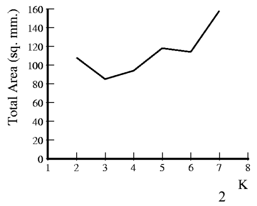
ESE534: Computer Organization

Day 2: January 18, 2012
Universality, Gates, Logic

Work Preclass Exercise

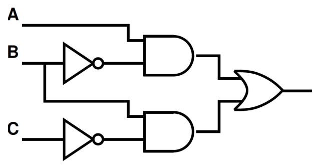
Last Time

- Computational Design as an Engineering Discipline
- Importance of Costs



Today

- Simple abstract computing building blocks
  - gates, Boolean Equations
  - RTL Logic (at least the logic part)
- Universality
- Logic in Gates
  - optimization
  - properties
  - Costs



Preclass 1

- Do the Case 1 circuits calculate the same thing?
- Case 2?

General

- How do we define equivalence?
  - How do we determine if two circuits are equivalent?

Model: Stateless Functions (Combinational Logic)

- Compute some "**function**"
  - $f(i_0,i_1,\ldots i_n) \rightarrow o_0,o_1,\ldots o_m$
- Each unique input vector
  - implies a particular, deterministic, output vector

Boolean Equivalence

- Two functions are equivalent when
  - They have the same outputs for every input vector
  - *i.e.*, they have the same truth table
- There is a **canonical** specification for a Boolean function
  - its Truth Table

Boolean Logic → Gates

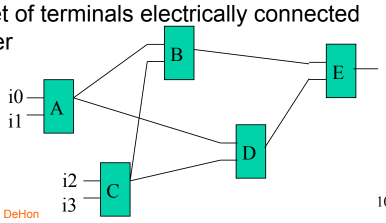
- Implement Boolean functions with gates
  - E.g. Problems 1 on preclass
- How does a gate relate to Boolean functions?

Implementation in Gates

- **Gate:** small Boolean function
- **Goal**: assemble gates to *cover* our desired Boolean function
  - Combine small Boolean functions to implement arbitrarily large Boolean function
- Collection of gates should implement *same* function
- *I.e.* collection of gates and Boolean function should have same Truth Table

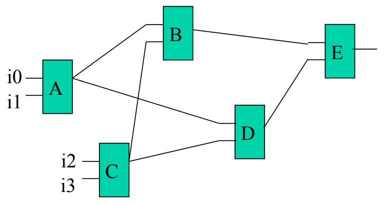
Netlist

- **Netlist:** collection of interconnected gates
  - A list of all the gates and what they are connected to
  - **Net**: set of terminals electrically connected together



Netlist

- **Netlist:**
  - A list of all the gates and what they are connected to



Netlist:
A=nand i0 i1
C=nand i2 i3
B=nand A B
D=nand A D
E=nand B D

Boolean Equations

- o=/a*/b*c+/a*b*/c+a*b*/c+a*/b*c
- Another way to express Boolean functions

a	b	c	o
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Terminology

- Literals -- a, /a, b, /b, ….
  - Qualified, single inputs
- Minterms --
  - full set of literals covering one input case
  - in y=a*b+a*c
    - a*b*c
    - a*/b*c

Boolean Equations

- o=/a*/b*c+/a*b*/c+a*b*/c+a*/b*c
- Truth table has one row for each **minterm**

a	b	c	o
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Boolean Equations

- o=/a*/b*c+/a*b*/c+a*b*/c+a*/b*c
- This particular equation has a an expression for every minterm whose output is a 1

a	b	c	o
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Boolean Equations

- Can be more compact:
  O=a*b*c*d*e+/a*/b*/c*/d*/e
- How many rows would truth table require?
- Often use in Register Transfer Language (RTL) expressions
  - Write logic (boolean equations) occur between registers

If's

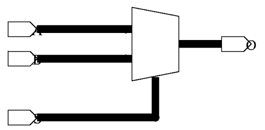
- If (a*b + /a*/b)
  c=d
- else
  c=e

How does this turn into Boolean logic?

