CS 348: Computer Networks

- Error Detection; 7th Aug 2012

Instructor: Sridhar Iyer
IIT Bombay

Bit level error detection

Single-bit, multi-bit or burst errors introduced due to channel noise.

- Detected using redundant information sent along with data.

- Full Redundancy:
  - Send everything twice
  - Simple but inefficient

Error correcting codes

- **error-correcting codes:** include enough redundant information with data so receiver can recover data (useful on simplex channels where retransmission cannot be requested)

- Beyond the scope of this course!

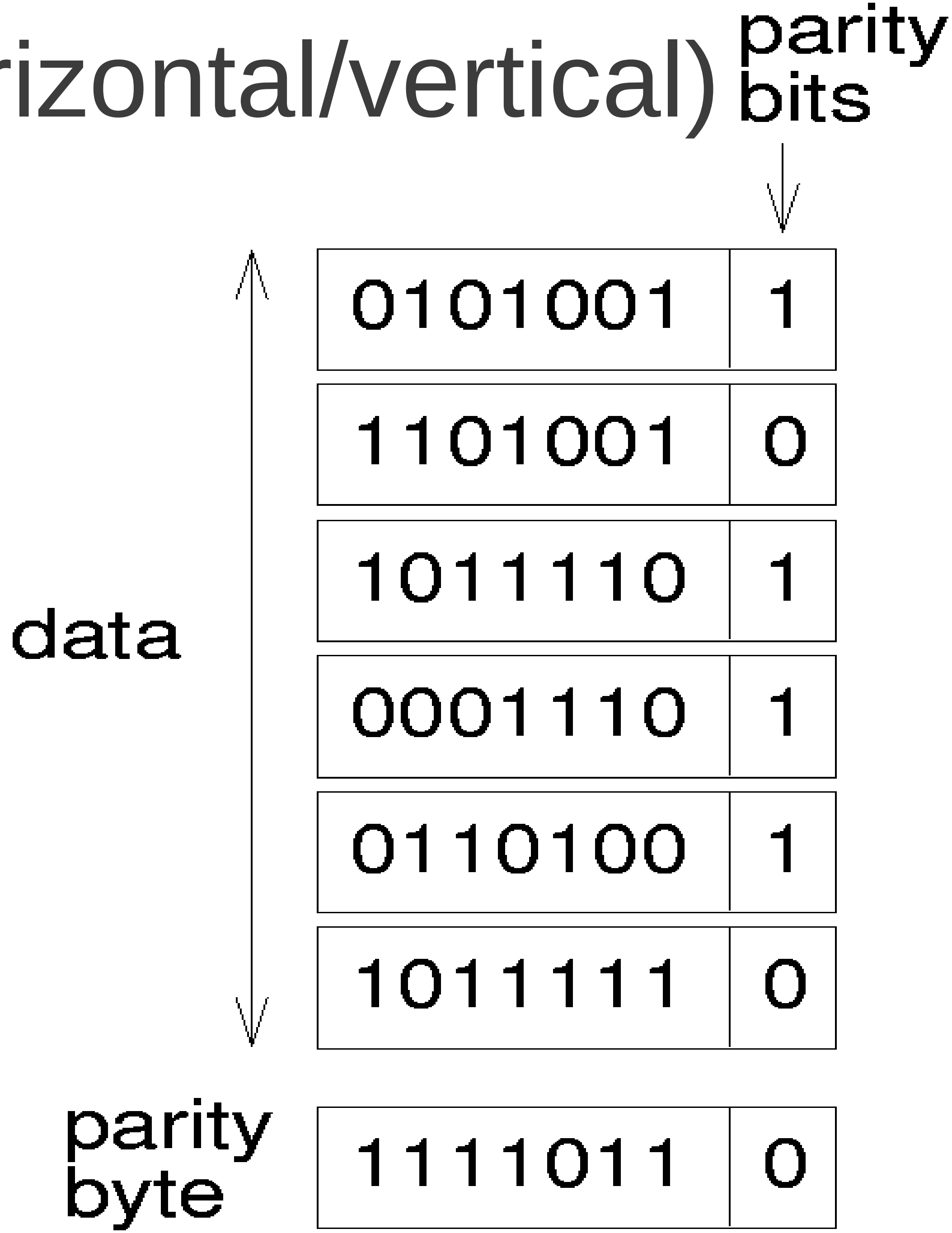
Error detecting codes

- **error-detecting codes**: include enough redundancy to allow receiver to detect error.

- **more efficient and preferred solution**
  - parity check.
  - checksum.
  - cyclic redundancy check (CRC or polynomial code).

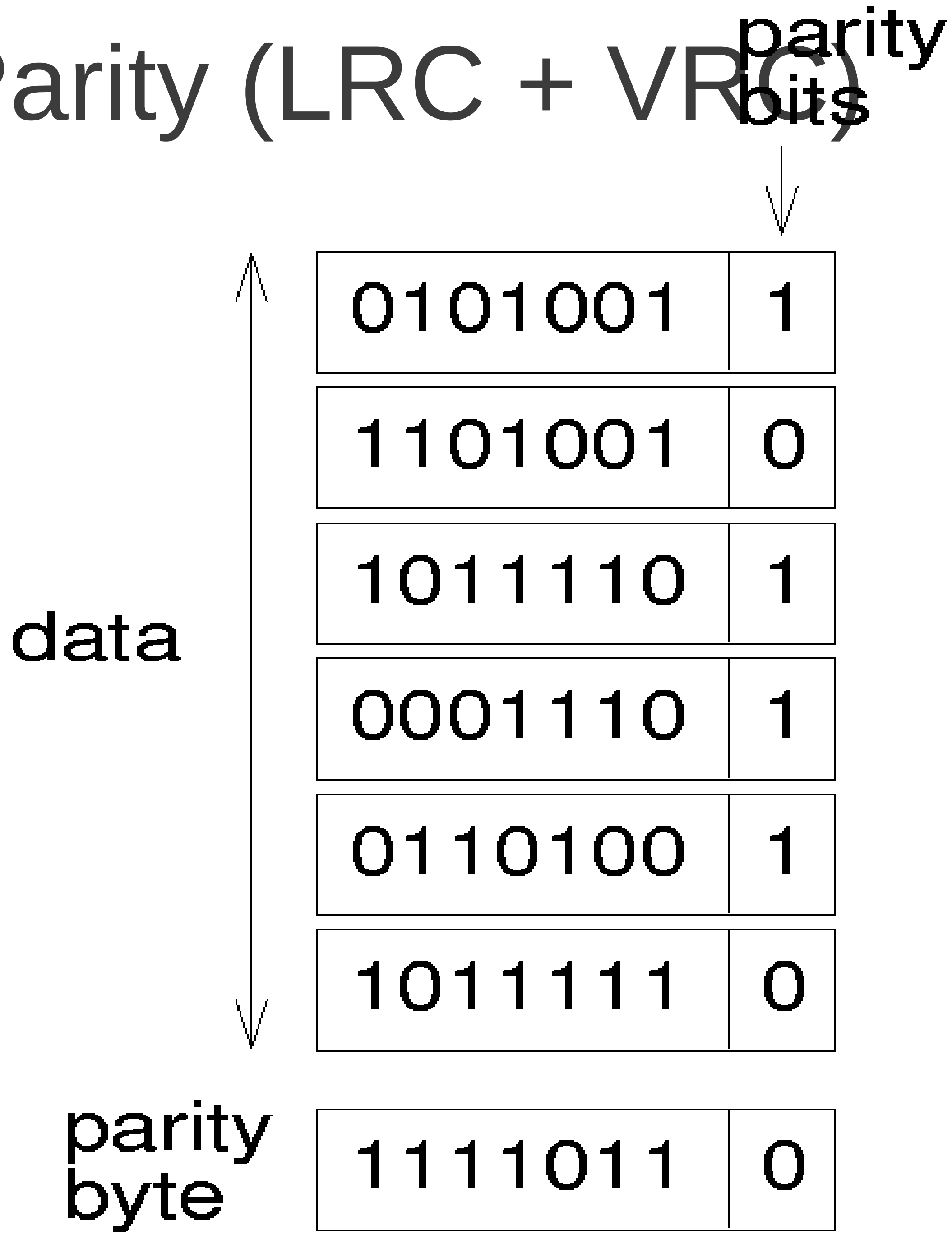
Parity (horizontal/vertical)

