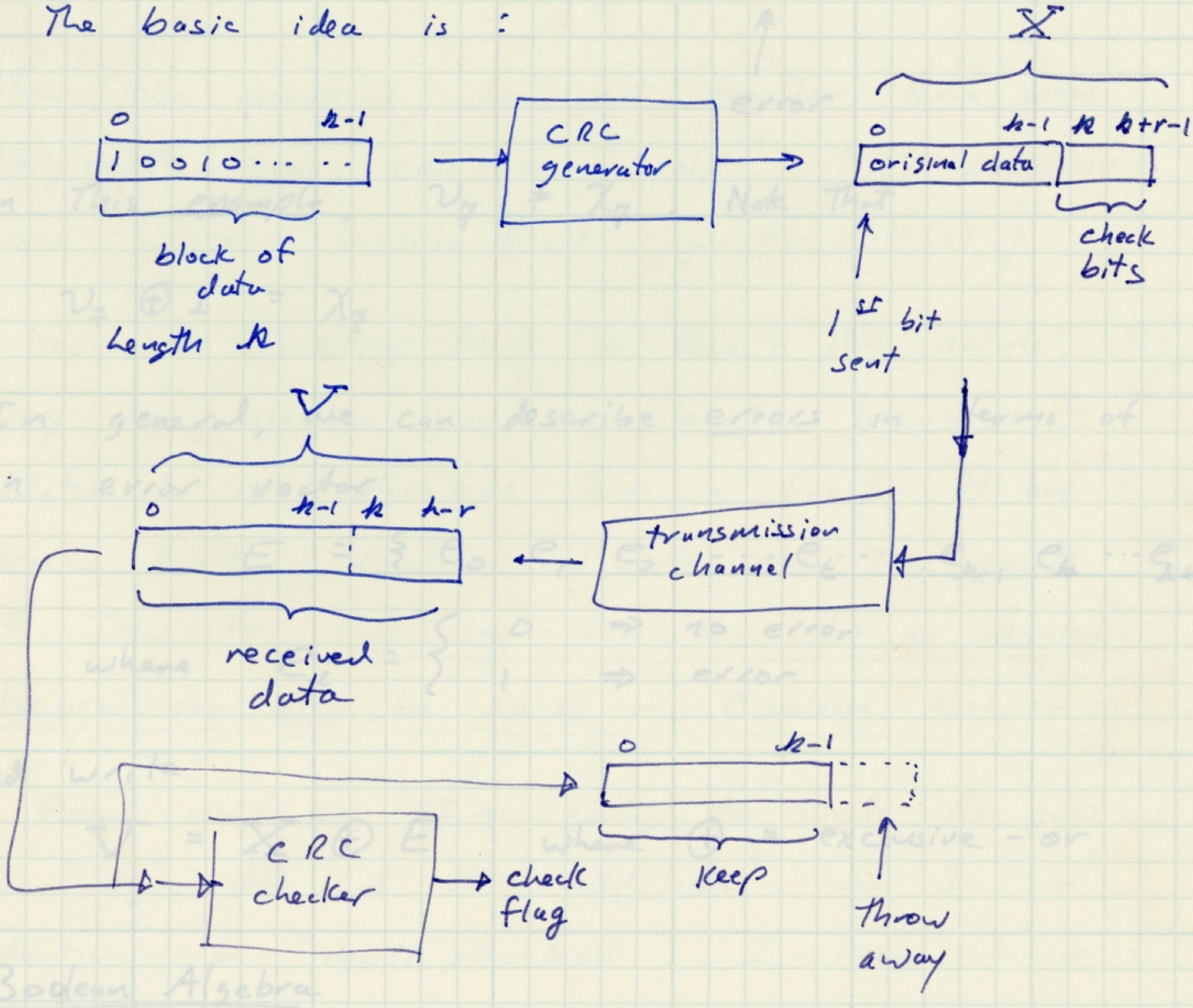
EE 504 Adv. Embedded Controllers   Guest Lecture   950206

