



Introduction to Digital Design

Week 5: Design Process, More Gates

Yao Zheng
University of Hawai'i at Mānoa
Department of Electrical and Computer Engineering

1

Overview

- Combinational design process
 - Translate from equation (or table) to circuit.
- More gates
 - NAND, NOR, XOR, XNOR.
- Muxes and decoders
 - Decoder: converts input binary number to one high output.
 - Multiplexor: routes one of its N data inputs to its one output, based on binary value of select inputs.
- Additional considerations
 - Circuit delay and critical path.
 - Active low Inputs.
 - Schematic capture and simulation.

2

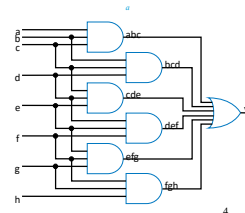
Combinational Logic Design Process

Step	Description
Step 1: Capture the function behavior	Create a truth table or equations, <i>whichever is most natural for the given problem</i> , to describe the desired behavior of each output of the combinational logic.
Step 2: Convert to circuit	2A: Create equations This substep is only necessary if you captured the function using a truth table instead of equations. Create an equation for each output by ORing all the minterms for that output. Simplify the equations if desired. 2B: Implement as a gate-based circuit For each output, create a circuit corresponding to the output's equation. (Sharing gates among multiple outputs is OK optionally.)

3

Example: Three 1s Pattern Detector

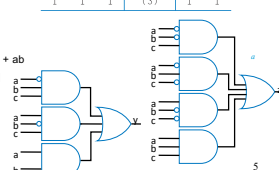
- Problem: Detect three consecutive 1s in 8-bit input: abcdefgh
 - 00011101 $\rightarrow$ 1
 - 10101011 $\rightarrow$ 0
 - 11110000 $\rightarrow$ 1
- Step 1: Capture the function**
 - Truth table or equation?
 - Truth table too big: $2^8=256$ rows
 - Equation: create terms for each possible case of three consecutive 1s
 - $y = abc + bcd + cde + def + efg + fgh$
- Step 2a: Create equation** -- already done
- Step 2b: Implement as a gate-based circuit**



4

Example: Number of 1s Counter

- Problem: Output in binary on two outputs yz the # of 1s on three inputs
 - 010 $\rightarrow$ 01
 - 101 $\rightarrow$ 10
 - 000 $\rightarrow$ 00
- Step 1: Capture the function**
 - Truth table or equation?
 - Truth table is straightforward
- Step 2a: Create equations**
 - $y = a'bc + ab'c + abc' + abc$
 - $z = a'b'c + a'bc' + ab'c' + abc$
 - Optional: Let's simplify y:
 - $y = a'bc + ab'c + ab(c' + c) = a'bc + ab'c + ab$
- Step 2b: Implement as a gate-based circuit**



5

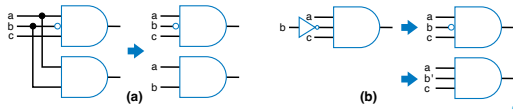
Example: Number of 1s Counter

[Video](#)

6

Simplifying Notations

- Used in previous circuit



List inputs multiple times
→ Less wiring in drawing

Draw inversion bubble
rather than inverter. Or list
input as complemented.

7

Example: Keypad Converter

- Keypad has 7 outputs

- One per row
- One per column

- Key press sets one row and one column output to 1

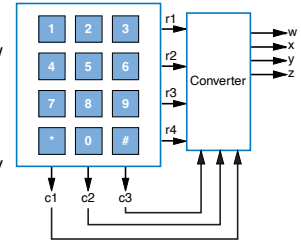
- Press "5" → $r2=1, c2=1$

- Goal: Convert keypad outputs into 4-bit binary number

- 0-9 → 0000 to 1001

- * → 1010, # → 1011

- nothing pressed: 1111



8

Example: Keypad Converter

- Step 1: Capture behavior

- Truth table too big (2^7 rows); equations not clear either

- Informal table can help

TABLE 2.7 Informal table for the 12-button keypad to 4-bit code converter.

Button	Signals	4-bit code outputs			
		w	x	y	z
1	r1 c1	0	0	0	1
2	r1 c2	0	0	1	0
3	r1 c3	0	0	1	1
4	r2 c1	0	1	0	0
5	r2 c2	0	1	0	1
6	r2 c3	0	1	1	0
7	r3 c1	0	1	1	1

$w = r3c2 + r3c3 + r4c1 + r4c3 + r1r2r3r4c1c2c3$

$x = r2c1 + r2c2 + r2c3 + r3c1 + r1r2r3r4c1c2c3$

$y = r1c2 + r1c3 + r2c3 + r3c1 + r4c1 + r4c3 + r1r2r3r4c1c2c3$

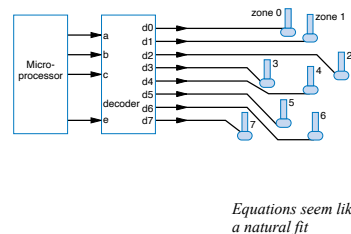
$z = r1c1 + r1c3 + r2c2 + r3c1 + r3c3 + r4c3 + r1r2r3r4c1c2c3$

Step 2b: Implement
as circuit (note
sharable gates) ...

