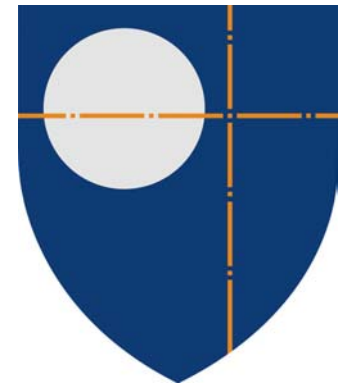


# EE472: Senior Design Project



Analog Subthreshold CMOS  
Probability Gates



Matt Aldrich  
5/04/04



# CMOS Probability Circuits - Introduction

- EE472 Senior Project
- Analog Subthreshold CMOS Gates: Characteristics and Applications
  - Motivation
  - Background Theory
  - Subthreshold Transistor Operation: Basic CMOS Devices
    - Subthreshold operation
    - Adding currents, walkthrough of classic quadrant multiplier
  - Application: Analog “Block Code” Decoder
  - Conclusion/Thanks



# Motivation

- Setting the stage: Analog decoding motivation
  - Block Codes, Convolutional Codes, Turbo Codes
    - Analog vs Digital: cost, size, power, speed, accuracy
  - Decoding as a decision making process
    - Decoder attempts to figure out whether received information is correct or not: may give an “estimate” of how accurate – probability
  - Analog CMOS VLSI
    - Using circuits’ non-linearities to add and multiply



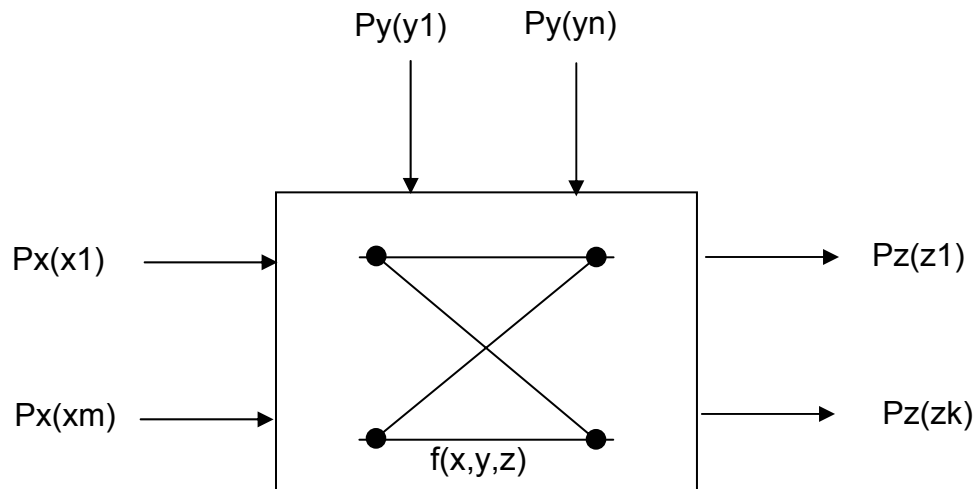
## Motivation (2)

- Decoding: A decision process
  - Suppose we receive a piece of data and it is unclear what it is?
    - How do we figure this out?
  - If the piece of data came with other pieces of data, and we could assure ourselves to a certain degree that those other pieces weren't contaminated, then we can accurately guess what that first piece was meant to be.
  - This is accomplished with the sum-product rule: Probabilities rather than guesses are added and multiplied to assert the “correctness” of data
- Implementing a solution
  - Digital Solution: multiplies and adds
  - Analog Solution: manipulate characteristics of BJTs & Subthreshold CMOS to achieve these multiplies and adds
- Previous work, project basis:
  - Loeliger et al, Analog Decoding research
  - Mead's bio inspired subthreshold analog VLSI work



# Calculating Using Probabilities

- Probabilistic decoding operations can be defined in terms of one generic unit
  - Theory of operation based upon the Sum Product Algorithm
  - Operation allows us to sum and multiply past, current, and future probabilities
  - Form the basis of all analog decoding networks

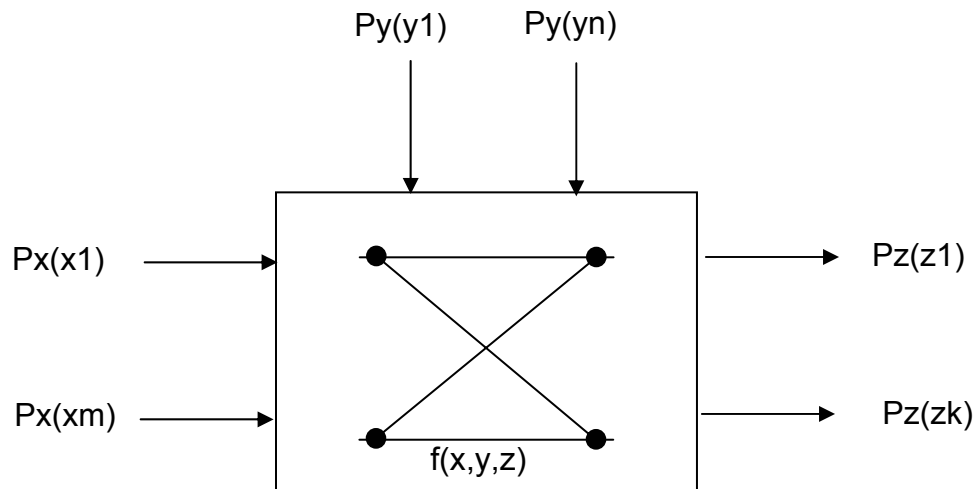




## Calculating Using Probabilities (2)

$$p_z(z) = \gamma \sum \sum p_x(x) p_y(y) f(x, y, z)$$

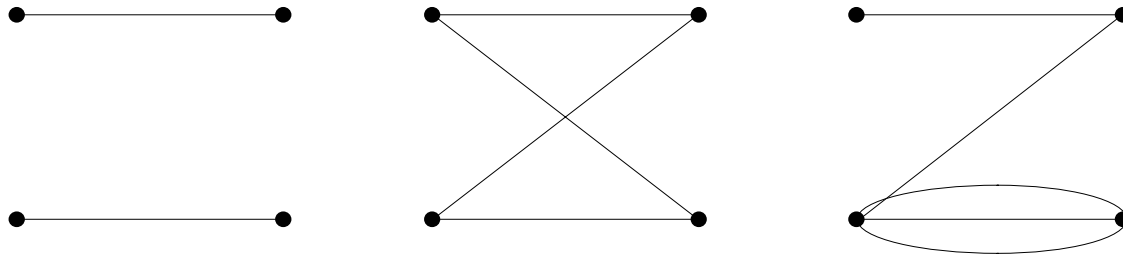
Scale factor gamma ensures  $p_z(z) = 1$   
 $f(x,y,z) = \{0,1\}$  = defined or undefined