- 1 bit error detectable, not correctable
- 2 bit error not detectable



Rectangular Parity (LRC + VRC)

- 1 bit error correctable
- 2 bit error detectable
- Slow, needs memory



Checksum

- IP header; TCP/UDP segment checksum.
  - View message as 16-bit integers.
  - Add these integers using 16-bit ones-complement arithmetic.
  - Take the ones-complement of the result.
  - Resulting 16-bit number is the checksum.

- Can detect all 1 bit errors.
- See example in textbook.

Cyclic Redundancy Check

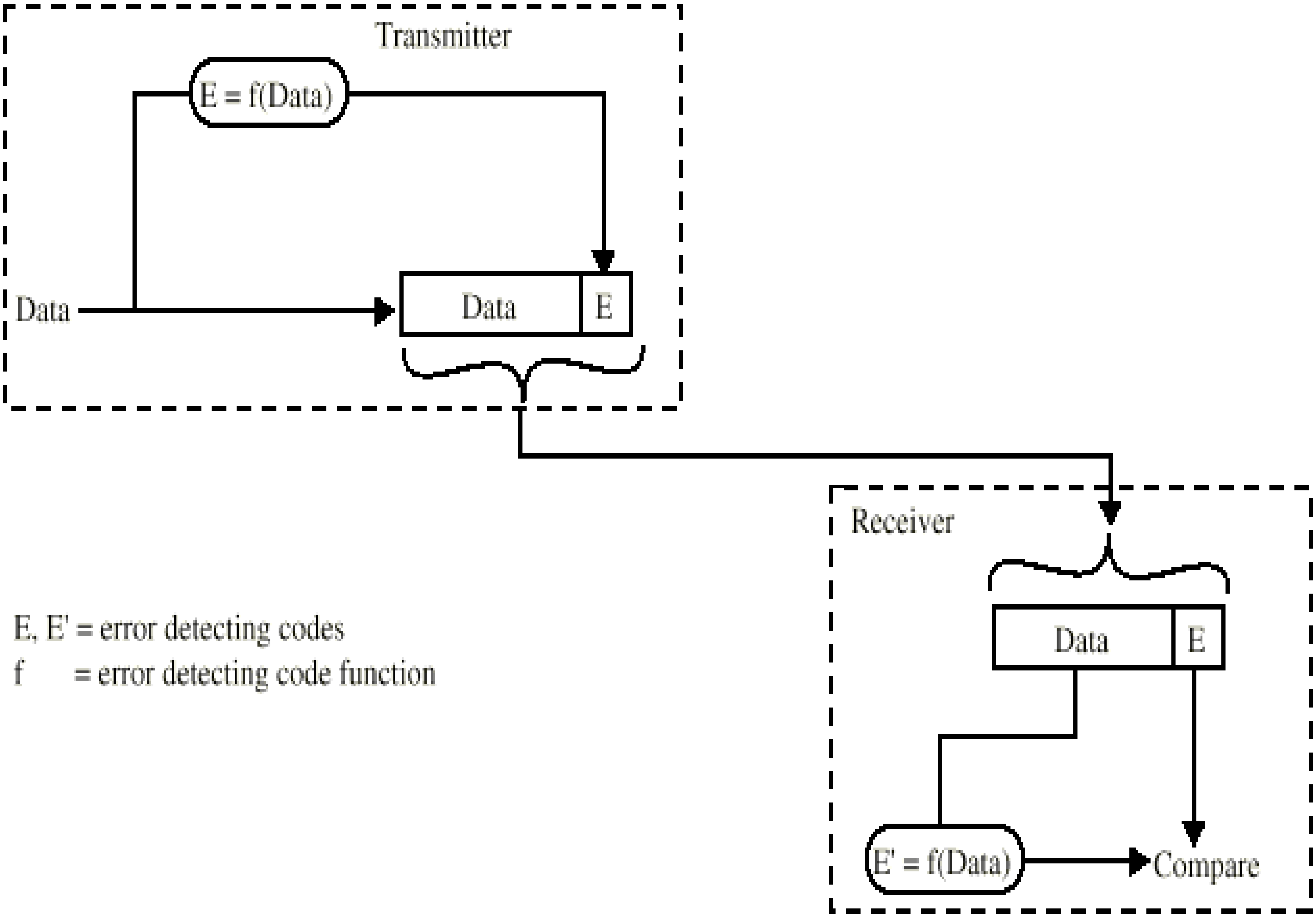
- Based on binary division instead of addition.
- Powerful and commonly used to ‘detect’ errors.
- Uses modulo 2 arithmetic:
  - Add/Subtract := XOR (no carries for additions or borrows for subtraction)
  - 2^k * M := shift M towards left by k positions and then pad with zeros
- Digital logic for CRC is fast. no delay, no storage

