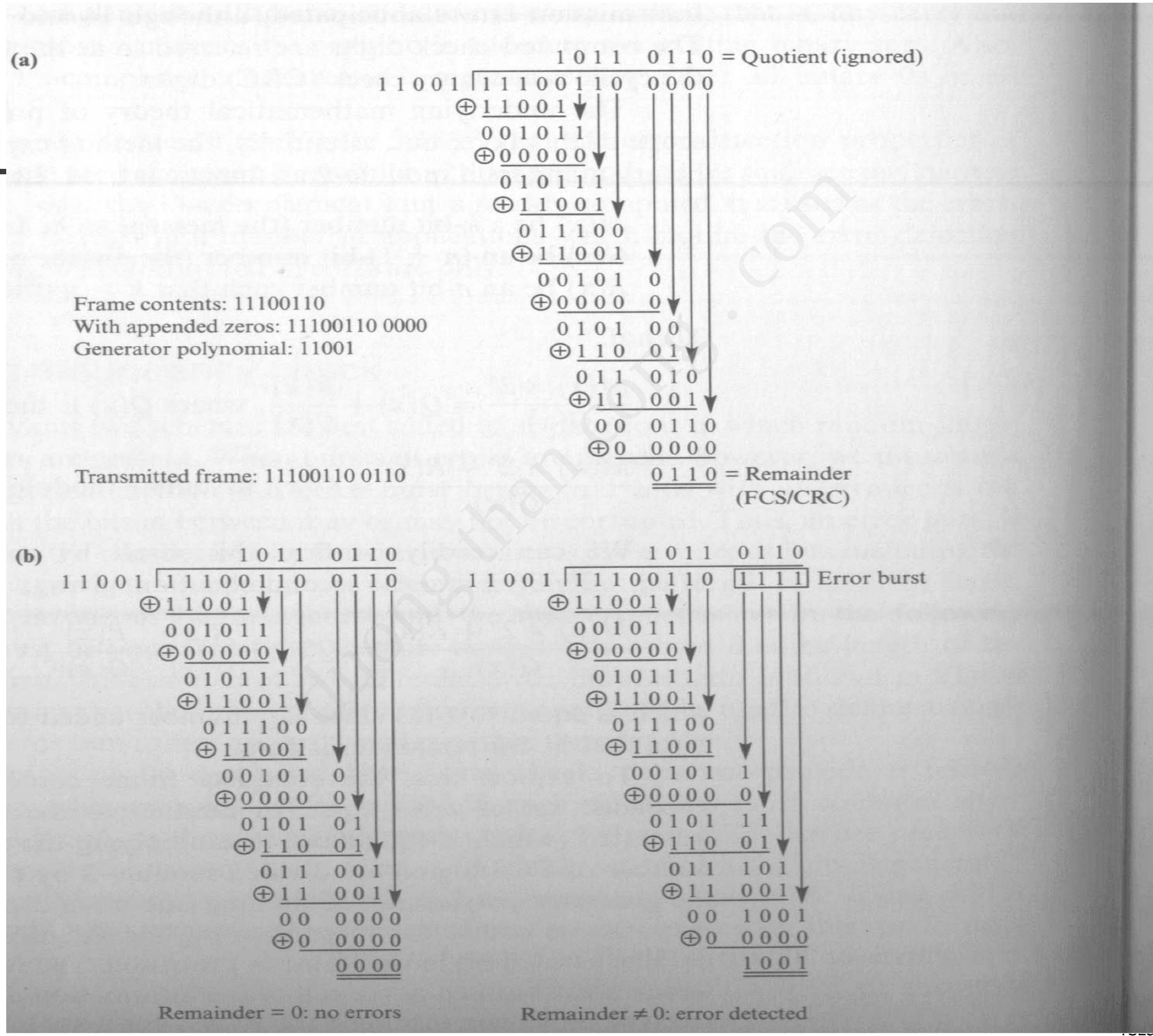
    Điều này được thực hiện theo đa thức hoặc theo modulo 2.
  - ***Bước 3:* Thông điệp cần truyền đi là T(x):**

    $T(x) = x^n \,.\, M(x) + R(x)$

    Điều này được thực hiện theo đa thức hoặc thêm n bit của số dư (R(x)) trong phép chia ở Bước 2 vào sau k bit của bản tin cần truyền
- **Bên thu :**
  - **Việc phát hiện lỗi được thực hiện bằng cách lấy chuỗi dữ liệu thu được chia modulo 2 cho đa thức sinh G(x) như sau:**

    $$\frac{T(x)}{G(x)} = \frac{x^n M(x) + R(x)}{G(x)} = \frac{x^n M(x)}{G(x)} + \frac{R(x)}{G(x)} = Q(x) + \underbrace{\frac{R(x)}{G(x)} + \frac{R(x)}{G(x)}}_{=0} = Q(x)$$

    - ***Do trong phép cộng modulo 2 thì 2 số giống nhau cộng lại bằng 0***
  - **Như vậy nếu phần dư trong phép chia này bằng 0 thì phía thu xem như không có lỗi xảy ra, ngược lại nếu khác 0 thì phía thu phát hiện được lỗi xảy ra khi truyền dữ liệu.**


(a)
1011 0110 = Quotient (ignored)
Frame contents: 11100110
With appended zeros: 11100110 0000
Generator polynomial: 11001
Transmitted frame: 11100110 0110
0110 = Remainder (FCS/CRC)
(b)
Error burst
Remainder = 0: no errors
Remainder ≠ 0: error detected

Các phương pháp phát hiện lỗi

- Hamming :
  - Khoảng cách mã (khảong cách Hamming) là số bit khác nhau giữa 2 từ mã.
  - Mã Hamming là bộ mã mà thêm một số bit kiểm tra ở một số vị trí nhất định trong thông tin cần truyền để tạo thành mã Hamming.
  - Bên thu không chỉ phát hiện được sai mà còn có thể sửa sai ở một số vị trí nhất định
  - Ví dụ :
    - Để mã hóa các số thập phân từ 0 – F ta cần 4 bit. Gọi 4 bit đó là $m_3m_2m_1m_0$.
    - Trước khi truyền ta chèn vào 3 bit kiểm tra $c_2c_1c_0$., 3 bit này được tính bằng cách EX-OR :
      - $c_0 = m_0 + m_1 + m_3$
      - $c_1 = m_0 + m_2 + m_3$
      - $c_2 = m_1 + m_2 + m_3$
    - Đầu thu thực hiện kiểm tra bằng cách tính :
      - $p_0 = c_0 + m_0 + m_1 + m_3$
      - $p_1 = c_1 + m_0 + m_2 + m_3$
      - $p_2 = c_2 + m_1 + m_2 + m_3$

      + Nếu không sai thì $p_0 = p_1 = p_2 = 0$

      + Nếu sai thì thì số nhị phân $p_2p_1p_0$ là vị trí của bit sai (các bit trong từ mã Hamming truyền đi được đánh thứ tự từ 0 đến 7) và sửa bằng cách đảo bit này.

Hamming

Calculating the Hamming Code

- The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
2. All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.
   - Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
   - Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
   - Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
   - Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
   - Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
   - Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)

   etc.
4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.
5. -> Then the receiver could calculate which bit was wrong and correct it. The method is to verify each check bit. Write down all the incorrect parity bits. Doing so, you will discover that parity bits 2 and 8 are incorrect. It is not an accident that 2 + 8 = 10, and that bit position 10 is the location of the bad bit. In general, check each parity bit, and add the positions that are wrong, this will give you the location of the bad bit.