# Soft Logic Building Blocks (Boolean Logic)

- The basis of analog decoding
  - Putting  $p_z(z) = \gamma \sum_x \sum_y p_x(x) p_y(y) f(x, y, z)$  to use:
  - The function,  $f(x, y, z)$  is defined iff  $f(x, y, z) = 1$
- Trellis form of blocks – essentially a state diagram





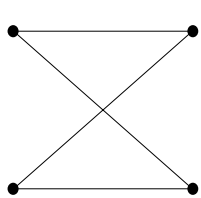
## Soft Logic Building Blocks (Boolean Logic) (2)

- A) Equal Gate – quick probability multiply

- ————— •  $f(x, y, z) = 1 \text{ iff } x = y = z$

- ————— • 
$$\begin{bmatrix} p_Z(0) \\ p_Z(1) \end{bmatrix} = \gamma \begin{bmatrix} p_X(0)p_Y(0) \\ p_X(1)p_Y(1) \end{bmatrix}$$

- B) XOR Gate – probability multiply and add



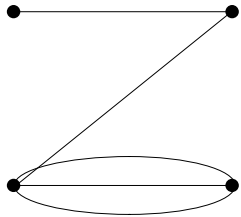
$$f(x, y, z) = 1 \text{ iff } z = x \oplus y$$

$$\begin{bmatrix} p_Z(0) \\ p_Z(1) \end{bmatrix} = \gamma \begin{bmatrix} p_X(0)p_Y(0) + p_X(1)p_Y(1) \\ p_X(0)p_Y(1) + p_X(1)p_Y(0) \end{bmatrix}$$



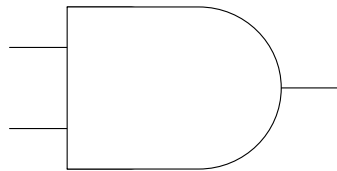
## Soft Logic Building Blocks (Boolean Logic) (3)

- C) Backwards AND – allows forward and backward reasoning



$$f(x, y, z) = 1 \text{ iff } z = x + y$$

$$\begin{bmatrix} p_Z(0) \\ p_Z(1) \end{bmatrix} = \gamma \begin{bmatrix} p_X(0)p_Y(0) + p_X(0)p_Y(1) \\ p_X(0)p_Y(1) + p_X(1)p_Y(1) \end{bmatrix}$$



**Example of backwards reasoning, not likely to happen in decoding situations**



# Theory Recap

- Next up – How do we implement this  $Pz(z=\{0.1\})$  notation?
- Recap
  - Stage is set: have a method of doing probability calculations
    - Equal Gate: fan in, fan out, **multiplication of two signals**
    - XOR: **multiplication of two signals, addition of similar types**
  - **We have a method for adding and multiplying probabilities, lets now investigate analog techniques for these gates!**



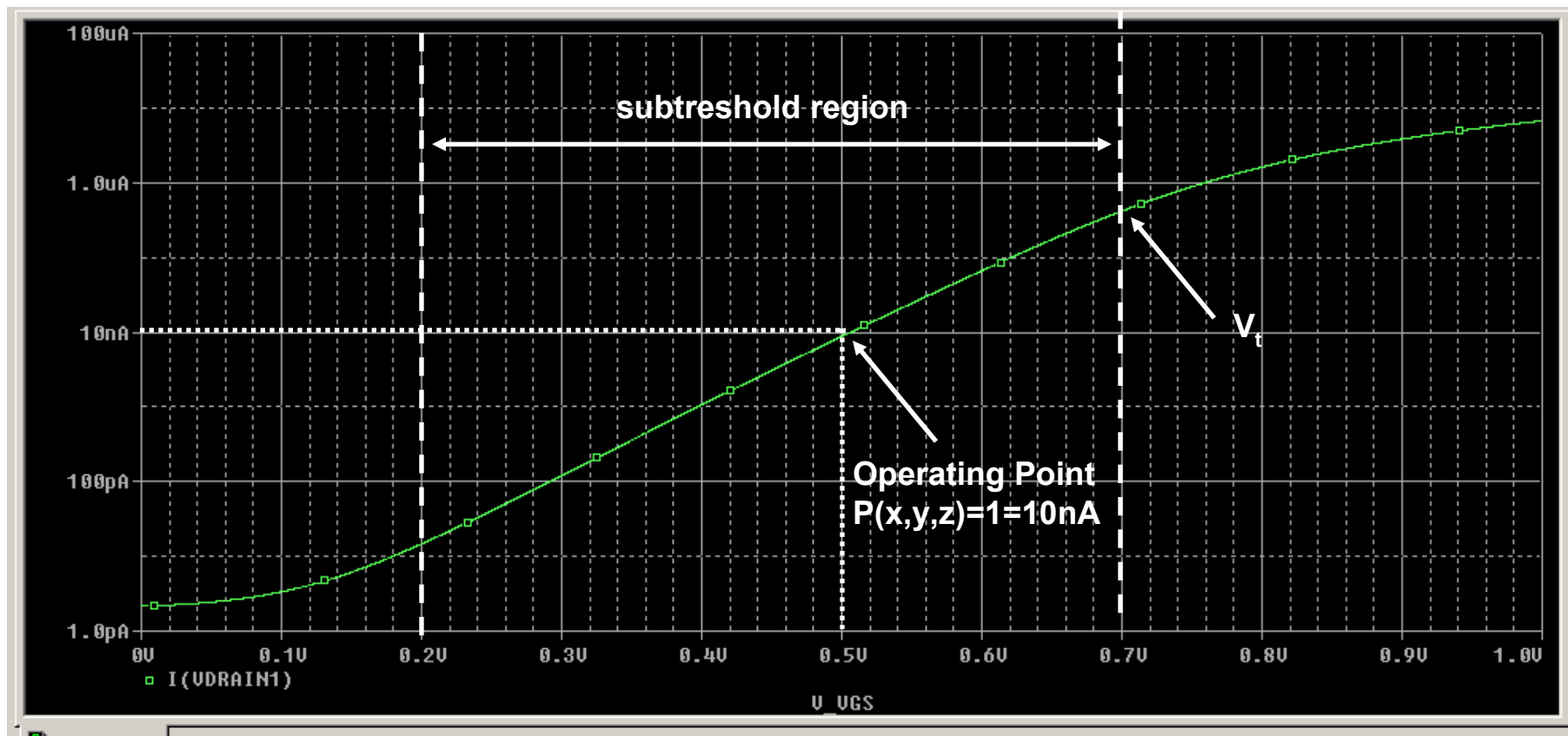
# Subthreshold Transistor Operation (Overview)