Understanding Cyclic Redundancy Check

First what is CRC? Basically, it's just a way to detect errors in data transmission. The basic idea is:



We need a way to describe these data blocks. Let The data bit at time t be d_t. This gives us a sequence of data

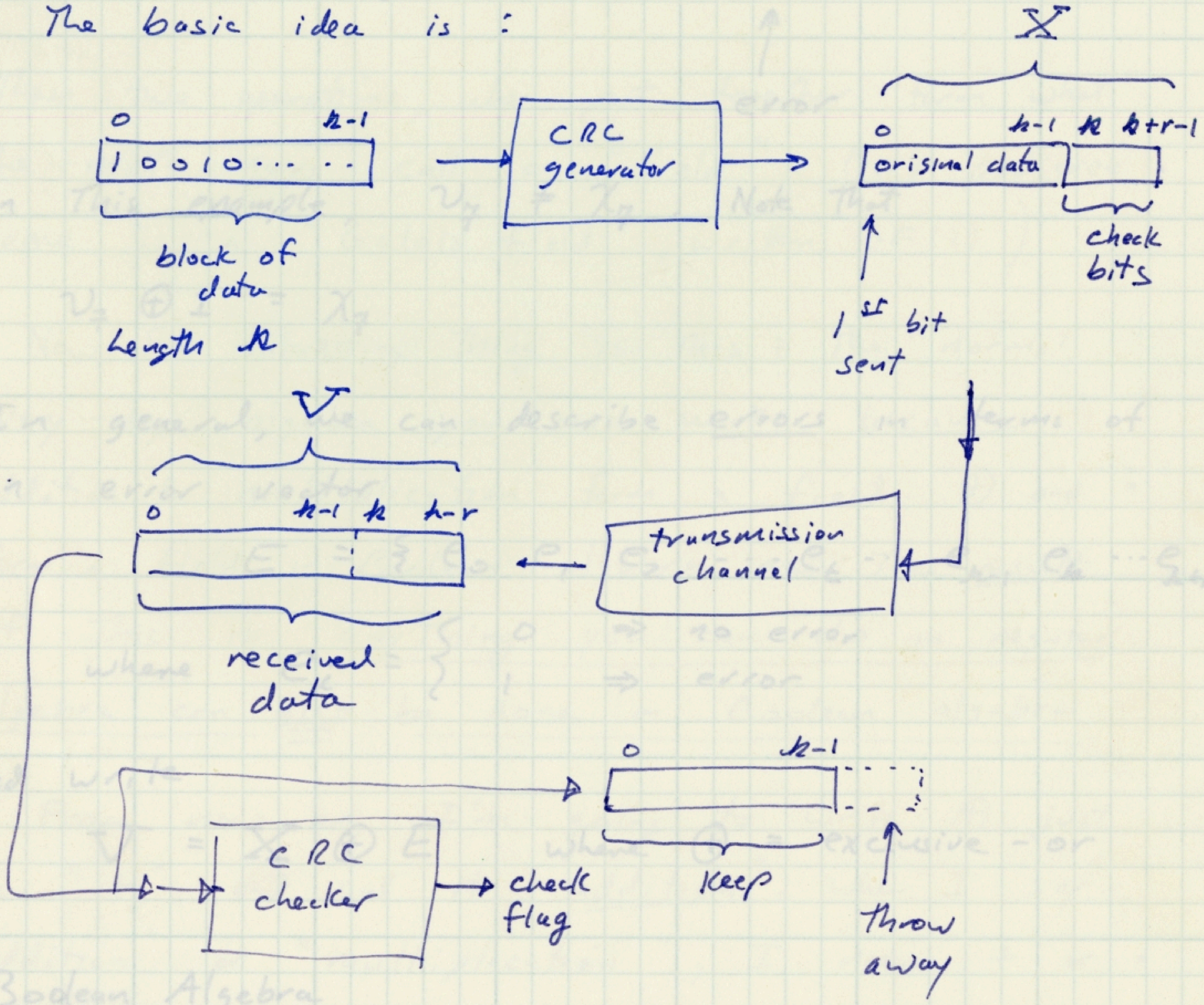
$$X = \{ d_0 \ d_1 \ d_2 \ \cdots \ d_t \ \cdots \ d_{k-1} \ d_k \ \cdots d_{k-r} \}$$