Here is an example:

- A byte of data: 10011010
  Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0
  Calculate the parity for each parity bit (a ? represents the bit position being set):
- - Position 1 checks bits 1,3,5,7,9,11:
  ? _ **1** _ **0** 0 **1** _ **1** 0 **1** 0. Even parity so set position 1 to a 0: **0** _ **1** _ **0** 0 **1** _ **1** 0 **1** 0
- - Position 2 checks bits 2,3,6,7,10,11:
  0 **? 1** _ 0 **0 1** _ 1 **0 1** 0. Odd parity so set position 2 to a 1: 0 **1 1** _ 0 **0 1** _ 1 **0 1** 0
- - Position 4 checks bits 4,5,6,7,12:
  0 1 1 **? 0 0 1** _ 1 0 1 **0**. Odd parity so set position 4 to a 1: 0 1 1 **1 0 0 1** _ 1 0 1 **0**
- - Position 8 checks bits 8,9,10,11,12:
  0 1 1 1 0 0 1 **? 1 0 1 0**. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 **0 1 0 1 0**
- -> Code word: 011100101010.
- The above example created a code word of 011100101010. Suppose the word that was received was 011100101110 instead. Then the receiver could calculate which bit was wrong and correct it. The method is to verify each check bit. Write down all the incorrect parity bits. Doing so, you will discover that parity bits 2 and 8 are incorrect. It is not an accident that 2 + 8 = 10, and that bit position 10 is the location of the bad bit. In general, check each parity bit, and add the positions that are wrong, this will give you the location of the bad bit.
- **Try one yourself**
- Test if these code words are correct, assuming they were created using an even parity Hamming Code . If one is incorrect, indicate what the correct code word should have been. Also, indicate what the original data was.
  010101100011
  111110001100
  000010001010

3.8 Kỹ thuật nén dữ liệu

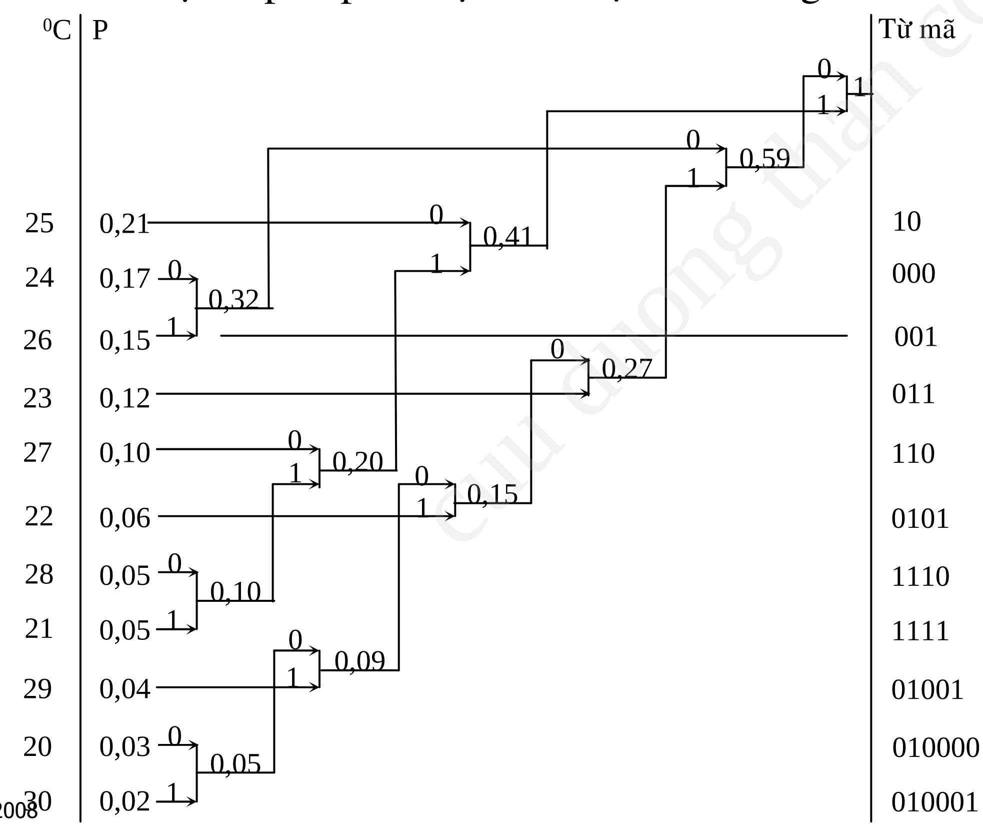
- Mục đích : Giảm kích thước và thời gian truyền.
- Các kỹ thuật nén cơ bản :
  - **Packed Decimal** : Khi truyền ký tự số dùng mã BCD 4 bit thay cho mã ASCII 7bits hay EDBIC
  - **Relative Coding** : Khi truyền các ký tự số, chỉ truyền sai số giữa các số liên tiếp nhau.
  - **Character Suppression :** Khi truyền các ký tự in được mà các ký tự giống nhau được truyền liên tiếp, thay vì truyền hết các ký tự thì chỉ truyền 1 ký tự đại diên và kèm theo là số các ký tự giống nhau.
  - **Huffman Coding**
  - **Run Length Coding (Facsimile compression)**

Kỹ thuật nén dữ liệu

- **Huffman Coding**
  - Là một mã thống kê tối ưu
  - Các tin xuất hiện nhiều, xác suất xuất hiện lớn thì được mã hóa bằng từ mã ngắn và ngược lại. Do đó độ dài trung bình của các từ mã sẽ nhỏ nhất, làm giảm thiểu rất nhiều lượng thông tin truyền trên đường dây nên giảm sai số.

Kỹ thuật nén dữ liệu

- **Huffman Coding**
  - Ví dụ : Để tạo mã cho việc đo nhiệt độ từ 20^0 đến 30^0 C người ta lấy xác suất của nó và được sắp xếp thứ tự xuất hiện như bảng.



- Sắp xếp các khả năng xuất hiện theo thứ tự giảm dần.
- Hai giá trị 0,1 gán cho 2 khả năng xuất hiện nhỏ nhất, 2 khả năng này gộp lại thành 1 và sắp xếp theo thứ tự giảm dần. Tương tự như vậy cho đến 2 khả năng cuối cùng (tổng sẽ = 1).
- Mã tương ứng của mỗi nhiệt độ được hình thành bằng cách chọn các bit 0,1 trên đường đi xuất phát từ mức nhiệt độ đến ngọn.
- Bit LSM sẽ nằm bên trái cây.

Kỹ thuật nén dữ liệu

- **Huffman Coding**
  - Hiệu suất sử dụng từ mã (entropy)

    $H(x)=\sum p_i \log_2 (1/p_i)$ (bits/symbol) = 3,14
  - Chiều dài trung bình của từ mã.