9

Example: Sprinkler Controller

- Microprocessor outputs which zone to water (e.g., cba=110 means zone 6)
- Decoder should set appropriate valve to 1



Step 1: Capture
behavior

$$d0 = a'b'c'e$$

$$d1 = a'b'ce$$

$$d2 = a'bc'e$$

$$d3 = a'bce$$

$$d4 = ab'c'e$$

$$d5 = ab'ce$$

$$d6 = abc'e$$

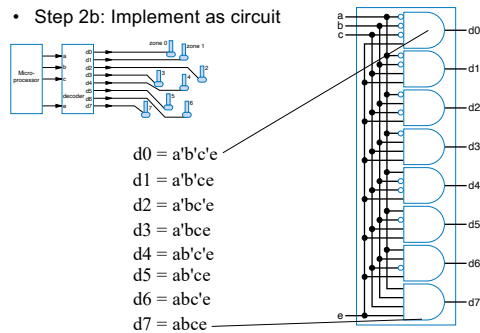
$$d7 = abce$$

Equations seem like
a natural fit

10

Example: Sprinkler Controller

- Step 2b: Implement as circuit



$$d0 = a'b'c'e$$

$$d1 = a'b'ce$$

$$d2 = a'bc'e$$

$$d3 = a'bce$$

$$d4 = ab'c'e$$

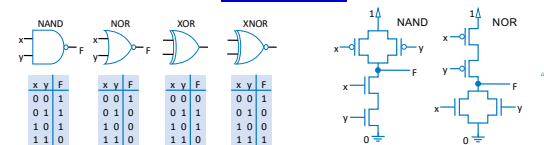
$$d5 = ab'ce$$

$$d6 = abc'e$$

$$d7 = abce$$

11

More Gates

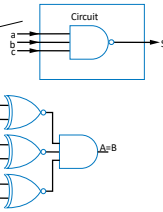


- NAND: Opposite of AND ("NOT AND")
- NOR: Opposite of OR ("NOT OR")
- XOR: Exactly 1 input is 1, for 2-input XOR. (For more inputs -- odd number of 1s)
- XNOR: Opposite of XOR ("NOT XOR")
- NAND same as AND with power & ground switched
- NOR same as OR with power/ground switched
- NAND/NOR more common
- AND in CMOS: NAND with NOT
- OR in CMOS: NOR with NOT

12

More Gates: Example Uses

- Aircraft lavatory sign example
 - $S = (abc)'$
 - Use NOR
- Detecting equality
 - Use XNOR
- Detecting odd # of 1s
 - Use XOR
 - Useful for generating "parity" bit common for detecting errors



13

13

Generate Parity Bit through XOR

- Parity bit: Extra bit added to data, intended to enable detection of error (a bit changed unintentionally)
 - e.g., errors can occur on wires due to electrical interference
- Even parity: Set parity bit so total number of 1s (data + parity) is even
 - e.g., if data is 001, parity bit is 1 → 0011 has even number of 1s
- Want equation, but easiest to start from truth table for this example

a	b	c	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Convert to eqn.

$$P = a'b'c + a'bc' + ab'c' + abc$$

14

Functional Completeness of NAND

- Any Boolean function can be implemented *using just NAND gates*. Why?
 - Need AND, OR, and NOT
 - NOT: 1-input NAND (or 2-input NAND with inputs tied together)
 - AND: NAND followed by NOT
 - OR: NAND preceded by NOTs
 - Thus, NAND is a universal gate
 - Can implement any circuit using just NAND gates
- Likewise for NOR



15

15

Number of Possible Boolean Functions

- How many possible functions of 2 variables?
 - 2^2 rows in truth table, 2 choices for each
 - $2^{(2^2)} = 2^4 = 16$ possible functions
- N variables
 - 2^N rows
 - $2^{(2^N)}$ possible functions

a	b	F
0	0	0 or 1 2 choices
0	1	0 or 1 2 choices
1	0	0 or 1 2 choices
1	1	0 or 1 2 choices

$2^4 = 16$
possible functions

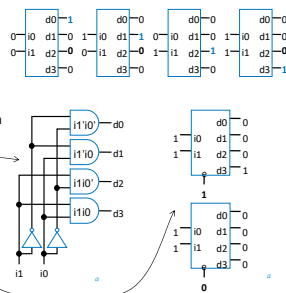
a	b	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		1															
		a AND b															
		a XOR b															
		a OR b															
		a NOR b															
		a XNOR b															
		a NAND b															

16

16

Decoders and Muxes

- Decoder**: Popular combinational logic building block, in addition to logic gates
 - Converts input binary number to one high output
- 2-input decoder: four possible input binary numbers
 - So has four outputs, one for each possible input binary number
- Internal design
 - AND gate for each output to detect input combination
- Decoder with enable e
 - Outputs all 0 if e=0
 - Regular behavior if e=1
- n-input decoder: 2^n outputs