- t=a*b+/a*/b
- c=t*d + /t*e

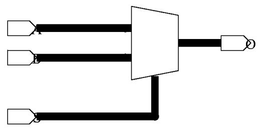
If→Mux Conversion

- Often convenient to think of IF's as Multiplexers
- If (a*b + /a*/b)
  c=d
- else
  c=e



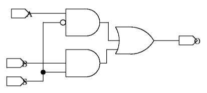
Muxes

- Mux:
  - Selects one of two (several) inputs based on control bit



Mux Logic

- Of course, Mux is just logic:
  - mux out = /s*a + s*b
  - Another small Boolean function



- Two views logically equivalent
  - mux view more natural/abstract when inputs are multibit values (datapaths)

Sum of Products

- o=/a*/b*c+/a*b*/c+a*b*/c+a*/b*c
- o=(a+b)(/b+/c)
  - a*b+a*/c+b*/c
- o=(a+/b)(b+c)+/b*/c
  - a*b+a*c+/b*c +/b*/c

Implementation

- How can I implement any Boolean function with gates?

Implementation

- Start with truth table
- Single output {0, 1}
  - Use inverters to produce complements of inputs
  - For each input case (minterm)
    - If output is a 1
      - Develop an AND to detect that case
        - » Decompose AND into gates
  - OR together all such minterms
    - Decompose OR into gates
- Multiple outputs
  - Repeat for each output

Universal set of primitives

- What primitives did we need to support previous implementation set?
- **Conclude:** can implement any Boolean function by a netlist of gates selected from a small set.
- **Homework (pr3):** How small can set be?

Engineering Goal

- Minimize resources
  - area, gates
- Exploit *structure* of logic
- “An Engineer can do for a dime what everyone else can do for a dollar.”

Minimum Sum of Products

- o=/a*/b*c+/a*b*/c+a*b*/c+a*/b*c

/b*c + b*/c

Minimum Sum of Products

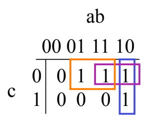
- o=(a+b)(/b+/c)

a*/b+a*/c+b*/c

a*/b+a*/c+b*/c

a*/b + b*/c

Term a*/c is redundant



c \ ab	00	01	11	10
0	0	1	1	1
1	0	0	0	1

Note: Preclass 1

Least Cost is not always MSP

- o=(a+b)(c+d) 3 2-input gates
  - a*b+a*c+b*c +b*d
  - (a*b+a*c)+(b*c+b*d) 7 2-input gates
- Product of Sums smaller…

Logic Optimization

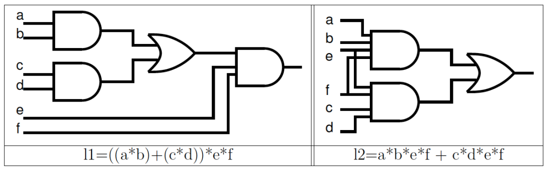
- There are many logical equivalent specifications for a function.
- Freedom to choose
- Exploit to select one that costs the least
- Potentially **different** from the one specified by the designer
  - Value of canonical form, equivalence

Cheapest?

- Which of the equivalent solutions is cheapest depends on the cost model.

Minimize Area (preclass 4)

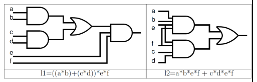
- Area minimizing solutions **depends** on the technology cost structure



31

Minimize Area (preclass 4cd)

- I1: ((a*b) + (c*d))*e*f
- I2: ((a*b*e*f)+(c*d*e*f))



- Area:
  - I1: 2*A(and2)+1*A(or2)+1*A(and3)
  - I2: 2*A(and4)+1*A(or2)
- all gates take unit area:
  - ❑ A(I2)=3 < A(I1)=4
- gate size proportional to number of inputs:
  - ❑ A(I1)=2*2+2+3=9 < A(I2)=2*4+2=10

32

Best Solution Depends on Costs

- This is a simple instance of the general point:

  …When technology costs change

  ➔ the optimal solution changes.

- In this case, we can develop an algorithm that takes the costs as a parameter.

33

Don't Cares

- Sometimes will have incompletely specified functions:

a	b	c	o
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	x
1	0	0	x
1	0	1	0
1	1	0	0
1	1	1	0

How should pick x's to minimize area?

34

Don't Cares

- Will want to pick don't care values to minimize implementation costs:

a	b	c	o
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	x
1	0	0	x
1	0	1	0
1	1	0	0
1	1	1	0

a	b	c	o
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

35

Logic Optimization

- Logic Optimization
  - Two Level Minimization
  - Covering w/ reconvergent fanout
- Can be formulated precisely and solved optimally
  - We can write programs to do this for us!
- Is formally a **hard** problem.