CRC algorithm

To transmit message M of size of n bits:

- Source and destination agree on a common bit pattern P of size k+1 (k > 0)

- Source: Add (in modulo 2) bit pattern (F) of size k to the message M (k < n), such that
  - 2^k * M + F = T is evenly divisible (modulo 2) by pattern P.
- Destination: check if above condition is true
  - i.e. (2^k * M + F)/ P = 0



E, E' = error detecting codes
f = error detecting code function

Frame check sequence (FCS)

- Given M and P, find appropriate F (frame check sequence)

1. Multiply M with 2^k (add k zeros to end of M)
2. Divide (in modulo 2) the product by P
   - » The remainder R is the required FCS
1. Add the remainder R to the product 2^k*M
2. Transmit the resultant T

Example

- M = 10011010
- P = 1101

M * 2^3 = 10011010000

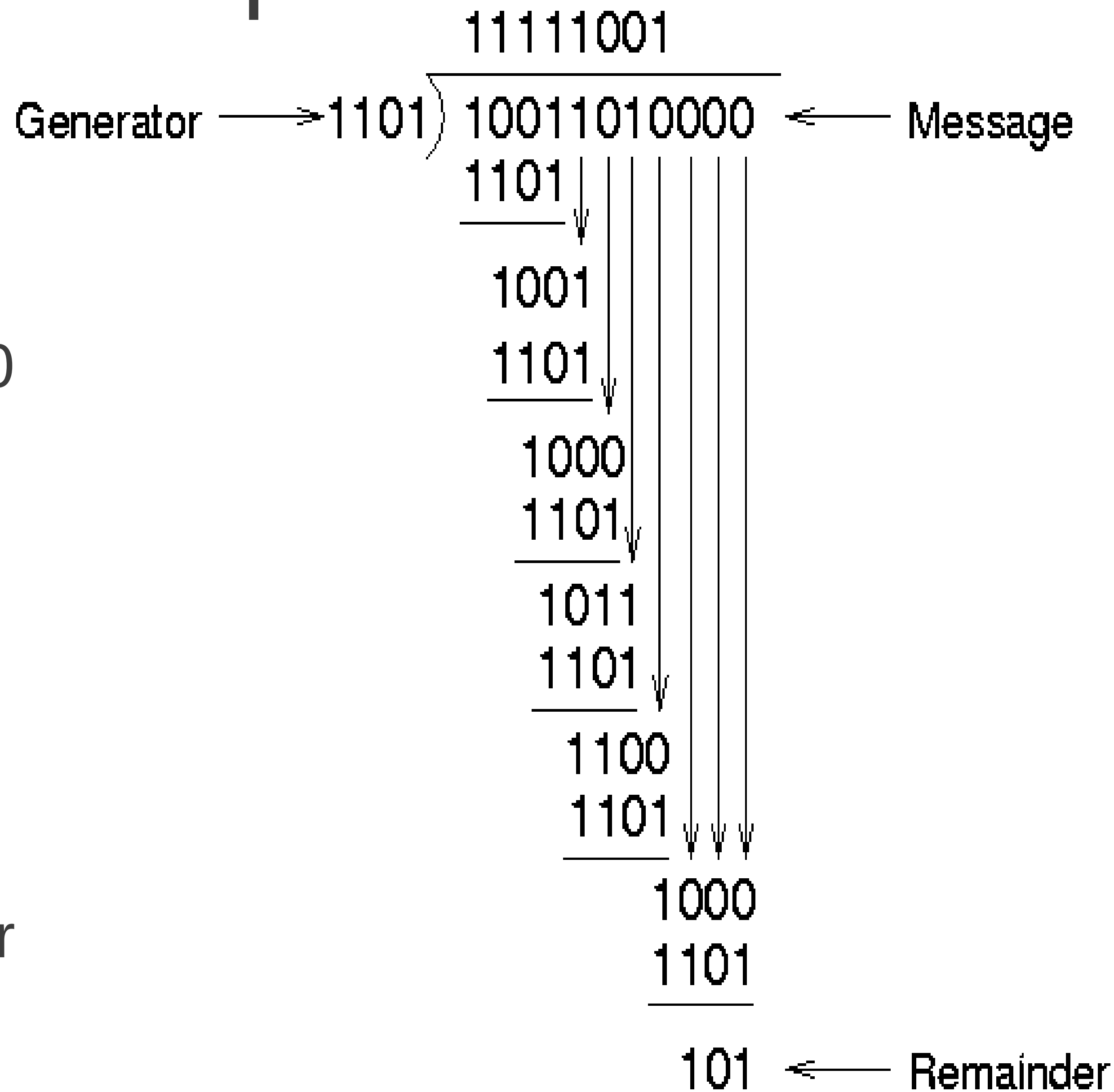
F = 10011010000/1101

= 101

T= 10011010000 + 101
=10011010101

At receiver

- T/P => No remainder



Polynomial representation

- Represent *n*-bit message as an *n*-1 degree polynomial;
  - M=10011010 corresponds to
  - $M(x) = x^7 + x^4 + x^3 + x^1$.
- Let *k* be the degree of some divisor polynomial *C*(*x*); (also called Generator Polynomial)
  - P = 1101 corresponds to
  - $C(x) = x^3 + x^2 + 1$.

Sender actions

- Multiply $M(x)$ by x^k;
  - 10011010000: $x^{10} + x^7 + x^6 + x^4$
- Divide result by $C(x)$ to get remainder R(x);
  - 10011010000/ 1101 = 101
- Send P(x);10011010000 + 101 = 10011010101

Receiver actions

- Receive $P(x) + E(x)$ and divide by $C(x)$
  - E(x) represents the error with 1s in position of errors
- Remainder zero only if:
  - $E(x) = 0$ (no transmission error), or
  - $E(x)$ is exactly divisible by $C(x)$.
- Choose $C(x)$ to make second case extremely rare.