- How do we perform sum and multiply functions discussed earlier?
  - Addition of currents: Kirchoff's current law – sum of currents into node=0
    - Short a wire, add a current!
  - Multiplication of currents: Gilbert Multiplier
    - BJT: used exponential, logarithmic properties to achieve current multiplication
    - CMOS transistors operating in the subthreshold region achieve this type of response
- Not possible without Carver Mead & previous research on bio circuits
  - Mead proposed early model of drain current in subthreshold region
    - Subthreshold region: positive charge at gate *almost* balance by negatively charged depletion regions
      - Simple: transistor is “barely” on, we operate below  $V_t$ , and hence at current levels on the order of nanoamps to microamps.
      - Further simplify equations: make  $V_{ds} > 4 \cdot V_t$  = ensure operation in saturation region (recall linear/saturation operating regimes)



# Subthreshold Region – Operating Point

NMOS (saturation  $V_{gs} > 4 \cdot V_t$ )  $W/L = 3.2\mu/1.6\mu$  in MOSIS 1.5 $\mu$  process



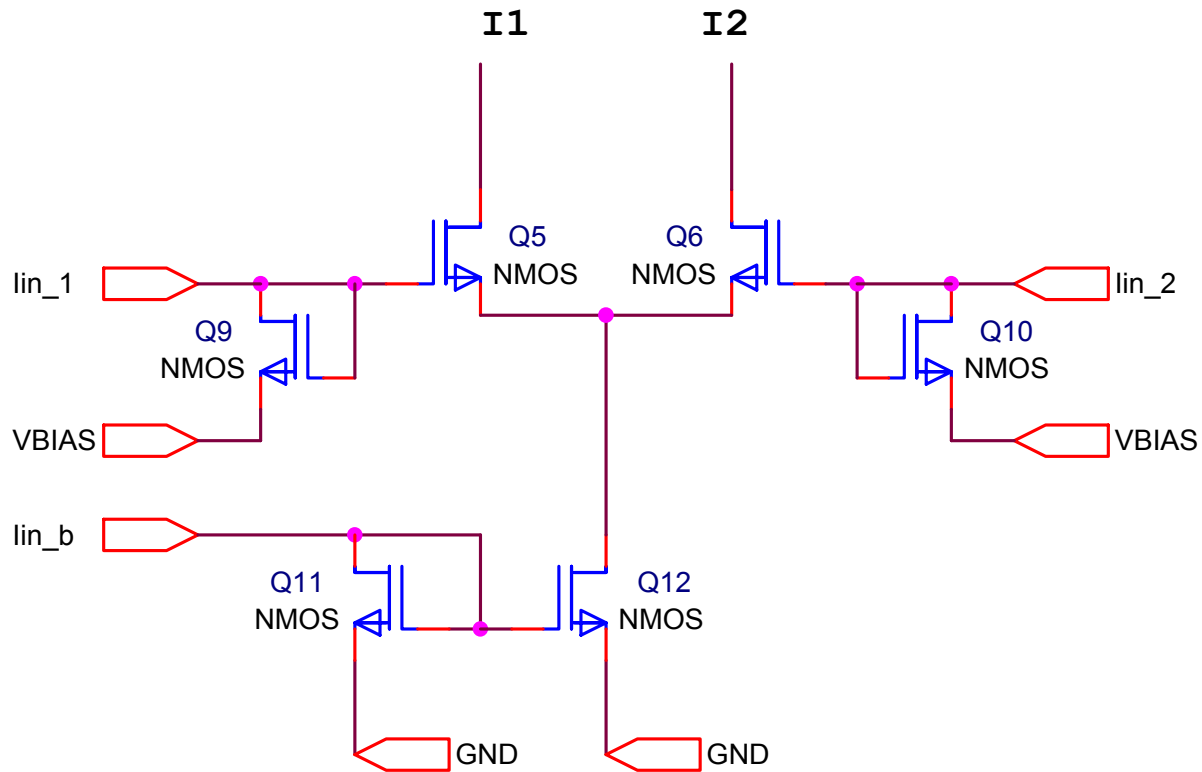


# A Subthreshold Gilbert Multiplier (in 1 minute!)

- Diode Connected Transistors
  - Logarithmic component of multiplier (ensure saturation at gate)
  - Form basis for current mirrors
    - Current mirrors: sink or source current, in and out of the circuit by using NMOS or PMOS
- Differential Pairs
  - Have logarithm component, need current multiplier with an exponential response
    - We need a circuit that can take an exponential of a voltage and outputs a current
    - Use a differential pair of CMOS transistors!