36

Logic Optimization is NP-hard

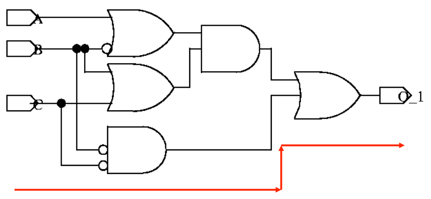
- Technically: NP-hard in general
  - **Informally:** *unlikely* we will be able to guarantee to solve the problem in time less than exponential in the number of inputs
  - **Practically:** 100s of inputs in seconds
    - Most problems not exponential
- Cover how to attack in an ESE535
  - can point you at rich literature
  - can find software to do it for you

Delay in Gates

- Simple model:
  - each gate contributes a fixed delay for passing through it
  - can be different delay for each gate type
  - *e.g.*
    - inv = 10ps
    - nand2=15ps
    - nand3=20ps
    - and2=25ps

Path Delay

- Simple Model: Delay along path is the sum of the delays of the gates in the path



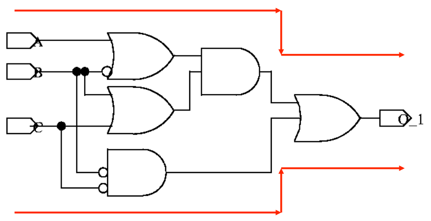
Path Delay = Delay(Or2)+Delay(Or2)

Critical Path

- Path lengths in circuit may differ
- Worst-case performance of circuit determined by the longest path
- Longest path designated **Critical Path**

Multiple Paths

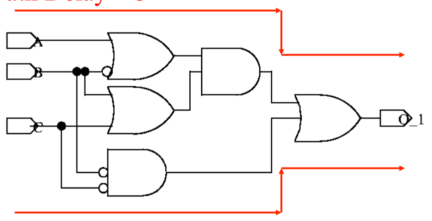
Path Delay = Delay(Or2i1)+Delay(And2)+Delay(Or2)



Path Delay = Delay(Or2)+Delay(Or2)

Critical Path = Longest

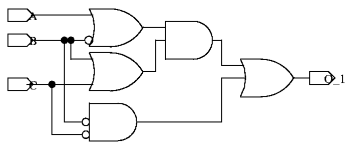
Path Delay = 3



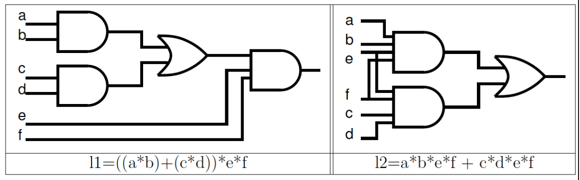
Path Delay = 2

Critical Path

- There is always a set of critical paths
  - set such that the path length of the members is at least as long as any other path length
- May be many such paths

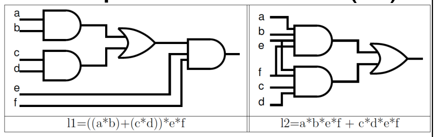


Delay also depend on Costs



- Delay:
  - I1: D(and2)+D(or2)+D(and3)
  - I2: D(and4)+D(or2)

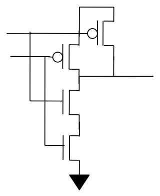
Delay also depend on Costs (4f)

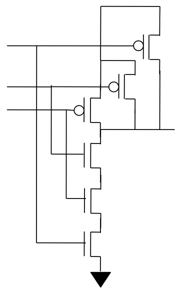


- Delay:
  - I1: D(and2)+D(or2)+D(and3)
  - I2: D(and4)+D(or2)
- D(and2)=22ps, D(and3)=27ps, D(and4)=33ps
  - ❑D(I2)=63ps
  - ❑D(I1)=79ps

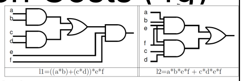
Gate Delay

Why delay increase with number inputs?



