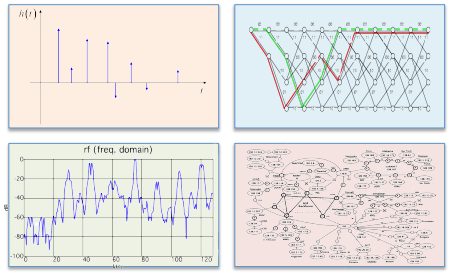


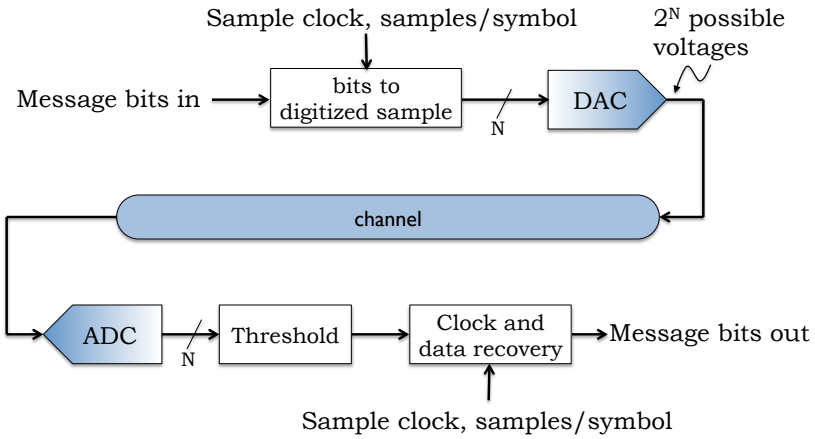
INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Spring 2011
Lecture #3

- Analog woes, the digital abstraction
- Basic digital recipes for sending information

Diagram of a Communication Channel



Representing information with voltage

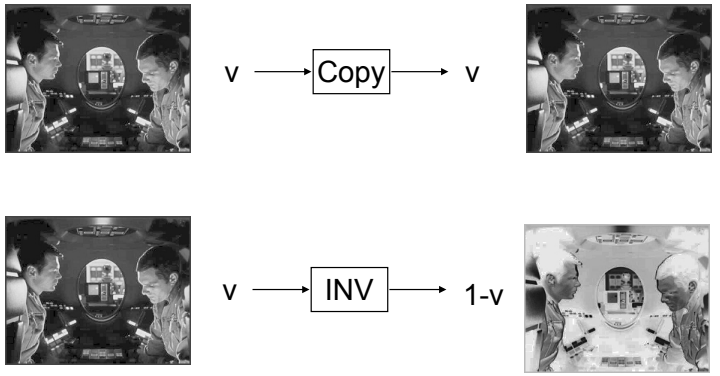
Representation of each point (x, y) on a B&W Picture:

0 volts: BLACK
1 volt: WHITE
0.37 volts: 37% Gray
etc.

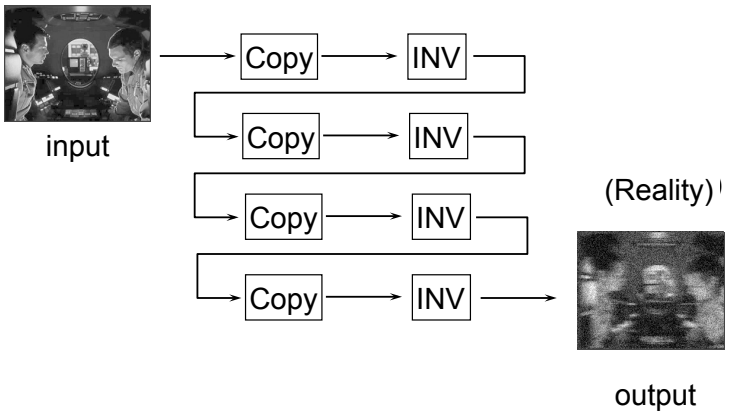
Representation of a picture:
Scan points in some prescribed raster order… generate voltage waveform

System Building Blocks

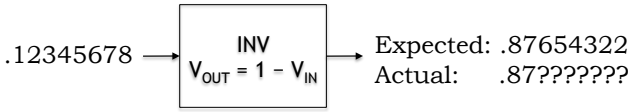
- First let's introduce some processing blocks:



Let's build a system!