# Presto – A Gilbert Multiplier!

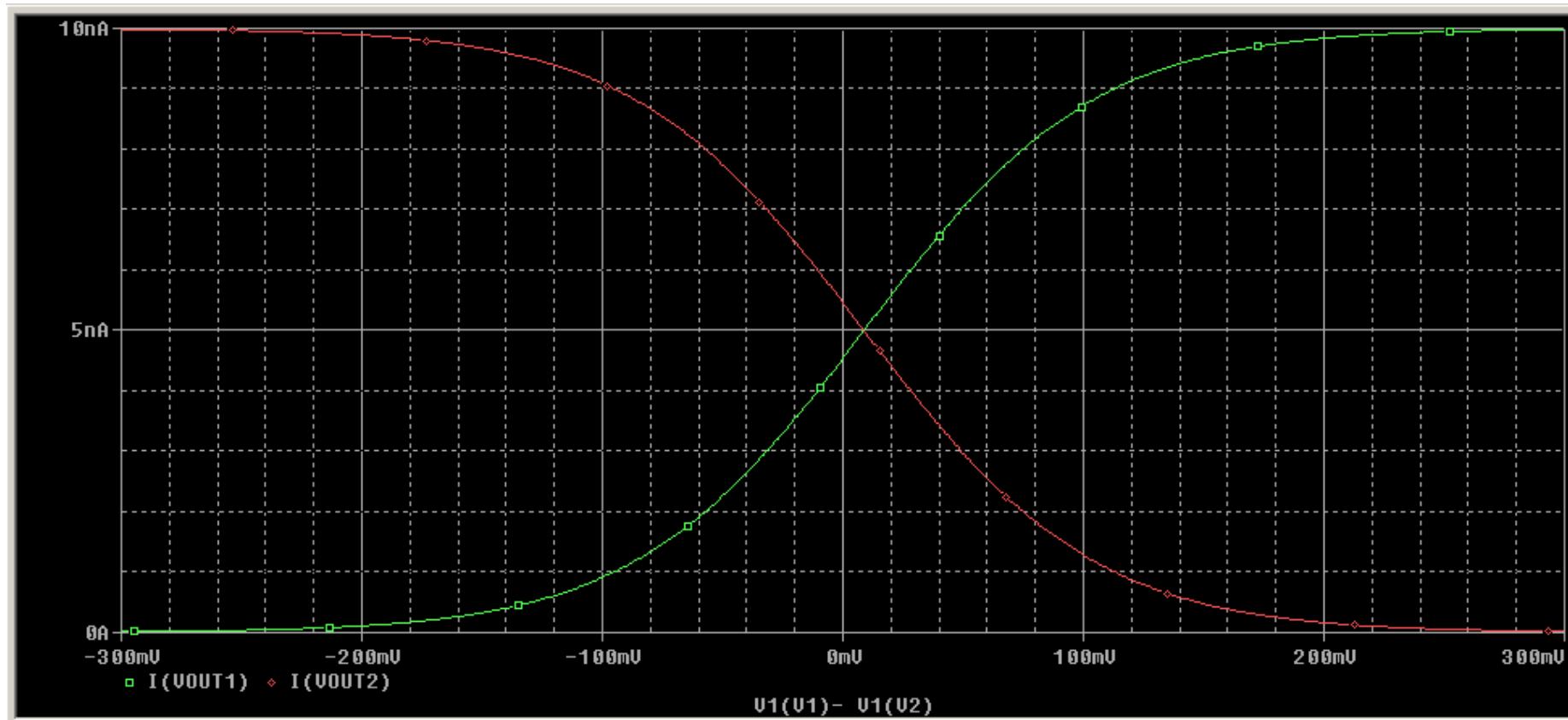


$$I_1 = \frac{I_{inb} I_{in1}}{I_{in1} + I_{in2}}, I_2 = \frac{I_{inb} I_{in2}}{I_{in1} + I_{in2}}$$
$$I_{in1} + I_{in2} = 1 \rightarrow I_1 = I_{inb} I_{in1}, I_2 = I_{inb} I_{in2}$$



# Response of the Differential Pair (1.5u process, min size)

$$I_1 = I_b \frac{e^{V_1}}{e^{V_1} + e^{V_2}} \Rightarrow \frac{e^{V_s} e^{V_1}}{e^{V_1} + e^{V_2}}, \text{ where } I_b = I_d = I_o e^{\frac{V_g - V_s}{V_t}}$$





# Boolean Soft Logic Gates ↔ Transistor Level Implementation

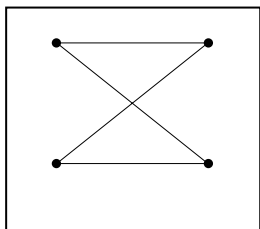
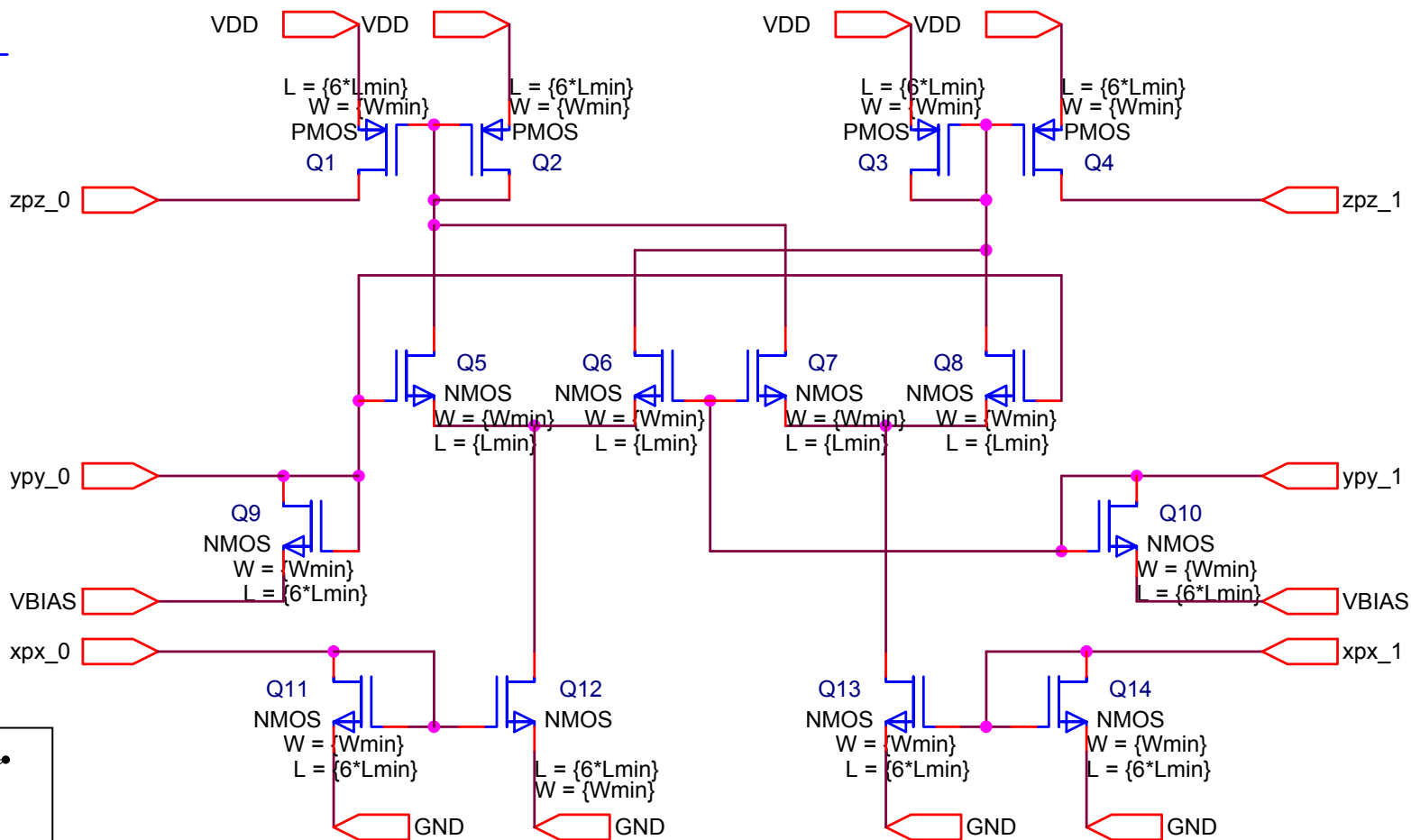
- At the crossroads
  - we know how to add and multiply probability distributions
  - We also know how to add and multiply currents
  - Probability of a bit being a 1 or 0 is now a current level with  $p(b=1)=10nA$
- Draw parallels between trellis diagrams and circuits
  - Look at two simple two bit examples:
    - Soft-XOR gate
    - Soft-Equal gate
  - What do we care about, how do we measure performance?
    - Care about transient response to a current pulse – shows reaction of circuit
    - What's important: Power supply, current draw, transistor sizing, output rise time
    - We compare these specifications to digital adds and multiplies, tradeoffs
    - Process vs. performance – feature size will be biggest factor in rise time!