$$n = k + r \text{ bits}$$

Understanding Cyclic Redundancy Check

First what is CRC? Basically, it's just a way to detect errors in data transmission.

The basic idea is :



we need a way to describe these data blocks. Let the data bits at time t be d_t. This gives us a sequence of data

$$X = \{ d_0 \; d_1 \; d_2 \cdots d_t \cdots d_{k-1} \; d_k \cdots d_{k-r} \}$$

$$n = k + r \text{ bits}$$

if $V \neq X$, That means an error has occurred.

Example:

$$\begin{array}{rccccccccl} t= & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\ X = \{ & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \} \\ V = \{ & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \\ & & & & & & \uparrow & & & \\ & & & & & & \text{error} & & & \end{array}$$

in This example, $v_7 \neq x_7$. Note That

$$v_7 \oplus 1 = x_7$$

In general, we can describe errors in terms of an error vector

$$E = \{ e_0 \; e_1 \; e_2 \cdots e_t \cdots e_{k-1} \; e_k \cdots e_{k+r} \}$$

$$\text{where} \quad e_i = \begin{cases} 0 & \Rightarrow \text{no error} \\ 1 & \Rightarrow \text{error} \end{cases}$$

and write

$$V = X \oplus E \quad \text{where } \oplus = \text{exclusive-or}$$

Boolean Algebra

You all remember Boolean algebra. It's called Boolean algebra because The math operators obey certain special properties. In particular

$$\oplus = \text{"addition" in Boolean algebra}$$

$$0 \oplus 0 = 0 \qquad 1 \oplus 0 = 1$$

$$0 \oplus 1 = 1 \qquad 1 \oplus 1 = 0$$

"Multiplication" in Boolean algebra is just the logic "AND" operation

$0 \cdot 0 = 0 \qquad 1 \cdot 0 = 0$

$0 \cdot 1 = 0 \qquad 1 \cdot 1 = 1$

These two operations, when put together, form what the math guys call a "field" (it's pedigree name is a "Galois field", written $GF(2)$)

Now, the interesting thing is this: the normal, every-day algebra you use works because addition & multiplication form a field. $\oplus$ and $\cdot$ form a field in Boolean algebra and the result of this is anything you can do in regular algebra can also be done in Boolean algebra.

From now on, I'm going to write $\oplus$ just as "+" and call it "addition". When I say "addition" or "multiplication", I mean + or $\cdot$ under Boolean algebra.

First, let's look at subtraction. If I have a number "x" and a number "y", when I write "$y - x$" what I mean is

"add to y the additive inverse of x"