    $N = \sum p_i N_i$ (bits/symbol) = 3,18 ;4
  - Hiệu suất của mã hóa

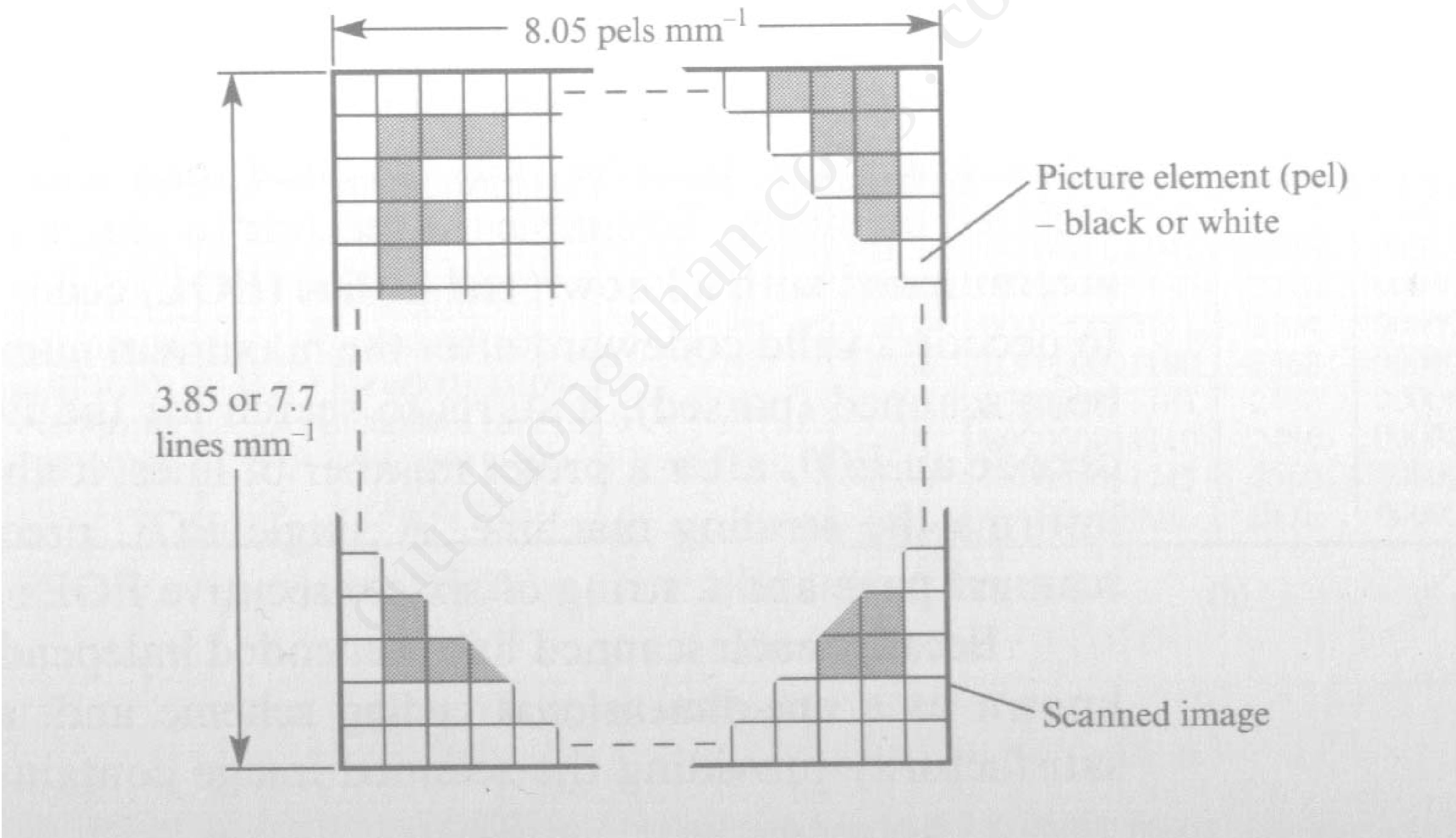
    $h = H(x)/N$ = 98% ;78%
  - Tốc độ bit nhị phân

    $R= rN$

Kỹ thuật nén dữ liệu

- **Run Length Coding (Facsimile compression)**
  - Sử dụng trong máy Facsimile trắng đen.
  - Một trang Fax được chia
    - Theo chiều dọc khoảng khoảng 3.85 hoặc 7.7 lines/mm, tương đương 100 hoặc 200 lines / inche
    - Mỗi line được số hoá với tốc độ 8.05 phần tử ảnh (pels)/mm.
    - Mỗi điểm ảnh trắng mã hoá '0', điểm đen mã hoá '1'
  - Một trang Fax khi chưa nén được mã hóa khoảng 2 triệu bits.

Kỹ thuật nén dữ liệu



Kỹ thuật nén dữ liệu

- **Run Length Coding (Facsimile compression)**
  - Thực tế khi truyền bức Fax thì sẽ có những line mà có khoảng điểm ảnh trắng hay đen liên tục, để giảm bớt số bit trước khi truyền ta dùng phương pháp nén Facsimile:
    - Các từ mã cố định và chia thành 2 nhóm the termination-codes (Hình a: Chiều dài từ 0 – 63 pixel) và the make-up codes (Hình b: Chiều dài là k x 64 pixel).
    - Để bên nhận đồng bộ thì ký mã EOL(End Of Line) được thêm vào ở cuối mỗi line.
    - Kết thúc trang là chuỗi 6 EOL liện tiếp.
    - Trong trường hợp bên thu không giải mã được EOL thì sẽ ngưng quá trình nhận và thông báo cho bên phát biết.

White run length	Code-word	Black run length	Code-word
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	00001100111
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000011001010
27	0100100	27	000011001011
28	0011000	28	000011001100
29	00000010	29	000011001101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	0010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011010011
36	00010101	36	000011010100
37	00010110	37	000011010101
38	00010111	38	000011010110
39	00101000	39	000011010111
40	00101001	40	000001101100
41	00101011	41	000001101101
42	00101011	42	000011011010
43	00101100	43	000011011011
44	00101101	44	000001010100
45	00000100	45	000001010101
46	00000101	46	000001010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101
50	01010011	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111
54	00100101	54	000000111000
55	01011000	55	000000100111

(a)

White run length	Code-word	Black run length	Code-word
56	01011001	56	000000101000
57	01011010	57	000001011000
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

(a) cont.

White run length	Code-word	Black run length	Code-word
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	0000001101100
576	01101000	576	0000001101101
640	01100111	640	0000001001010
704	011001100	704	0000001001011
768	011001101	768	0000001001100
832	011010010	832	0000001001101
896	011010011	896	0000001110010
960	011010100	960	0000001110011
1024	011010101	1024	0000001110100
1088	011010110	1088	0000001110101
1152	011010111	1152	0000001110110
1216	011011000	1216	0000001110111
1280	011011001	1280	0000001010010
1344	011011010	1344	0000001010011
1408	011011011	1408	0000001010100
1472	010011000	1472	0000001010101
1536	010011001	1536	0000001011010
1600	010011010	1600	0000001011011
1664	011000	1664	0000001100100
1728	010011011	1728	0000001100101
1792	00000001000	1792	00000001000
1856	00000001100	1856	00000001100
1920	00000001101	1920	00000001101
1984	000000010010	1984	000000010010
2048	000000010011	2048	000000010011
2112	000000010100	2112	000000010100
2176	000000010101	2176	000000010101
2240	000000010110	2240	000000010110
2304	000000010111	2304	000000010111
2368	000000011100	2368	000000011100
2432	000000011101	2432	000000011101
2496	000000011110	2496	000000011110
2560	000000011111	2560	000000011111
EOL	00000000001	EOL	00000000001