# Soft-XOR Gate

## PARAMETERS:

$L_{min} = 1.6\mu$   
 $W_{min} = 3.2\mu$





# Soft XOR – Design Notes

- Design Explanations
  - Supply voltage & Transistor Width
    - Several simulations were run in order to determine “best” supply voltage
    - 1.5v and a bias of (1/4) of supply (375mV) on a two input XOR yielded a power consumption of 30nW (current draw ~20nA)
    - Sizing of transistors – minimum width – ensure low capacitive effects (fast response). Current mirrors – 6 times minimum length, to avoid short channel effects (which spoil current sourcing properties).
- Test Case
  - XOR response – test cases, for simplicity used full probabilities. IE to see a response of 1 on the output, input probabilities were  $p(x=0) = 1$  &  $p(y=1)=1$
  - Circuit schematic was “loaded” with an identical copy of itself – similar to testing a digital inverter, additional copies are used to properly load circuit.

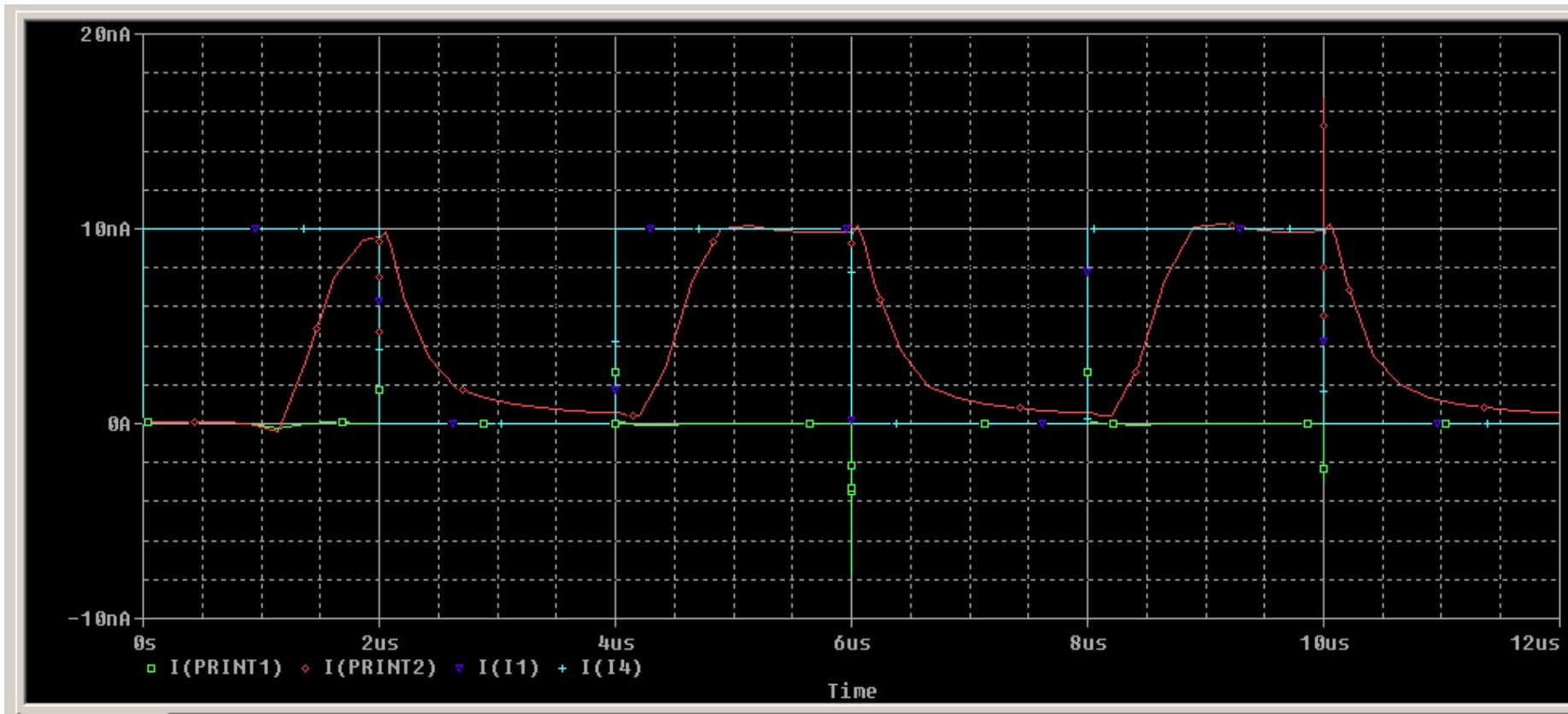


# Soft XOR Transient Response (1)

$$p_x(x_0 = 1) \oplus p_y(y_1 = 1) = p_z(z_1 = 1)$$

Variable I(PRINT2) = Pz(z1=1)

Delay t=0us-2us due possibly to capacitance charging of transistors. Rise =750ns, Fall= 250ns = 1MHz