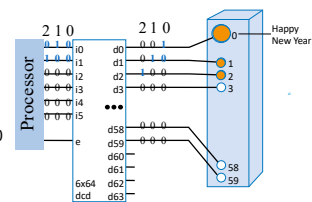


17

17

Decoder Example

- New Year's Eve Countdown Display
 - Microprocessor counts from 59 down to 0 in binary on 6-bit output
 - Want illuminate one of 60 lights for each binary number
 - Use 6x64 decoder
 - 4 outputs unused



18

18

Decoder Simulation

[Link](#)

19

19

Decoder Questions

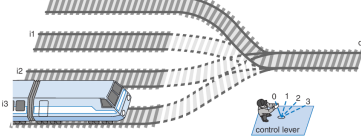
[Video](#)

20

20

Multiplexor (Mux)

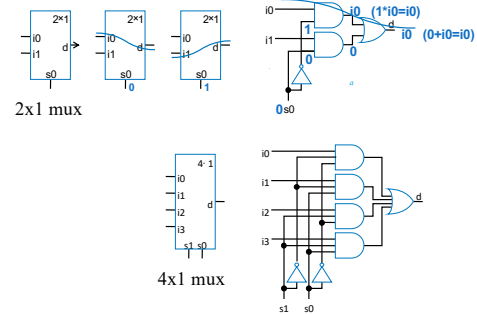
- Mux: Another popular combinational building block
 - Routes one of its N data inputs to its one output, based on binary value of select inputs
 - 4 input mux → needs 2 select inputs to indicate which input to route through
 - 8 input mux → 3 select inputs
 - N inputs → $\log_2(N)$ selects
 - Like a rail yard switch



21

21

Mux Internal Design

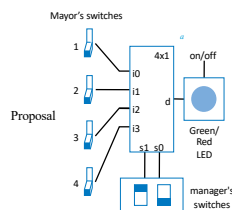


22

22

Mux Example

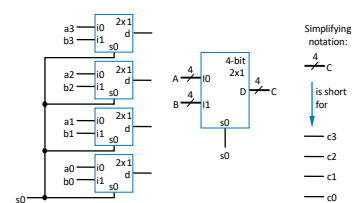
- City mayor can set four switches up or down, representing his/her vote on each of four proposals, numbered 0, 1, 2, 3
- City manager can display any such vote on large green/red LED (light) by setting two switches to represent binary 0, 1, 2, or 3
- Use 4x1 mux



23

23

Muxes Commonly Together – N-bit Mux



- Ex: Two 4-bit inputs, A (a3 a2 a1 a0), and B (b3 b2 b1 b0)
 - 4-bit 2x1 mux (just four 2x1 muxes sharing a select line) can select between A or B

24

24

N-bit Mux Example

From the car's central computer

To the above-mirror display

We'll design this later

button

- Four possible display items
 - Temperature (T), Average miles-per-gallon (A), Instantaneous mpg (I), and Miles remaining (M) – each is 8-bits wide
 - Choose which to display on D using two inputs x and y
 - Pushing button sequences to the next item
 - Use 8-bit 4x1 mux

25

Mux Questions

[Video](#)

26

Additional Considerations

Non-Ideal Gate Behavior -- Delay

(a) ideal

(b) more realistic

(c) with delay but otherwise ideal

- Real gates have some delay
 - Outputs don't change immediately after inputs change

27

Circuit Delay and Critical Path

Critical path delay = 6.5 ns
Hence, circuit's delay is 6.5 ns

- Wires also have delay
- Assume gates and wires have delays as shown
- Path delay – time for input to affect output
- Critical path – path with longest path delay
- Circuit delay – delay of critical path

28

Fan Out

- Many circuits have more than one output
- Can give each a separate circuit, or can share gates
- Ex: $F = ab + c'$, $G = ab + bc$

Option 1: Separate circuits

Option 2: Shared gates

29

Active Low Inputs

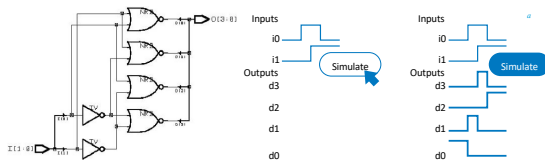
- Data inputs: flow through component (e.g., mux data input)
- Control input: influence component behavior
 - Normally active high – 1 causes input to carry out its purpose
 - Active low – Instead, 0 causes input to carry out its purpose
 - Example: 2x4 decoder with active low enable
 - 1 disables decoder, 0 enables
 - Drawn using inversion bubble

(a)

(b)

30

Schematic Capture and Simulation



- **Schematic capture**
 - Computer tool for user to capture logic circuit graphically
- **Simulator**
 - Computer tool to show what circuit outputs would be for given inputs
 - Outputs commonly displayed as **waveform**

31

31

Summary

- Combinational design process
 - Translate from equation (or table) to circuit.
- More gates
 - NAND, NOR, XOR, XNOR.
- Muxes and decoders
 - Decoder: converts input binary number to one high output.
 - Multiplexor: routes one of its N data inputs to its one output, based on binary value of select inputs.
- Additional considerations
 - Circuit delay and critical path.
 - Active low Inputs.
 - Schematic capture and simulation.

32

32