(b)

Transmission Control Circuits

- Universal asynchronous receiver transmitter (UART)
  - Start and stop bit insertion and deletion
  - Bit (clock synchronization)
  - Character synchronization
  - Parity bit generation and checking per character (BCC computed by controlling device)

Transmission Control Circuits

- Universal synchronous receiver transmitter
  - Low bit rate DPLL clock synchronization
  - Character synchronization
  - Synchronous idle character generation
  - Parity generation and checking per character (BCC computed by controlling device)

Transmission Control Circuits

- Universal synchronous / asynchronous receiver transmitter (USART)
  - Can be programmed to operate as either a UART or USRT
  - Has all programmable features of both devices

Transmission Control Circuits

- Bit-oriented protocol circuits (BOPs)
  - Opening and closing flag insertion and deletion
  - Zero bit insertion and deletion
  - CRC generation and checking
  - Idle pattern generation

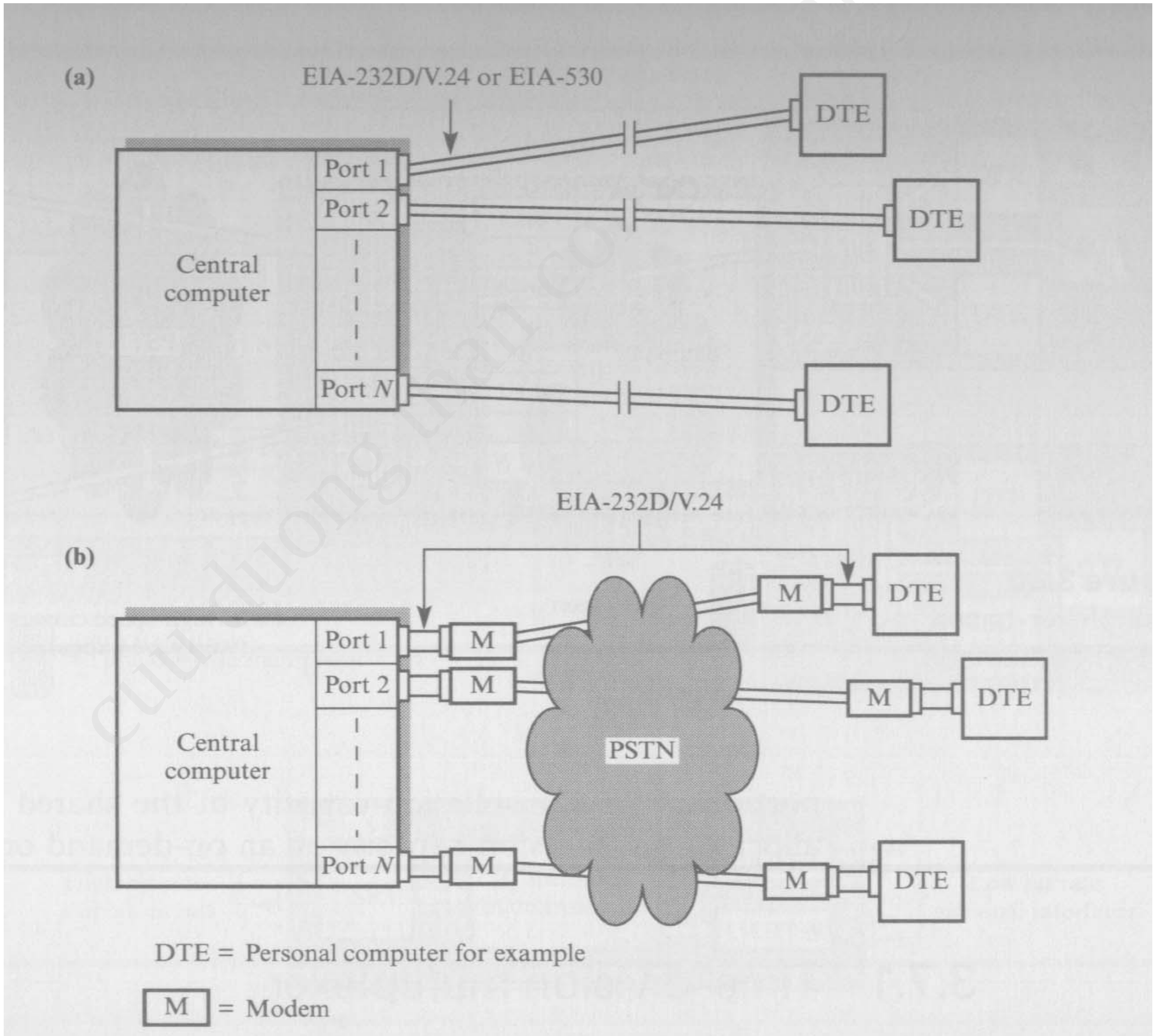
Transmission Control Circuits

- Universal communication control circuits
  - Can be programmed to operate either UART, a USRT or a BOP
  - Has all the programmable features of each circuit