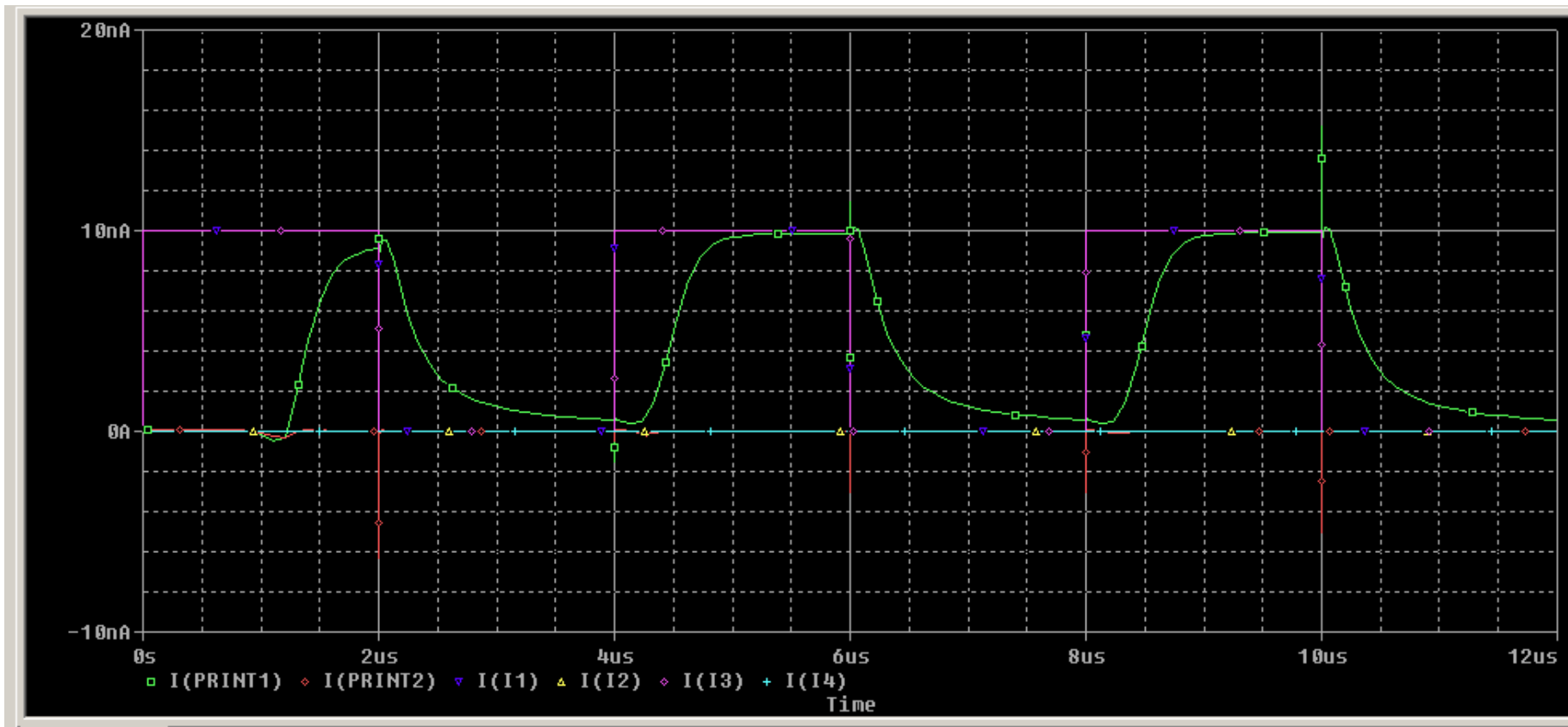


## Soft XOR Transient Response (2)

$$p_x(x_0 = 1) \oplus p_y(y_0 = 1) = p_z(z_0 = 1)$$

Variable I(PRINT1) = Pz(z0=1)

Delay t=0us-2us due possibly to capacitance charging of transistors. Rise =750ns, Fall= 250ns = 1MHz



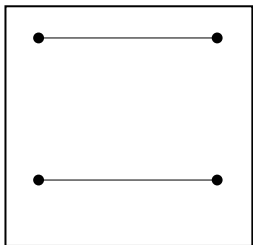
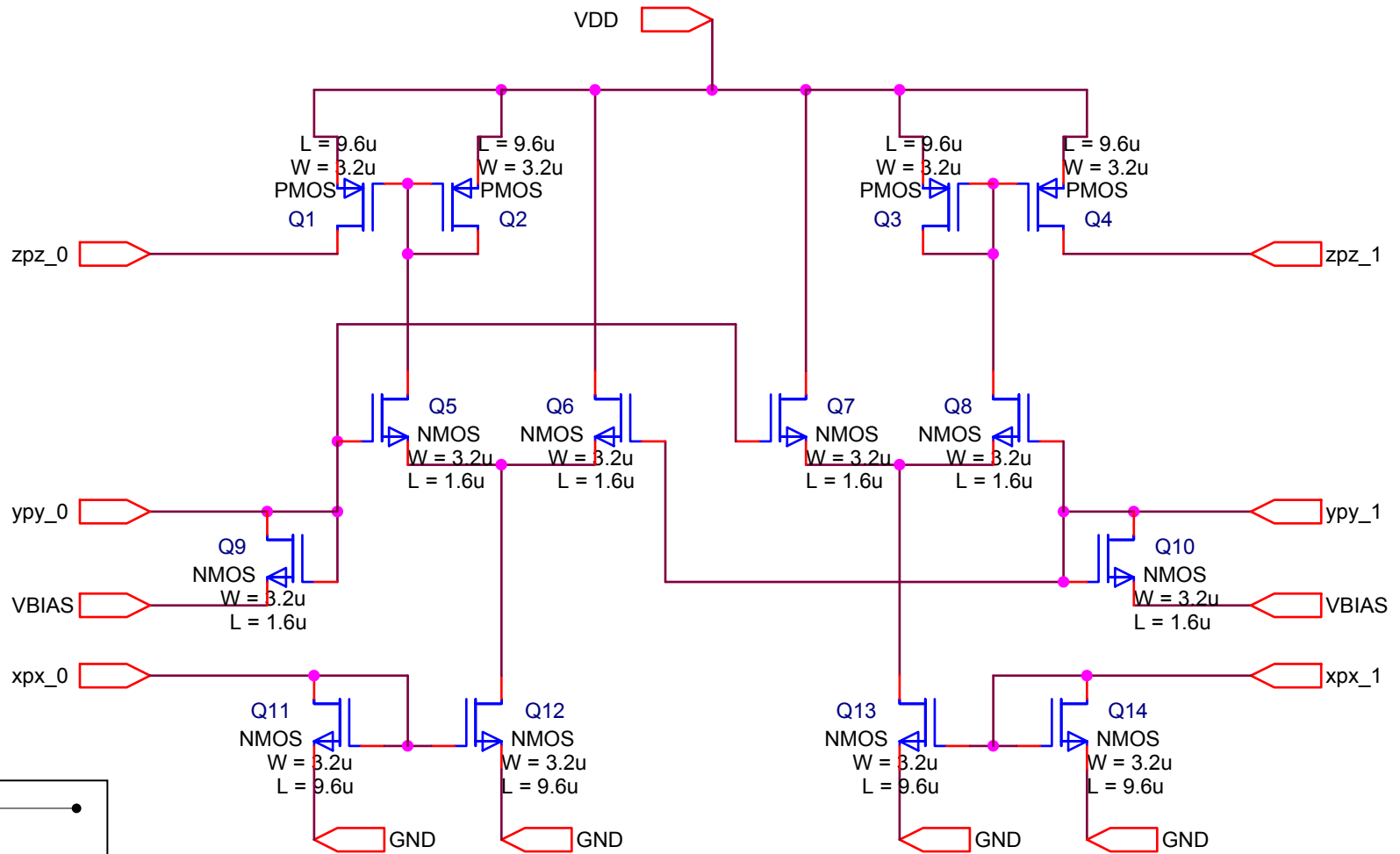