Generator polynomials

- CRC-8 $x^8 + x^2 + x^1 + 1$
- CRC-10 $x^{10} + x^9 + x^5 + x^4 + x^1 + 1$
- CRC-12 $x^{12} + x^{11} + x^3 + x^2 + 1$
- CRC-16 $x^{16} + x^{15} + x^2 + 1$
- CRC-32 $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

CRC detects

- all single bit errors
- almost all 2-bit errors
- any odd number of errors
- all bursts up to M, where generator length is M
- longer bursts with probability 2^-m

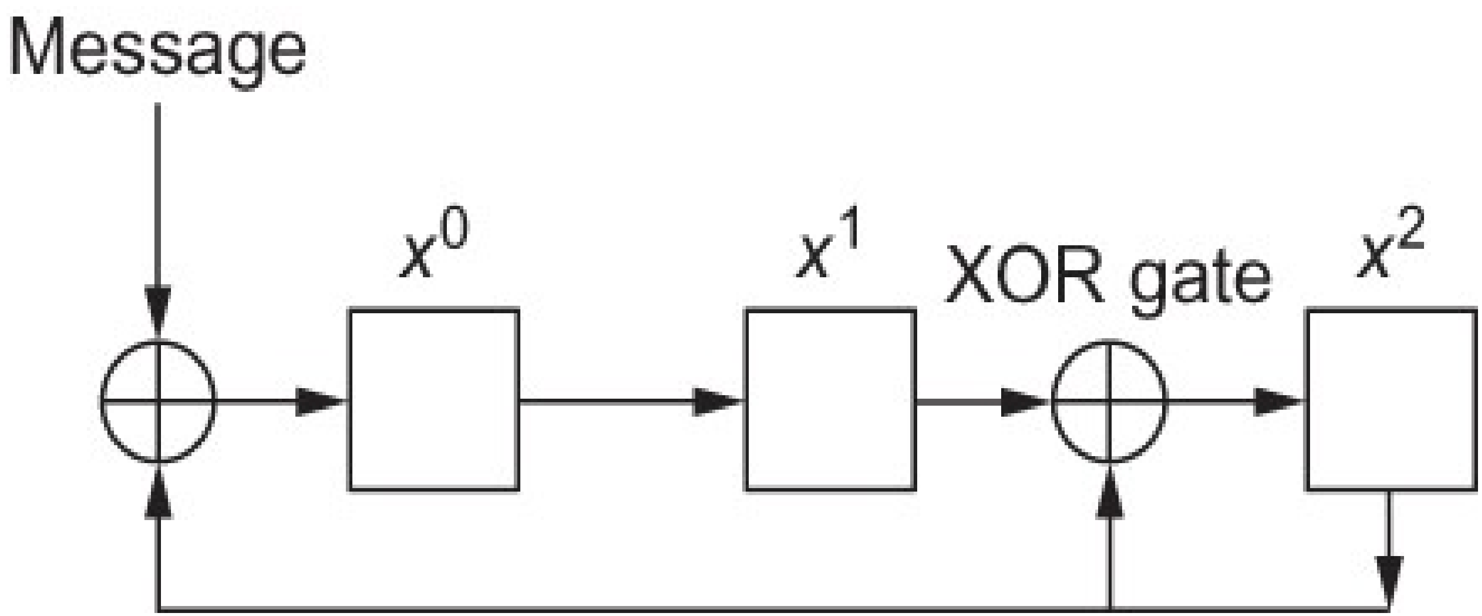
- Proofs – Beyond the scope of this course.

