# EECS 427
# Lecture 6: Project architecture and intro to logical effort

Reading: handouts



# Last Time

- Interconnect discussion
  - Technology scaling leads to wires becoming more important
    - Smaller transistors are faster but smaller wires are slower (*reverse scaling*)
  - C or RC models are used
    - Whether to include R depends on ratio of wire resistance to device resistance
  - Capacitive crosstalk
    - Interference effects between adjacent wires
    - Can cause major changes in delay or functional glitches
    - Avoid running wires next to each other with small spacing for long distances
  - Repeaters are inserted uniformly to reduce RC delay
  - Cascaded inverter chain is used to drive a very large capacitive load effectively/quickly

# Lecture Overview

- Project architecture description (handout)
- Introduction to logical effort



# Project architecture

- 2-stage pipeline, 1 word per instruction
  - 1st stage of pipe: instruction fetch (IF)
  - 2nd stage: instruction decode (ID), execute (EX)
  - You can alter this but it's not as easy as it looks
- 16-bit words, with four 4-bit components
  - Most significant 4 bits are the operation code (opcode)
  - Tells which instruction (e.g., ADD, MOV, STOR) is to be performed
  - Next 4 bits give the register address to which the result of the instruction should be written (with a few exceptions)
  - Next 8 bits can contain several pieces of information:
    - Immediate data to be acted upon (rather than accessing this data from a register location)
    - Opcode extensions (since there are more than $2^4$ or 16 ops)
    - Address of source register to draw data from

# Example instructions

- Direct vs. immediate instructions
- *Add* Rsrc Rdest
  - Rdest ← Rdest *Add* Rsrc
  - Where Rdest and Rsrc are register addresses
- *Add* Imm Rdest
  - Rdest ← Rdest *Add* Imm
  - Where Imm is 8 bits of data (not an address)
- Typical instructions:
  - MOV moves data from 1 reg location to another
  - LOAD loads data from memory to the RF
  - STOR writes data to memory
  - Control flow instructions (conditional branches, jumps, jump and link)
- Look over baseline instructions and extra instructions, think about target application
- Weste 3rd edition handout is useful as overview of a processor architecture (note it does not exactly reflect our own architecture)



# Logical Effort: Outline

- Motivation
- Model the delay of one gate
- Next time:
  - The delay of a chain of gates (multistage)
  - Branching
  - Minimum delay
  - Best number of stages and gate sizing
  - Examples
  - Limitations

# Logical Effort motivation

- Sizing of a chain of inverters
  - Geometric progression
- How about more complex logic?
- Logical Effort objectives:
  - Quick & dirty, back of the envelope sizing
  - Make trade-off between circuits
- Reference:
  - I. Sutherland, B. Sproull, D. Harris, *Logic Effort - designing fast CMOS Circuits* Academic Press, 1999

# Delay scaling with device size

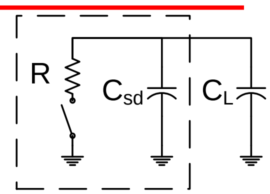
Let’s model an inverter with a resistor

$$D \propto R(C_{sd} + C_L)$$

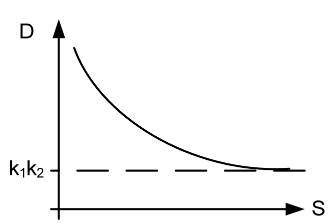
If you scale the inverter by a ratio of S

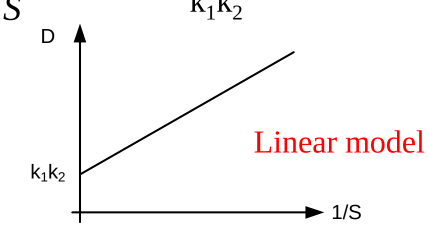
$$R = \frac{k_1}{S} \qquad D \approx \frac{k_1}{S}(k_2 S + Cout)$$

$$C = k_2 S \qquad D \approx k_1 k_2 + \frac{k_1}{S} Cout$$




The delay saturates to a value proportional to $k_1 k_2$



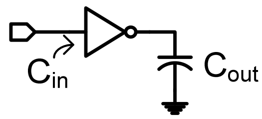
# Output / Input load

Let's model the delay as a function of the output / input load

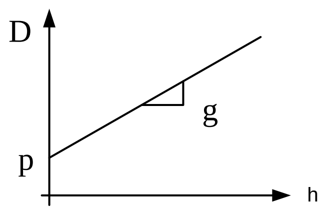
$C_{in} = k_3 S$

$D \approx k_1 k_2 + k_1 k_3 \frac{C_{out}}{C_{in}}$

Define $h = \frac{C_{out}}{C_{in}}$




$D \approx \underbrace{k_1 k_2}_{p} + \underbrace{k_1 k_3}_{g} \cdot h$




$D \approx p + g \cdot h$ $\quad f = g \cdot h$

p = Parasitic (intrinsic) delay
g = Logical effort
h = Electrical effort
f = Effort delay or Stage effort