# Soft XOR Results

- Output Response
  - Low power consumption: 30nW
  - 1MHz clocking speed for minimum sized transistors
    - Note: long transistor design increases capacitance
    - Possible explanation on slow initial response
  - Robust: Behavior accurately models our probability function  $z$  for an XOR
  - Analog Vs. Digital (fast clocking, modern process = .5u Itanium 2<sup>nd</sup> gen, assume only 1 FP Multiplier)
    - At a frequency of 1MHz, we achieve 4 multiplies and 2 adds
    - Add 2 bits = <1ns/clock, 1 clock cycle: 2 clock cycles = 4ns for addition
    - Multiply 2 bits (four times) < 3ns/clock, 3 clock cycles: 12 clock cycles = 36ns for mult
    - Digital = 40ns XOR operation, Analog = 1us XOR operation
    - 25Mhz vs 1Mhz
    - Have faith! We're simulating in an outdated educational 1.5u process!



# Equal Gate





# Equal Gate– Design Notes

- Design Explanations – Same as Soft XOR
  - Supply voltage & Transistor Width
    - Several simulations were run in order to determine “best” supply voltage
    - 1.5v and a bias of (1/4) of supply (375mV) on a two input XOR yielded a power consumption of 7.5pW (current draw 4pA)
    - Sizing of transistors – minimum width – ensure low capacitive effects (fast response). Current mirrors – 6 times minimum length, to avoid short channel effects (which spoil current sourcing properties).
- Test Case
  - XOR response – test cases, for simplicity used full probabilities. IE to see a response of 1 on the output, input probabilities were  $p(x=1) = 1$  &  $p(y=1)=1$
  - Circuit schematic was “loaded” with an identical copy of itself – similar to testing a digital inverter, additional copies are used to properly load circuit.

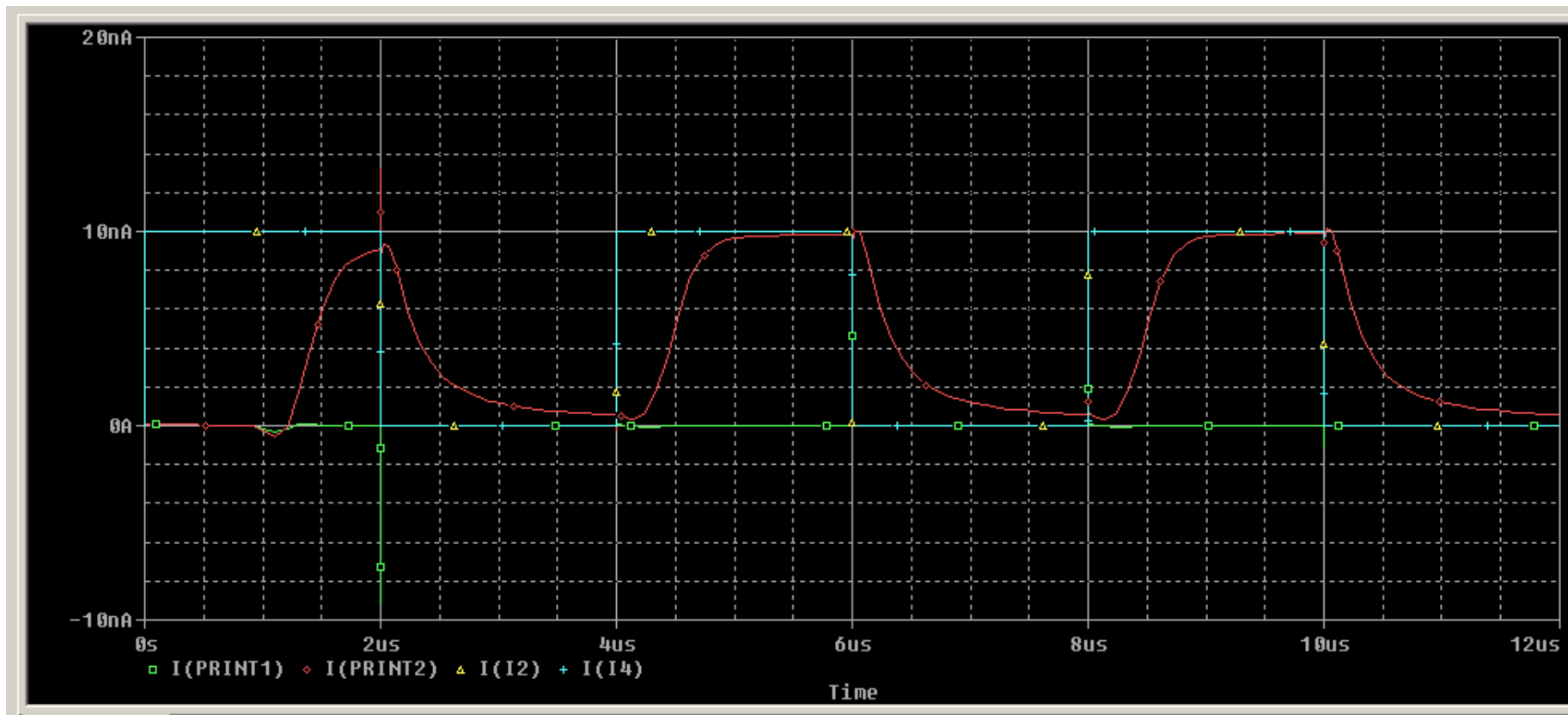


# Equal Gate – Transient Response

$$p_x(x_1 = 1)p_y(y_1 = 1) = p_z(z_1 = 1)$$

Variable I(PRINT2) = Pz(z1=1)

Delay t=0us-2us due possibly to capacitance charging of transistors. Rise =750ns, Fall= 250ns = 1MHz