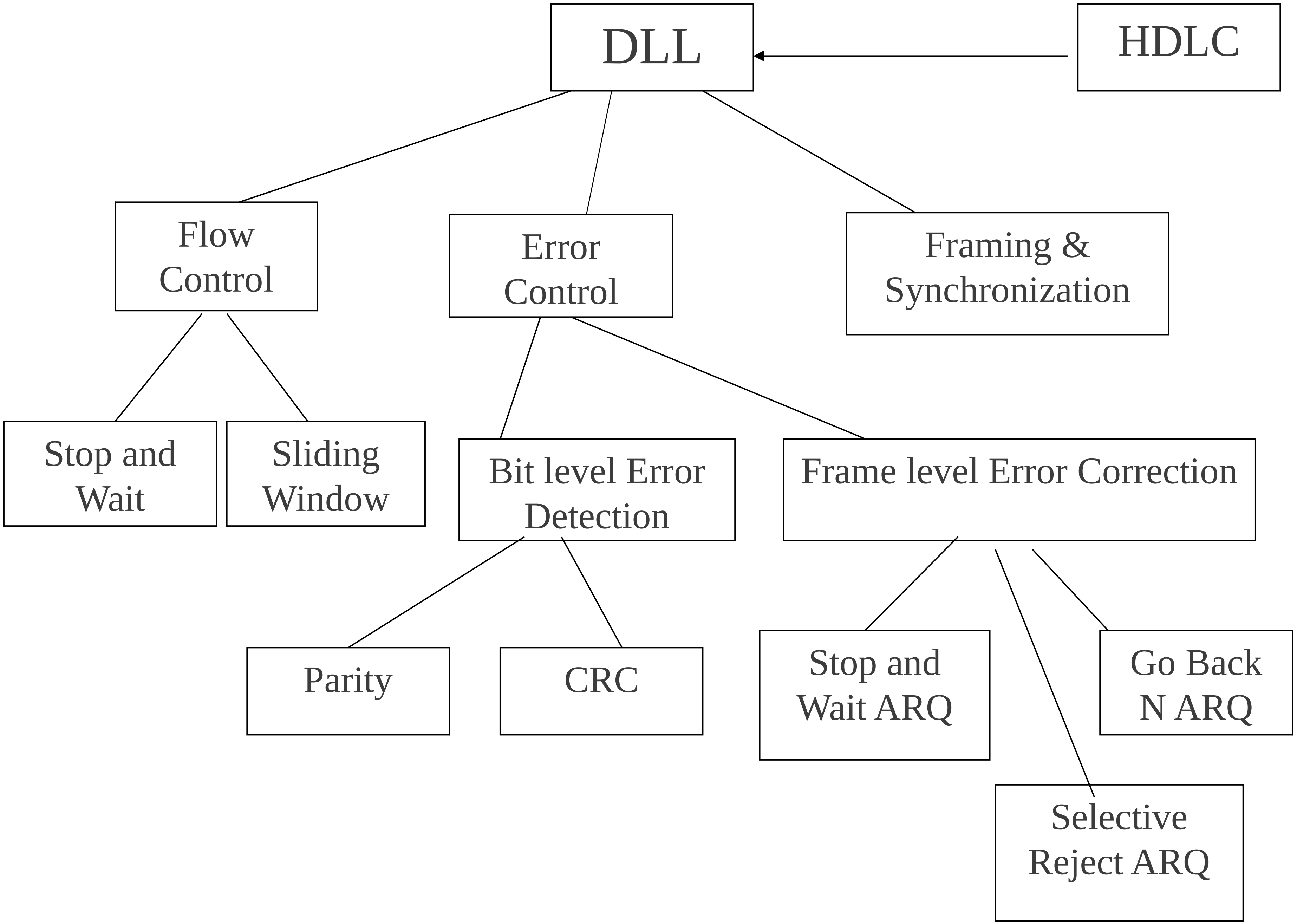
CRC Implementation

- Hardware
  - on-the-fly with a shift register
  - easy to implement with ASIC/FPGA

- Software
  - precompute remainders for 16-bit words
  - add remainders to a running sum
  - needs only one lookup per 16-bit block

CRC calculation using a shift-register




DLL
HDLC
Flow Control
Error Control
Framing & Synchronization
Stop and Wait
Sliding Window
Bit level Error Detection
Frame level Error Correction
Parity
CRC
Stop and Wait ARQ
Go Back N ARQ
Selective Reject ARQ