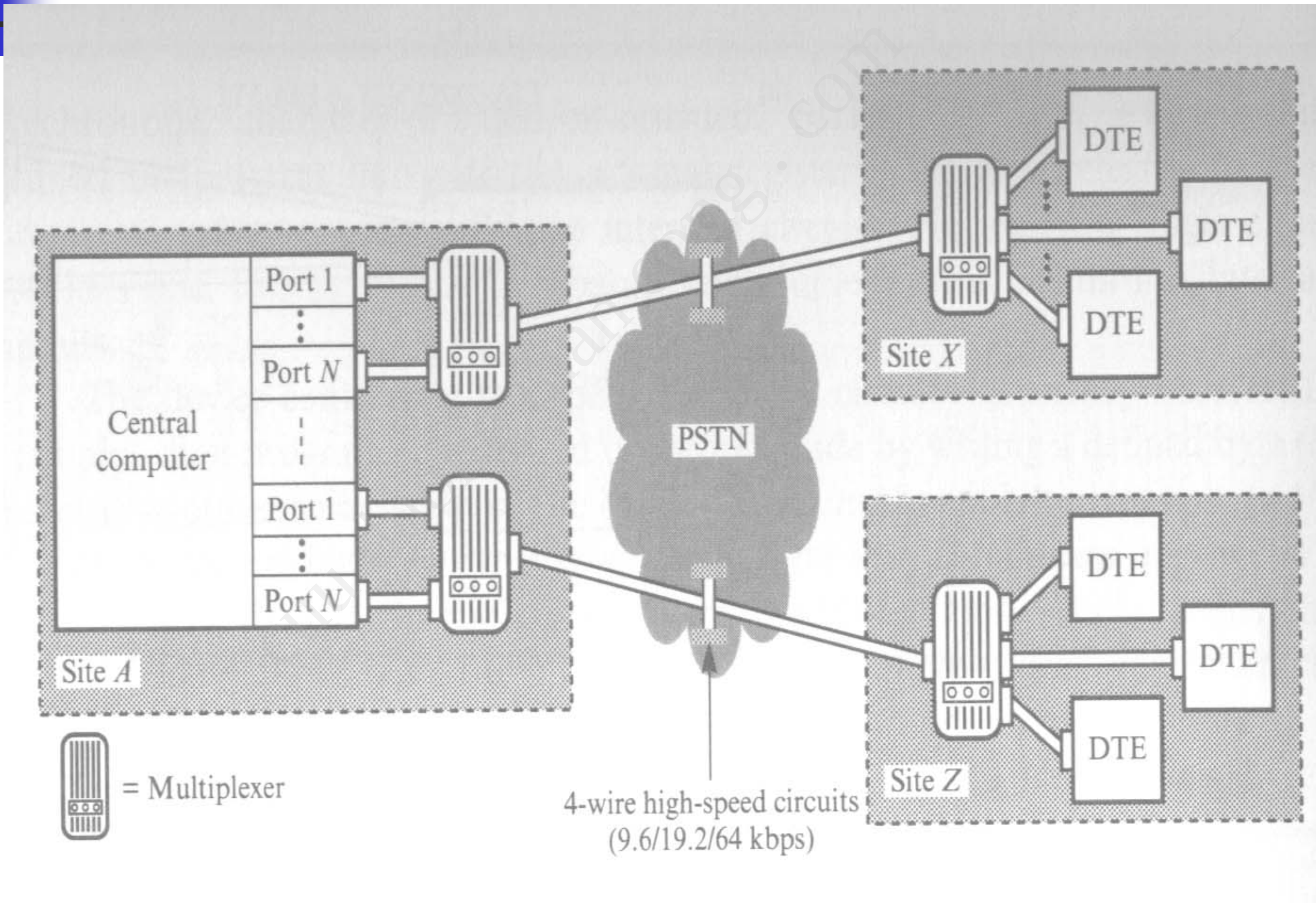
Communications Control Devices

- Simple terminal networks
- Multiplexer-based networks

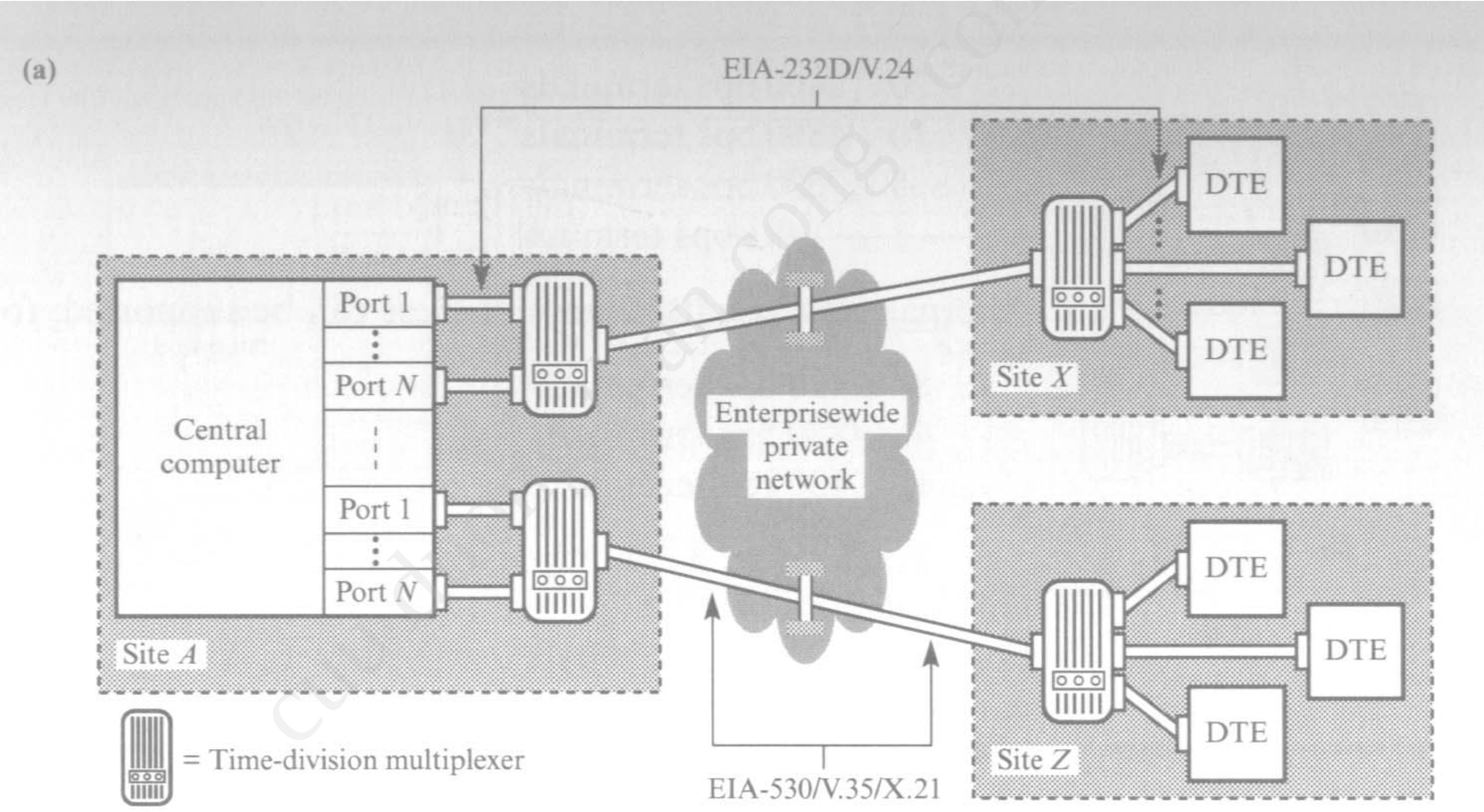
Communications Control Devices



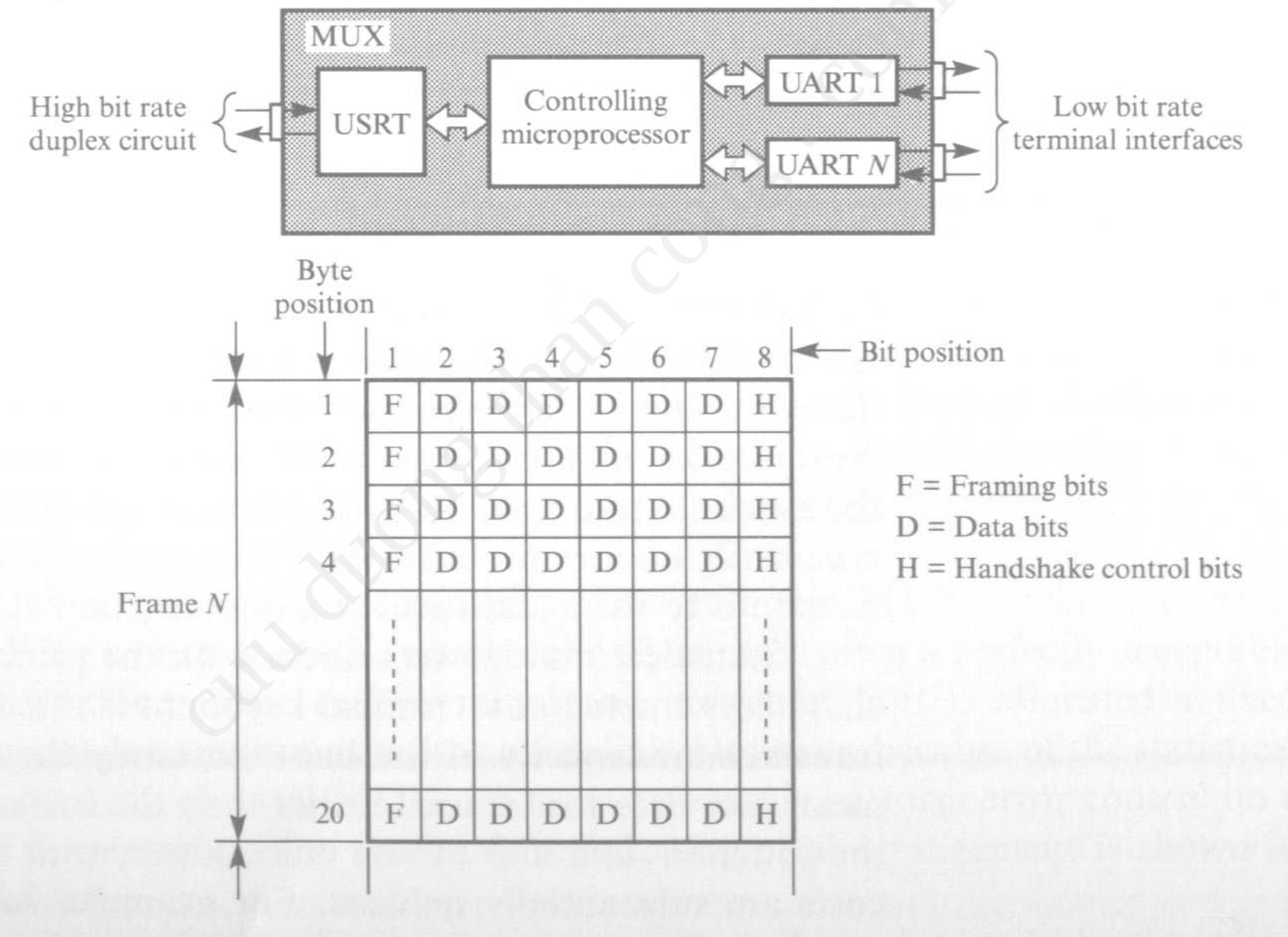
Communications Control Devices



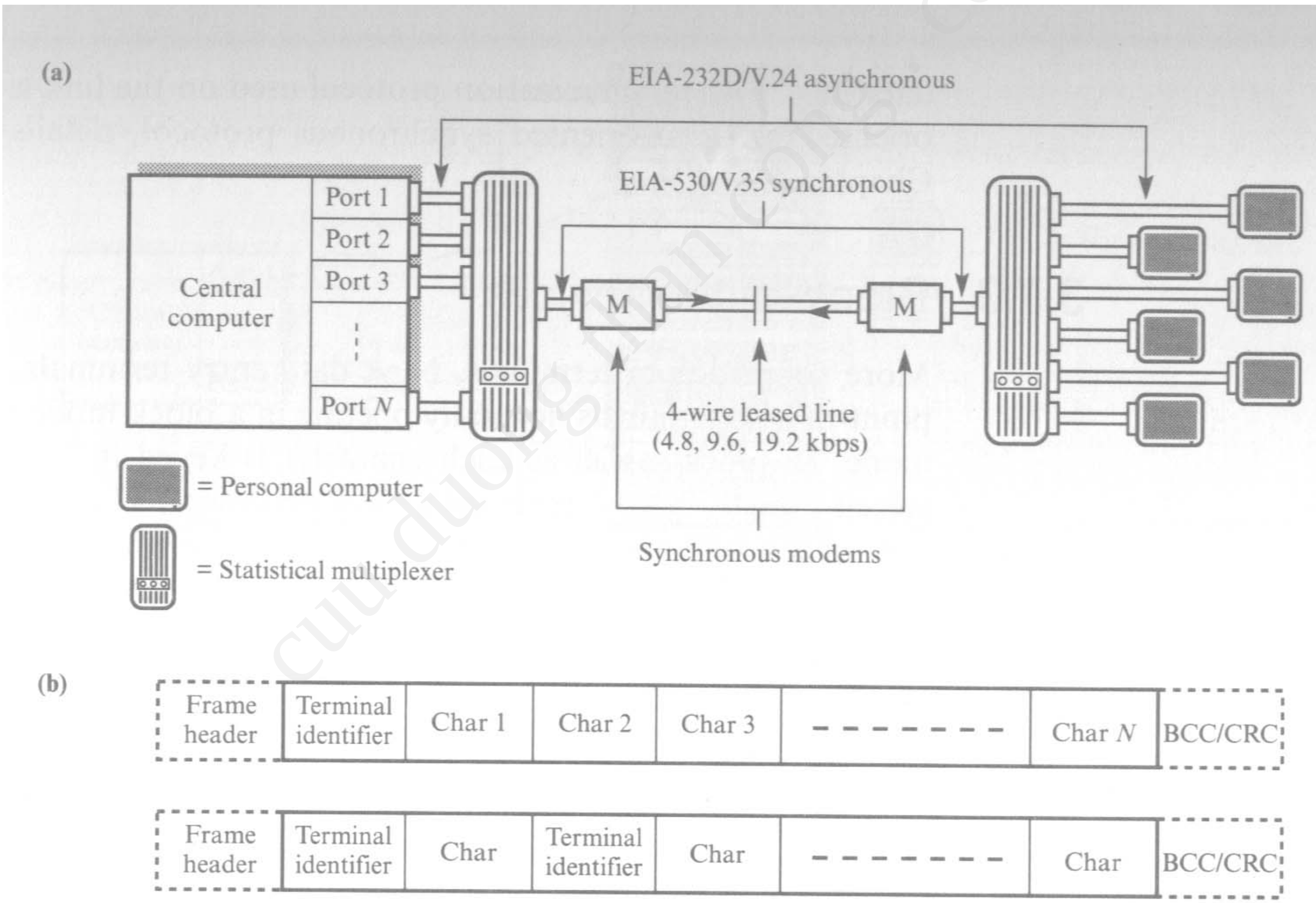
Time-division Multiplexer



Time-division Multiplexer