i.e., $-x$ is the number such that $x + (-x) = 0$

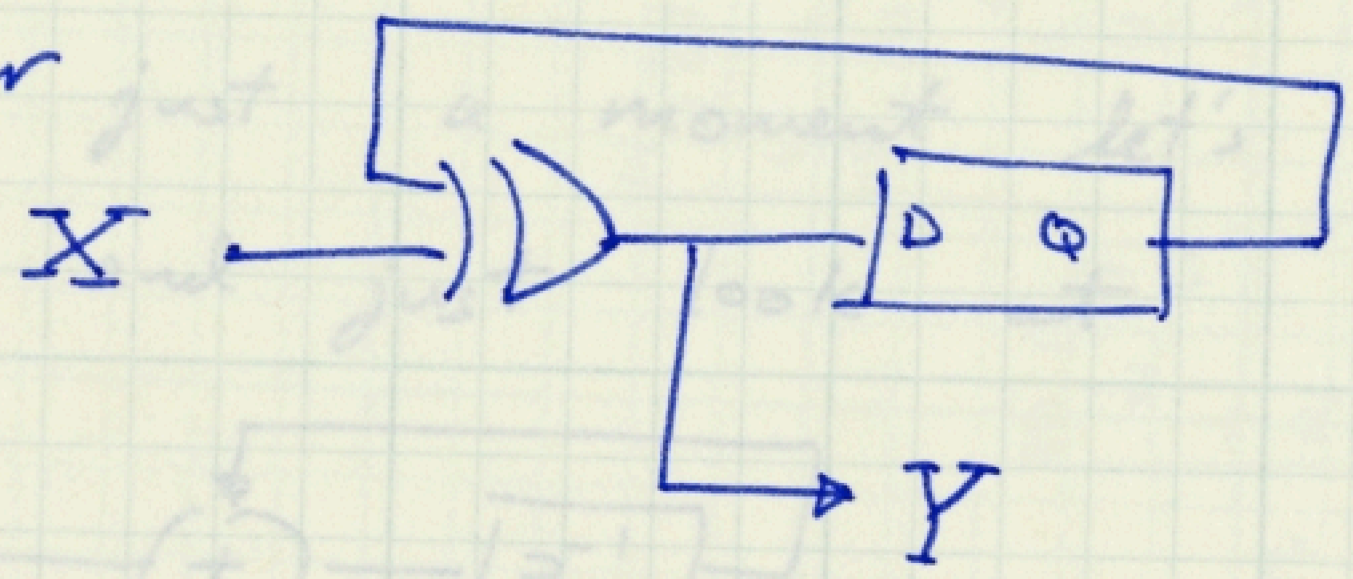
In Booleau algebra, $1 + 1 = 0$ so
"1" is the additive inverse of "1" !

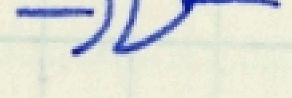
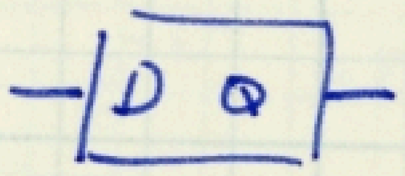
$\therefore \quad x + x = 0$ in boolean algebra

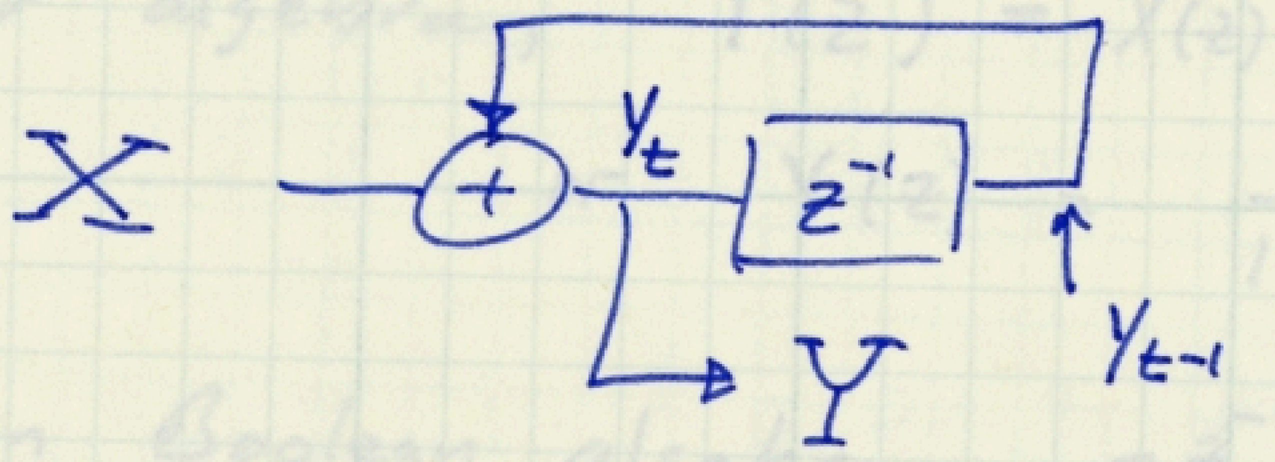
$\therefore \quad x = -x \Rightarrow$ you can't ever make a sign error !

