ESE534: Computer Organization

Day 2: January 20, 2010

Universality, Gates, Logic

Work Preclass Exercise

Last Time

- Computational Design as an Engineering Discipline
- Importance of Costs

Today

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

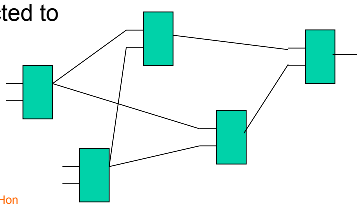
- Two functions are equivalent when
  - They have the same outputs for every input vector
  - *i.e.*, they have the same truth table
- There is a **canonical** specification for a Boolean function
  - its Truth Table

Implementation in Gates

- **Gate:** small Boolean function
- **Goal**: assemble gates to *cover* our desired 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



Implementation

- How can I implement any Boolean function with gates?

Implementation

- 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 I need to support previous implementation set?
- **Conclude:** can implement any Boolean function by a netlist of gates selected from a small set.
- **Homework (B.1):** How small can set be?

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

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

Boolean Equations

- Can be more compact:

  O=a*b*c*d*e+/a*/b*/c*/d*/e

- Often use in Register Transfer Language (RTL) expressions
  - Write logic (boolean equations) occur between registers
  - Homework 1 A.3 (deferred to HW2)

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

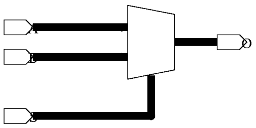
If→Mux Conversion

- Often convenient to think of IF's as Multiplexers
- If (a*b + /a*/b)

  c=d

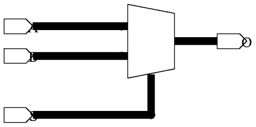
- else

  c=e



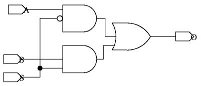
Muxes

- 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



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

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.”

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

Minimum Sum of Products

- $o=\boxed{/a*/b*c}+\boxed{/a*b*/c+a*b*/c}+\boxed{a*/b*c}$

/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

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

Term a*/c is redundant

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

Cheapest?

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

Minimize Area

- Area minimizing solutions **depends** on the technology cost structure
- Consider:
  - 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)

Minimize Area

- 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

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.

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

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

NP-hard in General

See end of
Slide set

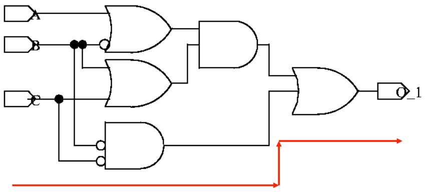
- Logic Optimization
  - Two Level Minimization
  - Covering w/ reconvergent fanout
- Are NP-hard in general
  - …but that’s **not** to say it’s not viable to find an optimal solution.
- Cover how to attack in an ESE535
  - can point you at rich literature
  - can find software to do it for you

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=22ps

Path Delay

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



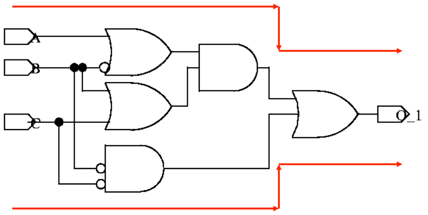
Path Delay = Delay(And3i2)+Delay(Or2)

Critical Path

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

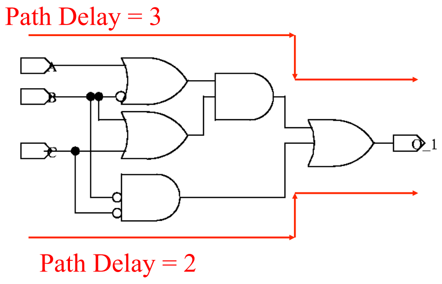
Multiple Paths

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



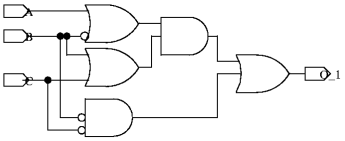
Path Delay = Delay(And3i2)+Delay(Or2)

Critical Path = Longest



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

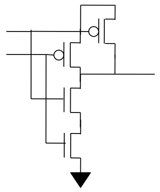
- Consider again:
  - I1: ((a*b) + (c*d))*e*f
  - I2: ((a*b*e*f)+(c*d*e*f))
- Delay:
  - I1: D(and2)+D(or2)+D(and3)
  - I2: D(and4)+D(or2)

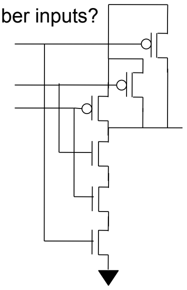
Delay also depend on Costs

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

Gate Delay

Why delay increase with number inputs?





Delay also depend on Costs

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

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)
- How long a side?
- Minimum wire length from input to output?

Delay in Gates make Sense?

- (continuing example)
- How big is it? (unit gate area) 2^n
- How long a side? $\mathrm{Sqrt}(2^n)= 2^{(n/2)}$
- Minimum wire length from input to output?
  - $2*2^{(n/2)}$
- Delay per unit length? (speed of light limit)
  - $\mathrm{Delay}\propto 2^{(n/2)}$

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 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
  - **Note deferring** A.3/C.2 to HW2
- Feedback sheets
  - Office hour preferences / constraints

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)

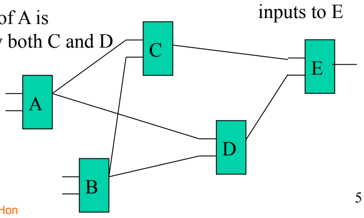
Post Lecture

- **Fanout:** output of gate drives inputs of more than one gate
- **Reconvergent Fanout:** multiple of the gates consuming one gates output contribute toward an output

Reconvergent Fanout

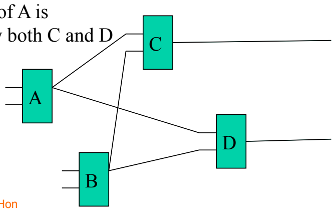
Fanout:
Output of A is
Used by both C and D

Reconvergence:
C and D both are
inputs to E



Non-Reconvergent Fanout

Fanout:
Output of A is
Used by both C and D



Post Lecture: NP-Hard

- NP-Hard
  - http://www.claymath.org/millennium/P_vs_NP/
  - Essay targeted at non-expert: http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.cs.chalmers.se/~een/PeqNP.ps