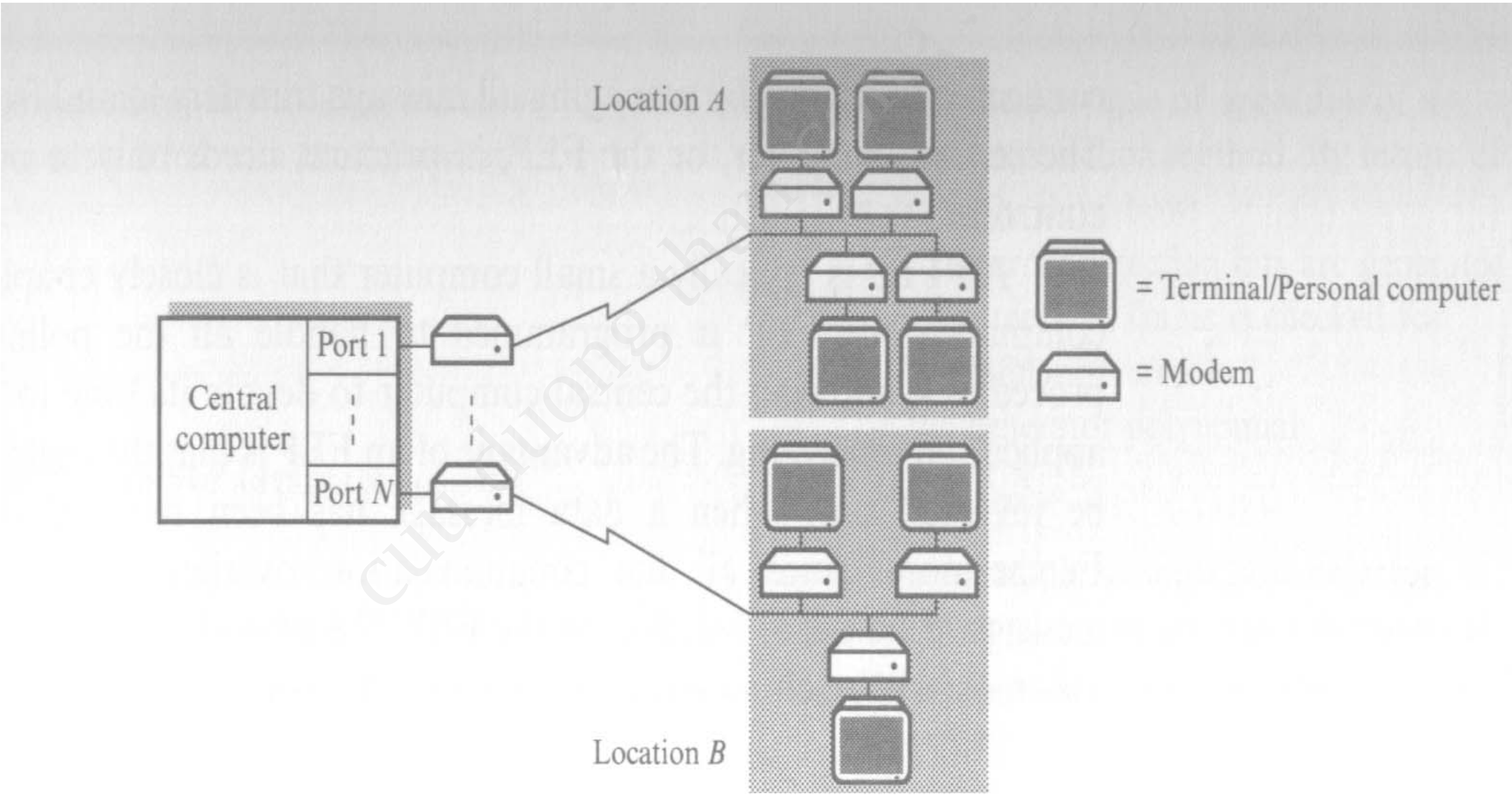
Statistical Multiplexer



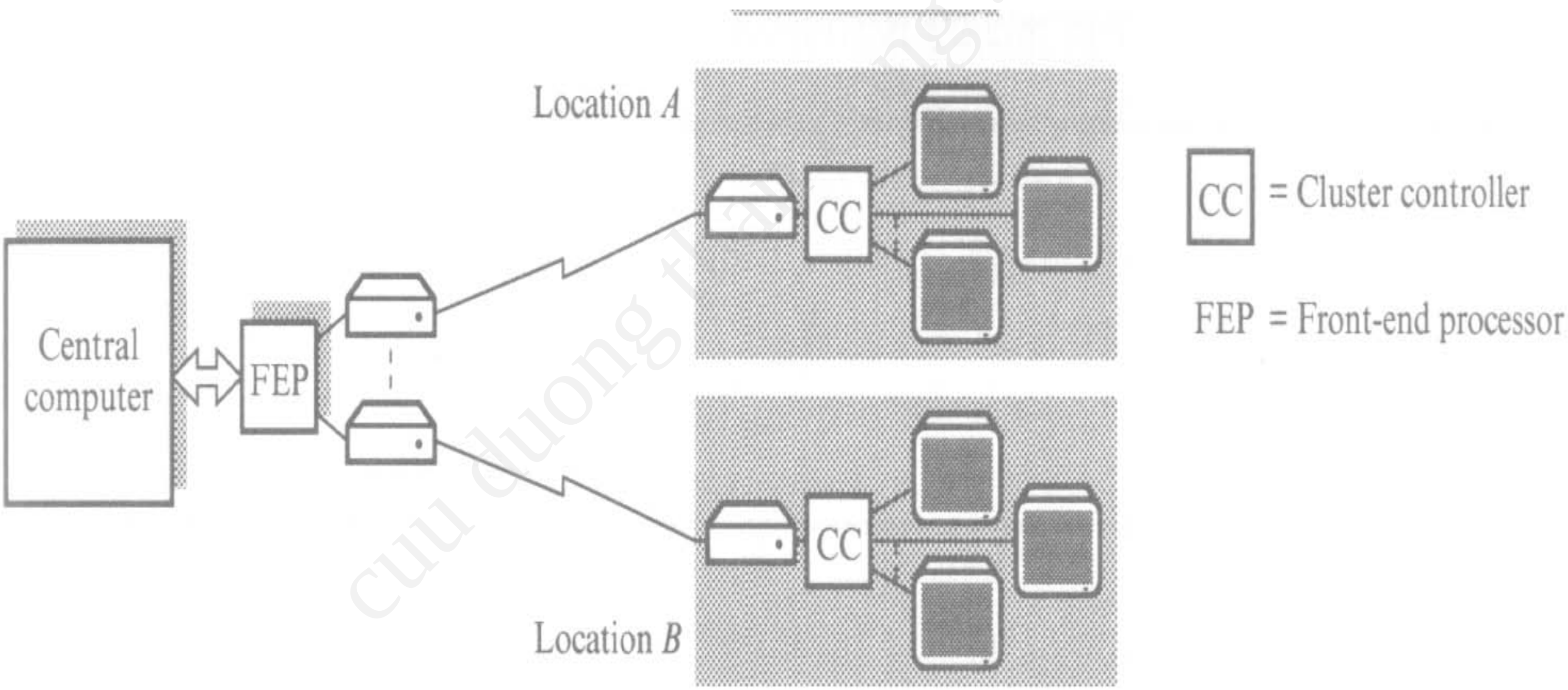
Block-mode Devices

- Multidrop (multipoint) lines
- Cluster controller
- Hub-polling

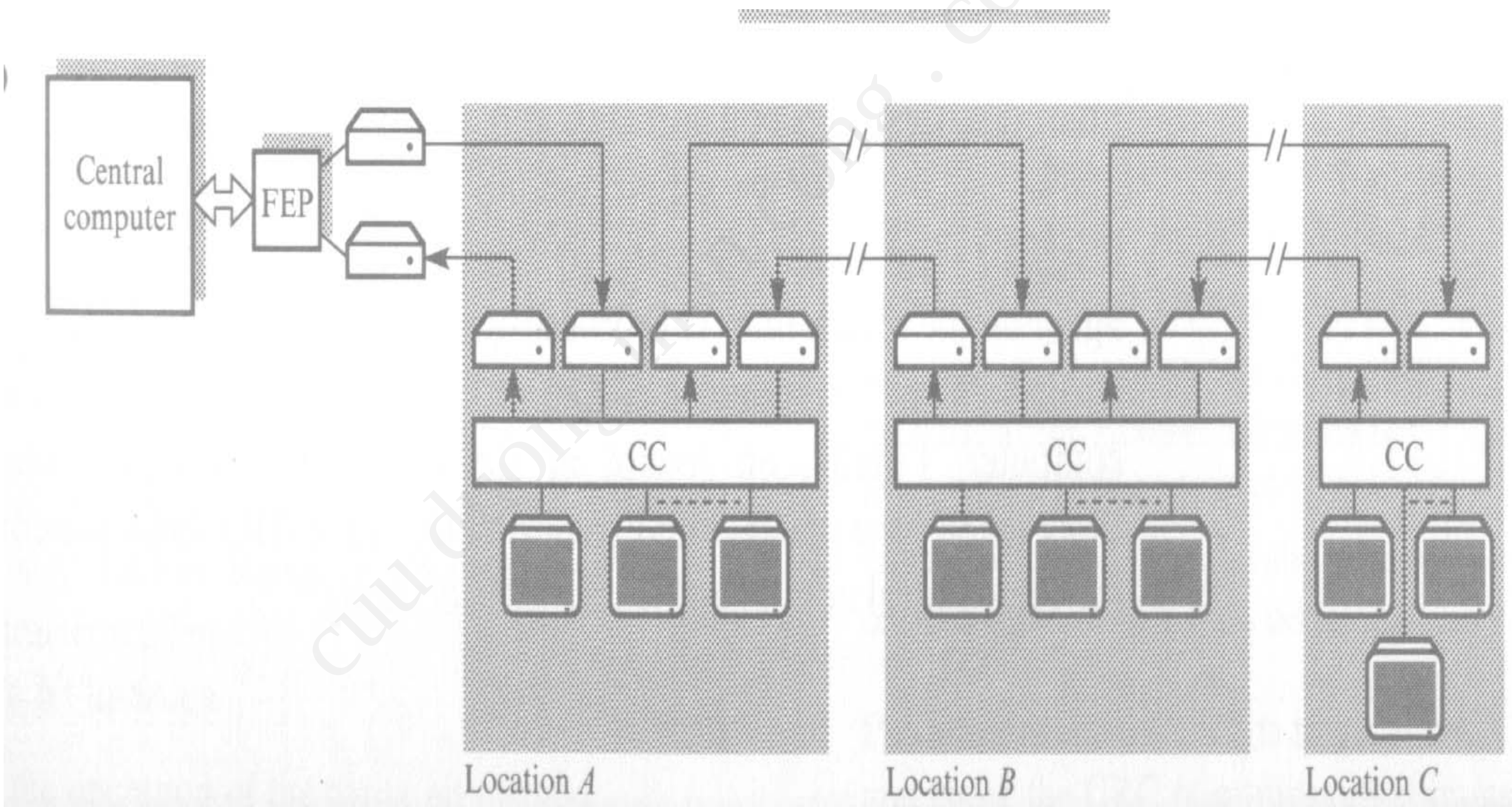
Block-mode Devices

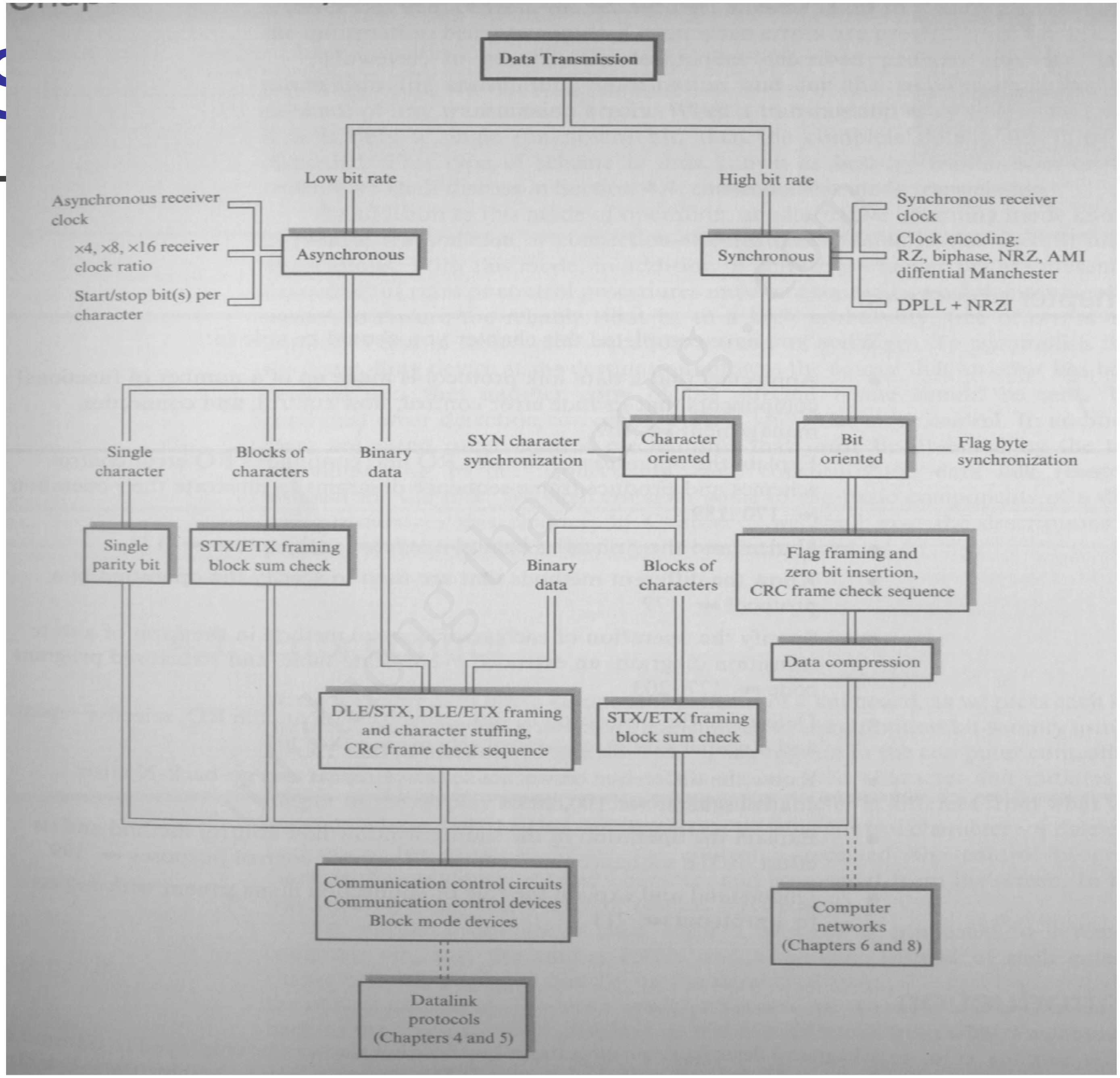


Block-mode Devices



Block-mode Devices




Data Transmission
Low bit rate
High bit rate
Asynchronous receiver clock
×4, ×8, ×16 receiver clock ratio
Start/stop bit(s) per character
Asynchronous
Synchronous
Synchronous receiver clock
Clock encoding:
RZ, biphase, NRZ, AMI
diffential Manchester
DPLL + NRZI
Single character
Blocks of characters
Binary data
SYN character synchronization
Character oriented
Bit oriented
Flag byte synchronization
Single parity bit
STX/ETX framing block sum check
Binary data
Blocks of characters
Flag framing and zero bit insertion, CRC frame check sequence
Data compression
DLE/STX, DLE/ETX framing and character stuffing, CRC frame check sequence
STX/ETX framing block sum check
Communication control circuits
Communication control devices
Block mode devices
Computer networks (Chapters 6 and 8)
Datalink protocols (Chapters 4 and 5)