Shift Register Sequences

consider



Since  = addition and  = delay,
I can write this in block diagram form



z^{-1} = "delay" operator

Now suppose $y_{-1} = 0$ and

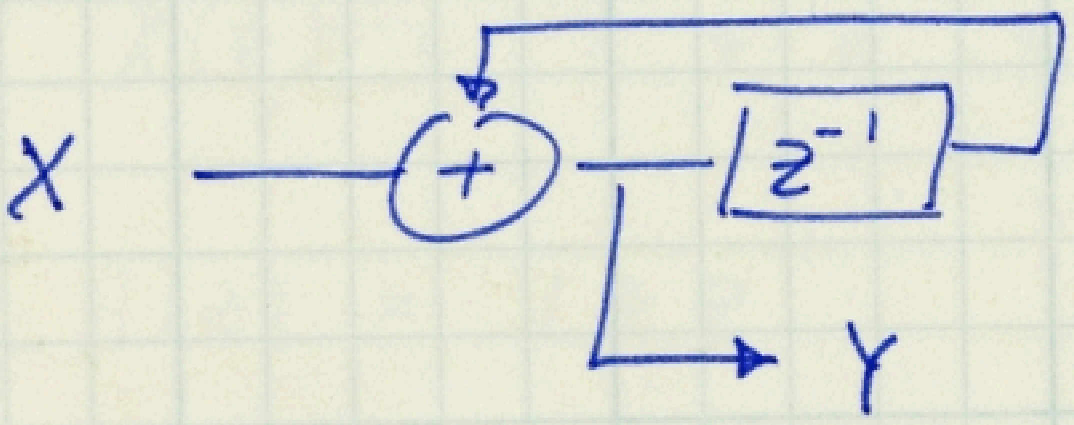
$X = \{ x_0 \;\; x_1 \; \cdots \} = \{ 1 \;\; 0 \;\; 1 \;\; 1 \}$

what is $Y = \{ y_0 \;\; y_1 \; \cdots \}$?

One way to solve this is to build a table

t	x_t	y_{t-1}	$y_t = x_t + y_{t-1}$
0	1	0	1
1	0	1	1
2	1	1	0
3	1	0	1
4	0	1	1
5	0	1	1
⋮	⋮	⋮	⋮

Now, for just a moment let's forget we've got a SR and just look at



In regular algebra, $Y(z) = X(z) + z^{-1} Y(z)$

or $Y(z) = \dfrac{X(z)}{1 - z^{-1}}$

However, in Boolean algebra $-z^{-1} = +z^{-1}$ because

"$-z^{-1}$" means "additive inverse of z^{-1}"

Now let's write $X = \{ 1\ 0\ 1\ 1\ 0\ 0 \cdots \}$ as a "z-transform"