# Units of effort

- Reference is the inverter:

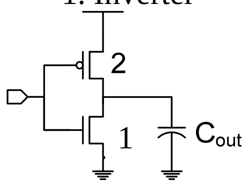
$$g_{inv} = 1\tau$$
$$p_{inv} = 0.6\tau$$

- g is a function of the complexity of a gate, not its size
- p is a function of the technology
- $\tau$ ~ 6.3ps in our 130nm process

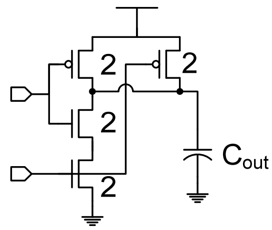
# Logical effort

- Logical effort of a gate is the ratio of its input capacitance to an inverter that delivers the same output current

1. Inverter




2. 2-input NAND




$$f_1 = f_2$$
$$g_1 h_1 = g_2 h_2$$
$$1\frac{C_{out}}{3C} = g_2 \frac{C_{out}}{4C}$$
$$g_2 = \frac{4}{3}$$
$$C_{sd,inv} = 3C$$
$$C_{sd,nand} = 6C$$
$$p_2 = 2p_{inv}$$

# Logical effort (g)

| | Number of inputs | | | | | |
|---|---|---|---|---|---|---|
| **Gate type** | **1** | **2** | **3** | **4** | **5** | **n** |
| Inverter | 1 | | | | | |
| NAND | | 4/3 | 5/3 | 6/3 | 7/3 | ( n + 2 )/3 |
| NOR | | 5/3 | 7/3 | 9/3 | 11/3 | ( 2n + 1 )/3 |
| Multiplexer | | 2 | 2 | 2 | 2 | 2 |
| XOR (parity) | | 4 | 12 | 32 | | |

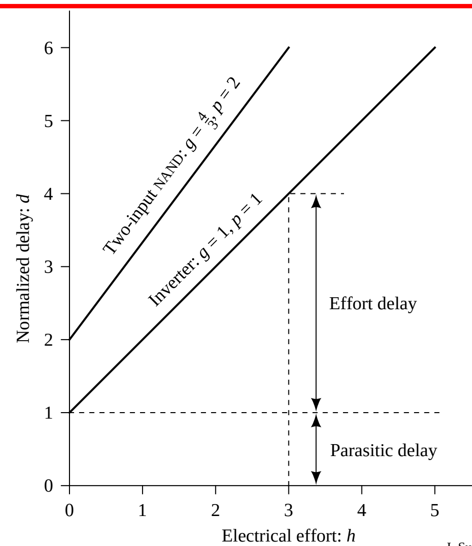
I. Sutherland, *et al*, Logical Effort, Academic Press, 1999

# Parasitic Delay (p)

| Gate type | Parasitic Delay |
|---|---|
| Inverter | $p_{inv}$ |
| n-input NAND | $n.p_{inv}$ |
| n-input NOR | $n.p_{inv}$ |
| n-way multiplexer | $2n.p_{inv}$ |
| XOR, XNOR | $4.p_{inv}$ |

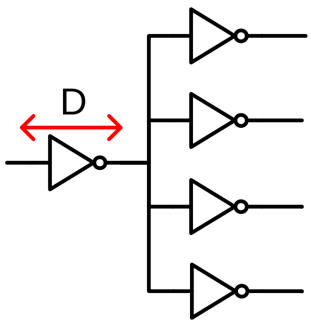
I. Sutherland, *et al*, Logical Effort, Academic Press, 1999

# Delay components




I. Sutherland, *et al*, Logical Effort, Academic Press, 1999

# F04 Example




$$C_{out} = 4C_{in}$$

$$h = 4$$

$$g = 1$$

$$p = 0.6$$

$$D = 4 + 0.6 = 4.6\tau$$

FO4 delay ~ L/3 ps where L = technology node in nm
Ex: 130nm → ~45ps, 45nm → 15ps