Analog Woes

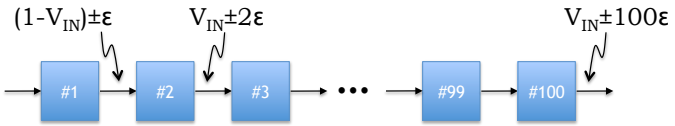


The actual value of V_{OUT} depends on many factors:

- Manufacturing tolerance of internal components
- Environmental factors (temp, power supply voltage)
- External influences (EM effects that affect voltages)
- How long we're willing to wait
- How much we're willing to spend

Truth in advertising: $V_{OUT} = (1 - V_{IN}) \pm \varepsilon$ If we call it ε maybe it'll seem small ☺

Analog Errors Accumulate

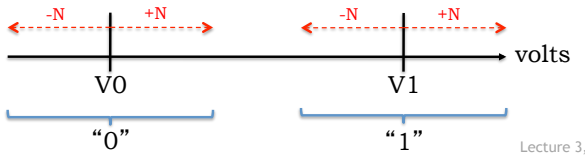


- If, say, ε = 1%, then result might be 100% off (urk!)
- Accumulation is good for money, bad for errors
- As system builders we want to guarantee output without having to worry about exact internal details
  - Bound number of processing stages in series OR
  - Figure out a way to eliminate errors at each processing stage. So how do we know *which part of the signal is message and which is error?*

Digital Signaling: Transmitting

To ensure we can distinguish signal from noise, we'll encode information using a fixed set of discrete values. For example, in a binary signaling scheme we would use two voltages (V0 and V1) to represent the two binary values "0" and "1".

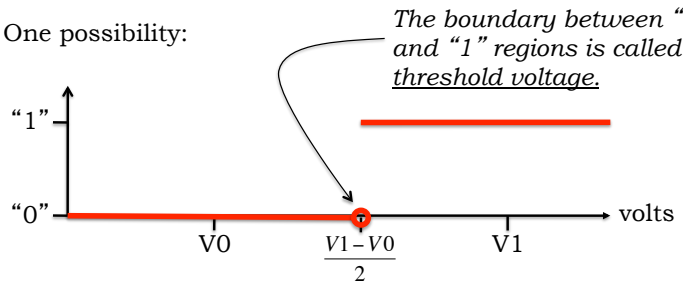
- voltages near V0 would be interpreted as representing "0"
- voltages near V1 would be interpreted as representing "1"
- if we would like our encoding to work reliably with up to ±N volts of noise, then we can space V0 and V1 far enough apart so that even noisy signals are interpreted correctly



Digital Signaling: Receiving

We can specify the behavior of the receiver with a graph that shows how incoming voltages are mapped to "0" and "1".

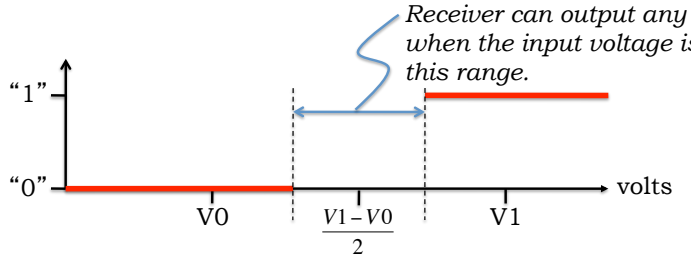
One possibility:



It would be hard to build a receiver that met this specification since it's very expensive and time consuming to accurately measure voltages (e.g., those near the threshold voltage).

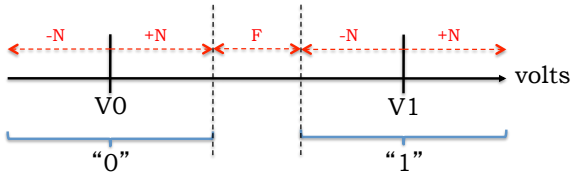
We Need a "Forbidden Zone"

We need to change our specification to include a "forbidden zone" where there is *no mapping* between the continuous input voltage and the discrete output:



Now the specification has some "elbow room" which allows for manufacturing and environmental differences from receiver to receiver.