$$X(z) = 1 + 0 \cdot z^{-1} + 1 \cdot z^{-2} + 1 \cdot z^{-3}$$

$$= 1 + z^{-2} + z^{-3}$$

what do we get when we write

$$Y(z) = \frac{X(z)}{1+z^{-1}} \; ?$$

We find out by long division : $(1+z^{-1})\overline{\big)X(z)}^{\,Y(z)}$

so

$$\begin{array}{r l}
 & \quad 1 + z^{-1} + z^{-3} + z^{-4} \\
1 + z^{-1} & \overline{\big) \; 1 + \phantom{z^{-1}+} z^{-2} + z^{-3}} \\
 & \underline{\;\;1 + z^{-1}} \\
 & \qquad z^{-1} + z^{-2} + z^{-3} \\
 & \qquad \underline{z^{-1} + z^{-2}\;\;\;\;\;\;\;\;} \\
 & \qquad\qquad\qquad z^{-3} \\
 & \qquad\qquad\qquad \underline{z^{-3} + z^{-4}} \\
 & \qquad\qquad\qquad\qquad z^{-4} \\
 & \qquad\qquad\qquad\qquad \underline{z^{-4} + z^{-5}} \\
 & \qquad\qquad\qquad\qquad\qquad z^{-5} \;\; \text{etc}
\end{array}$$

(can't make a sign error!)

so

$$Y(z) = 1 + z^{-1} + z^{-3} + z^{-4} + z^{-5} + \cdots$$

or

$$\begin{array}{l c c c c c c c c}
Y = \{ & 1 & 1 & 0 & 1 & 1 & 1 & \cdots & \} \\
 & \uparrow & & & & & & & \\
t = & 0 & 1 & 2 & 3 & 4 & 5 & \cdots &
\end{array}$$

Wow! Did we just get lucky? No. Anything you can do in regular algebra you can also do in Boolean algebra. Under boolean algebra, a shift register plus EX-OR gates is a linear, time-invariant system!

Polynomial division

When you first learned arithmetic, you learned it as "~~product~~ quotients" and "remainders"

$$7 \div 3 = 2 \text{ w/ Remainder} = 1$$

$$\begin{array}{r|l} & 2 \leftarrow \text{quotient} \\ 3 & 7 \\ & \underline{6} \\ & 1 \leftarrow \text{remainder} \end{array}$$

We can do the same thing with polynomials

$$X(z) = 1 + z^{-2} + z^{-3} \Leftrightarrow \{1 \ 0 \ 1 \ 1 \ 0\}$$

↑ one extra place for 1 shift in S.R.

$$H(z) = 1 + z^{-1}$$ ("Shift Register polynomial")

Then $X(z) \div H(z)$ becomes

$$\begin{array}{r|l} & 1 + z^{-1} + 0 \cdot z^{-2} + z^{-3} \cancel{+ z^{-4}} \\ \hline 1 + z^{-1} & 1 + z^{-2} + z^{-3} + 0 \cancel{z^{-4}} \\ & \underline{1 + z^{-1}} \\ & \quad z^{-1} + z^{-2} \\ & \quad \underline{z^{-1} + z^{-2}} \\ & \qquad z^{-3} \\ & \qquad \underline{z^{-3} + z^{-4}} \\ & \qquad\quad z^{-4} \leftarrow \text{remainder} \\ & \qquad\quad \cancel{\underline{z^{-4} + z^{-5}}} \\ & \qquad\quad \cancel{z^{-5}} \leftarrow \cancel{\text{remainder}} \end{array}$$