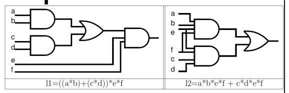


Delay also depend on Costs (4g)



- Delay:
  - I1: D(and2)+D(or2)+D(and3)
  - I2: D(and4)+D(or2)
- D(and2)=22ps, D(and3)=27ps, D(and4)=33ps
  - ❑D(I2)=63ps < D(I1)=79ps
- D(and2)=22ps, D(and3)=27ps, D(and4)=55ps
  - ❑D(I2)=79
  - ❑D(I1)=85

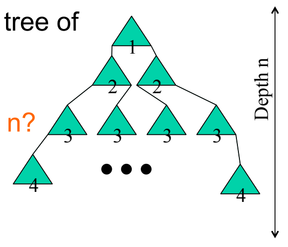
Delay and Area Optimum Differ



- I1: ((a*b) + (c*d))*e*f
- I2: ((a*b*e*f)+(c*d*e*f))
- D(and2)=22ps, D(and3)=27ps, D(and4)=33ps
  - ❑ D(I2)<D(I1)
- gate size proportional to number of inputs:
  - ❑ A(I1)<A(I2)
- **Induced Tradeoff** -- cannot always *simultaneously* minimize area and delay cost

Does delay in Gates make Sense?

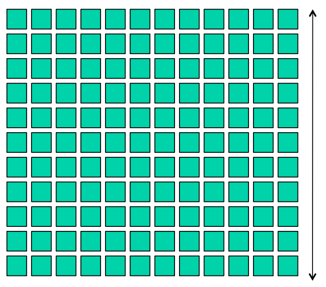
- Consider a balanced tree of logic gates of depth (tree height) n.
- Does this have delay n?
  - (unit delay gates)
- How big is it?
  (unit gate area)



Does delay in Gates make Sense?

- Consider a balanced tree of logic gates of depth (tree height) n.
- Does this have delay n?
  - (unit delay gates)
- How big is it? (unit gate area) 2^n
- How long a side?

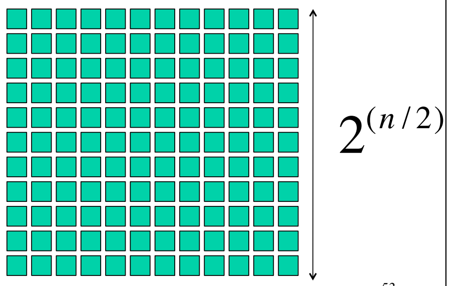
How long is a side?



Delay in Gates make Sense?

- (continuing example)
- How big is it? (unit gate area) 2^n
- How long a side? Sqrt(2^n)= $2^{(n/2)}$
- Minimum wire length from input to output?

Minimum wire length?



Delay in Gates make Sense?

- (continuing example)
- How big is it? (unit gate area) 2^n
- How long a side? Sqrt(2^n)= $2^{(n/2)}$
- Minimum wire length from input to output?
  - Ballpark ~ $2^{(n/2)}$
- Delay relate to wire length?
  (hint: think speed of light limit)
  - Delay $\propto$ wire length (if buffer properly)

Delay in Gates make Sense?

- Minimum wire length from input to output?
  - Ballpark ~ $2^{(n/2)}$
- Delay relate to wire length?
  - Delay$\propto$wire length (if buffer properly)
  - Say: Delay=WireLength*c
- TotalDelay=n*Dgate + WireLength*c
- TotalDelay=n*Dgate+$2^{(n/2)}$*c

It's not all about costs…

- …or maybe it is, just not always about a single, linear cost.
- Must manage complexity
  - Cost of developing/verifying design
  - Size of design can accomplish in fixed time
    - (limited human brainpower)
- **Today:** human brainpower is most often the bottleneck resource limiting what we can build.

Admin: Reminder

- Slides on web (morning before class)
  - Post-class may updated if feedback/class indicates something unclear
- Reading: Monday's on Blackboard
- Assignment 1 Due Monday
  - Beginning of class
- Piazza group created
- Feedback sheets

Big Ideas [MSB Ideas]

- Can implement any Boolean function in gates
  - Small set of gates are **universal**, allowing us to implement any Boolean function

Big Ideas [MSB-1 Ideas]

- Canonical representation for combinational logic
- Transformation
  - don't have to implement the input literally
  - only have to achieve same semantics
  - trivial example: logic minimization
- Minimum depends on cost model
- Often tradeoff between costs (area-delay)