Digital Signaling: Final Specification



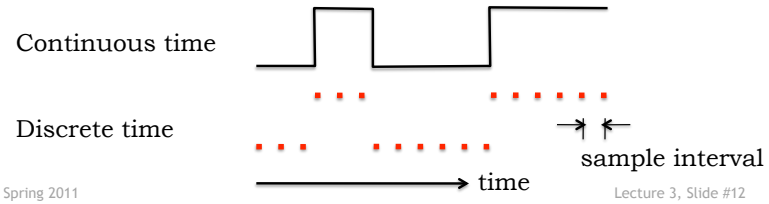
Engineering tradeoffs when choosing F, the width in volts of the forbidden zone:

Smaller F: allows larger N (better noise tolerance), but receiver is more expensive to build (tighter manufacturing and environmental tolerances).

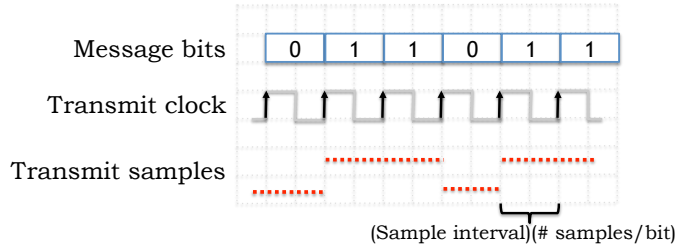
Larger F: less noise tolerance, but cheaper, faster receivers.

Digital Signaling in 6.02

- In 6.02 we'll represent voltage waveforms using sequences of voltage samples
  - Sample rate specifies the number of samples/second and hence the time interval between samples, e.g., 4e6 samples/second (4 Msps) means the time interval between samples is .25e-6 seconds (250ns).
  - Each transmission of a single bit ("0" or "1") will entail sending some number of consecutive voltage samples (V0 or V1 volts); we'll choose an appropriate number of samples/bit in each application. Goal: smaller is better!

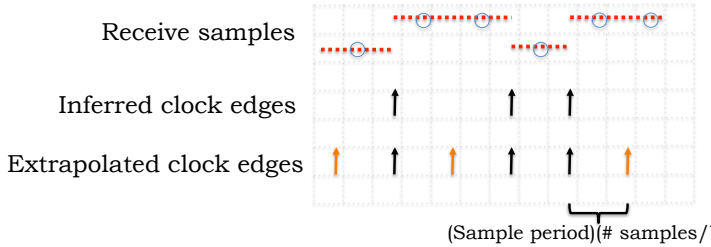


Transmitting Information



- Periodic events are timed by a clock signal
  - Sample period is controlled by the sample clock
  - Transmit clock is a submultiple of the sample clock
- Can receiver do its job if we only send samples and not the transmit clock?
  - Save a wire and the power needed to drive clock signal

Clock Recovery @ Receiver



- Receiver can infer presence of clock edge every time there's a transition in the received samples.
- Using sample period, extrapolate remaining edges
  - Now know first and last sample for each bit
- Choose "middle" sample to determine message bit

Two Issues for Recitation

- Don't want receiver to extrapolate over too long an interval
  - Differences in xmit & rcv clock periods will eventually cause receiver to mis-sample the incoming waveform
  - Fix: ensure transitions every so often, even if transmitting all 0's or all 1's (key idea: recoding)
- If recovered message bit stream represents, say, 8-bit blocks of ASCII characters, how does receiver determine where the blocks start?
  - Need out-of-band information about block starts
  - Fix: use special bit sequences to periodically synchronize receiver's notion of block boundaries. These sync sequences must be unique (i.e., distinguishable from ordinary message traffic).

Summary

- Analog signaling has issues
  - Real-world circuits & channels introduce errors
  - Errors accumulate at each processing step
- Digital Abstraction
  - Convention for analog signaling that lets us distinguish message from errors; restore signal at each step
  - Noise margins and forbidden zones
  - Recover digital data by comparing against threshold
- Receiver design
  - We don't send xmit clock, receiver does clock recovery
  - Determine bit from samples in "middle" of bit cell