So $X(z) \div H(z) = [Q(z), R(z)]$

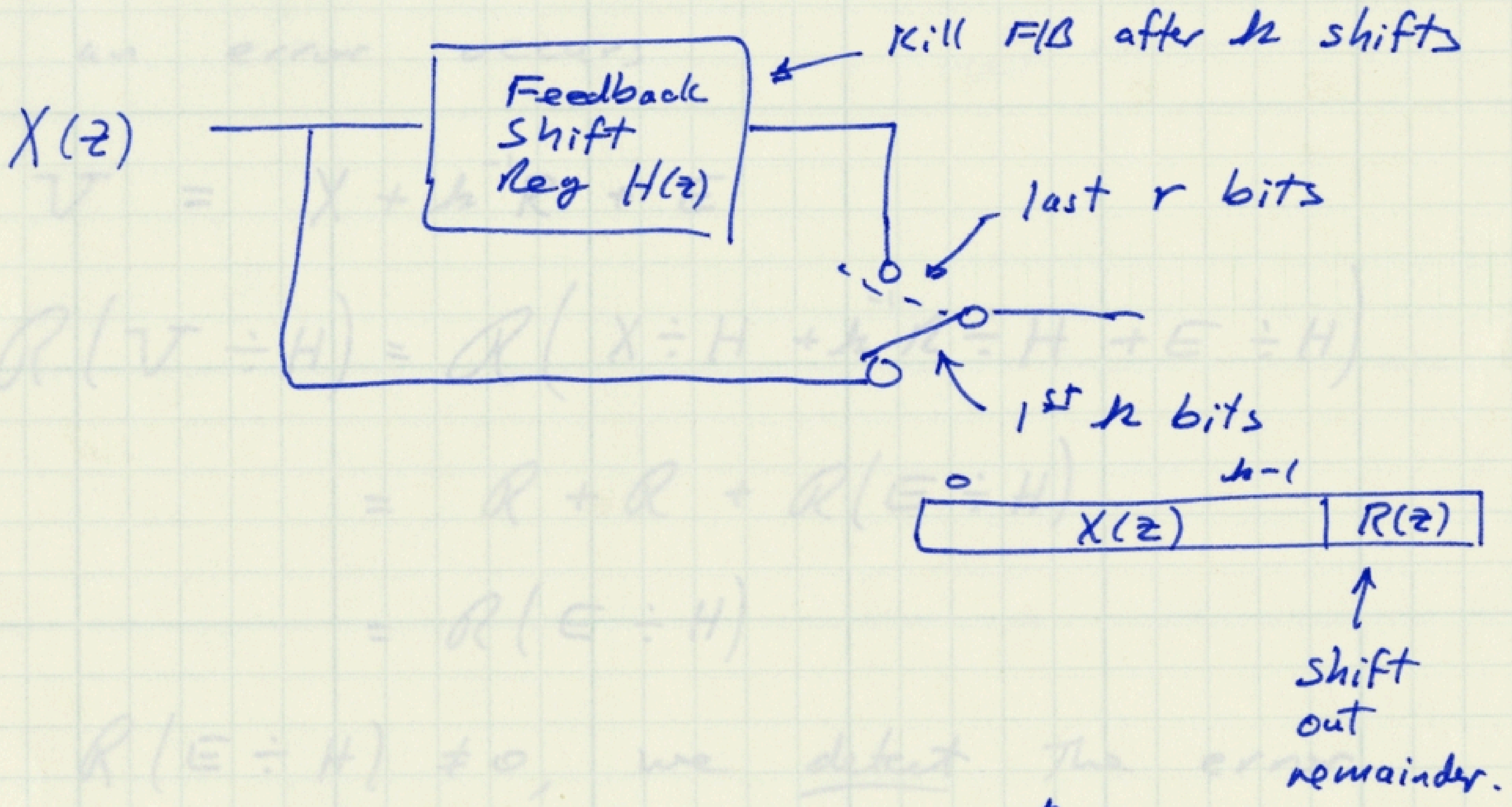
↑ Quotient ↑ remainder

$$Q(z) = 1 + z^{-1} + z^{-3} \cancel{+ z^{-4}}$$
$$R(z) = z^{-4}$$

Notice That $R(z)$ was just what was left in The shift register *after* The 4-bits of $X(z)$ was shift thru!

CRC

CRC generators work like this:



So transmitted bits are $X(z) + z^{-k} R(z)$

At The receiver, we do The same thing *except* we run *all* $n = k+r$ bits thru.

Since $X \div H \to R$

at rcvr, if $V = X + z^{-k} R$

$$\mathcal{R}\left[V \div H\right] \Rightarrow \mathcal{R}\left[\left(X + z^{-k} R\right) \div H\right] = \mathcal{R}\left[X \div H\right] + \mathcal{R}\left[z^{-k} R \div H\right]$$

but since $z^{-k} R$ is *shorter* than H, $\mathcal{R}\left[z^{-k} R \div H\right] = R$

ex: $\begin{array}{r} 0 \\ 5\,\overline{)\,4} \\ \underline{0} \\ 4 \leftarrow R \end{array}$

since $\mathcal{R}[X \div H] = R$

$$\mathcal{R}[V \div H] = R + R = 0$$

$\therefore$ if no errors, contents of S.R. = 0 after $n = k + r$ shifts.