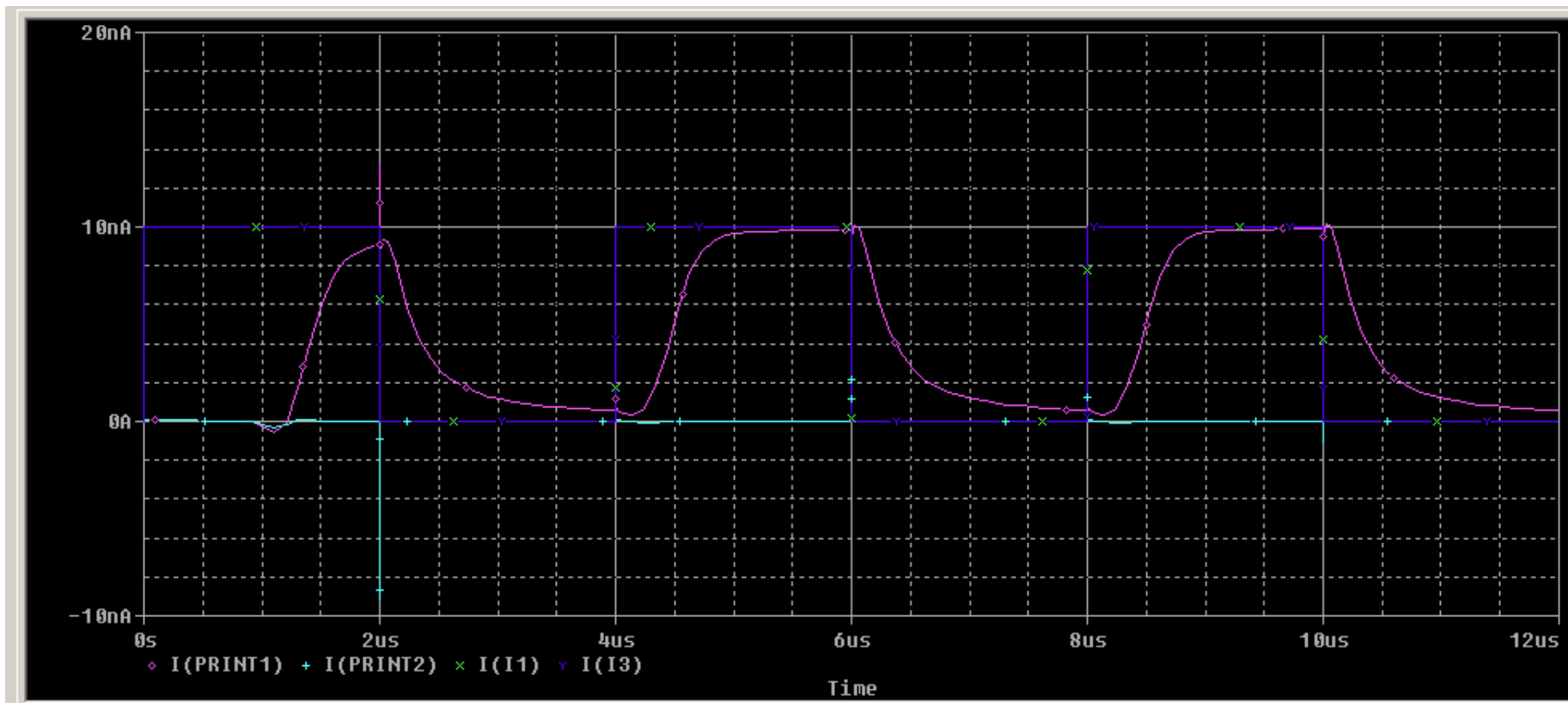


## Equal Gate – Transient Response (2)

$$p_x(x_0 = 1)p_y(y_0 = 1) = p_z(z_0 = 1)$$

Variable I(PRINT1) = Pz(z0=1)

Delay t=0us-2us due possibly to capacitance charging of transistors. Rise =750ns, Fall= 250ns = 1MHz





# Equal Gate Results

- Output Response
  - Low power consumption: 7pW
  - 1MHz clocking speed for minimum sized transistors
    - Note: long transistor design increases capacitance
    - Possible explanation on slow initial response
  - Robust: Behavior accurately models our probability function  $z$  for an Equal
  - Analog Vs. Digital (modern process, assume only 1 FP Multiplier)
    - At a frequency of 1MHz, we achieve 2 multiplies
    - Multiply 2 bits  $< 3\text{ns/clock}$ , 3 clock cycles:  $9\text{ns/mult} = 18\text{ns}$  for 2 mults.
    - Digital = 18ns Equal operation, Analog = 1us Equal operation
    - 55Mhz vs 1Mhz



## Application – A Block Code Decoder

- Example in Lustenberger thesis, implemented using BJTs
  - Simulation beyond scope of this presentation, possible inclusion in write up to demonstrate performance of building blocks
  - Consider the following:
    - Two bits are encoded into code words 5b in length. Assume the code words have a Hamming distance of 3 to reduce errors
    - Want to see whether encoded bits are received properly
    - Assume to accurately describe all code words we need 2 equal gates and 2 XOR
    - However, not all multiplies are performed at the same time, and not all additions are performed at the same time
    - Hence – output of info bit probabilities will depend largely on the 1us gate delay and how many gates we will use
  - Circuits more elaborate: involve 3 inputs in some cases
  - Circuit application to be included in write up.



# Putting it in Perspective

- Power Consumption is Key
  - What is the intended application for these decoders?
    - Possibility: Environments where speed is not an issue and low power consumption is critical – Cell Phones
  - Example scenarios compared results against processors >1Ghz clocking
  - A smaller process = less power(!), smaller feature sizes, less capacitance => greater speed
- Tradeoffs
  - Certainly quite a few, subthreshold CMOS circuits – an easy way to run up design costs, Commercial designs that implement subthreshold operation?
    - Alternative, BJT implementation: huge speed improvement, area increase.



## Conclusion / Gains

- Introduced basic theory to probability propagation – adds/multiplies
- Discussed CMOS operating region
- Developed (quickly) circuits to implement probability propagation and listed most large signal current outputs
- Introduced analog decoder building blocks XOR and Equal, AND and gave circuit representations of XOR and Equal for analog decoders
- Gave initial performance estimates of XOR and Equal given response to a unit pulse
  - Included max operating frequency, power, and design tradeoffs including a first order comparison to digital implementations
- Discussed possible application and necessary gate structures
- Mastered Orcad PSPICE design tool, practiced classical circuit analysis, learned some application specific probability and basic info theory



# Thank You

- Professor Richard Lethin
- Professor Peter Kindlmann
- Mike Bell
  
- Yale EE Department