<u>If</u> an error occurs

$$V = X + k^{-1}R + E$$

and $\mathcal{R}(V \div H) = \mathcal{R}\left(X \div H + k^{-1}R \div H + E \div H\right)$

$$= R + \mathcal{R} + \mathcal{R}(E \div H)$$

$$= \mathcal{R}(E \div H)$$

if $\mathcal{R}(E \div H) \neq 0$, we <u>detect</u> the error.

<u>CRC polynomials</u>

In CRC literature, they like to write $H(X)$ <u>instead of</u> $H(z)$. That is

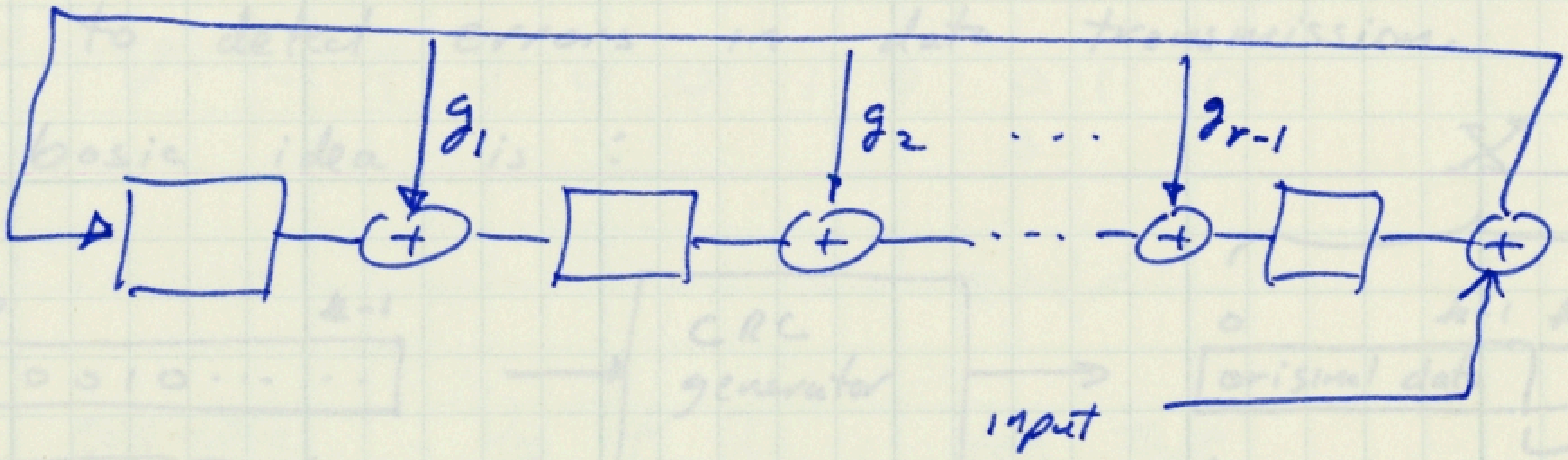
$$H(z) = 1 + z^{-1} + z^{-3} \Rightarrow H(X) = 1 + X + X^{3}$$

Certain $H(X)$s are preferred because they can't be factored into a product of smaller polynomials. The ones ~~for~~ that are important are the so-called "primitive polynomials".

if $H(x) = 1 + g_1 x + g_2 x^2 + \cdots + x^{r-1}$

$g_i = \begin{cases} 0 \\ 1 \end{cases}$

You <u>build</u> it as


g_1
g_2
. . .
g_{r-1}
+
+
+
+
input