ECE 274 Digital Logic – Fall 2008

Combinational Logic Design Process and Common Combinational Components

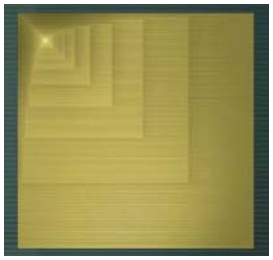
Digital Design 2.7 – 2.10

THE UNIVERSITY OF ARIZONA
TUCSON ARIZONA

Digital Design

Chapter 2:
Combinational Logic Design

Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
http://www.ddvahid.com



Digital Logic – Combinational Logic

Combinational Logic Design Process

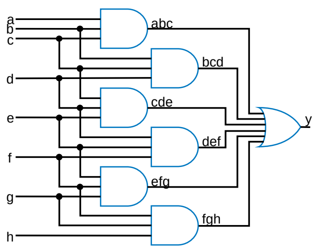
2.7

	Step	Description
Step 1	**Capture** the function	Create a truth table or equations, ***whichever is most natural for the given problem***, to describe the desired behavior of the combinational logic.
Step 2	**Convert** to equations	This step 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.
Step 3	**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.)

Digital Logic – Combinational Logic

Combinational Logic Design Process Example: Three 1s Detector

- Problem: Detect three consecutive 1s in 8-bit input: abcdefgh
  - 000**111**01 → 1   10101011 → 0
    11110000 → 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 2: Convert** to equation -- already done
  - **Step 3: Implement** as a gate-based circuit

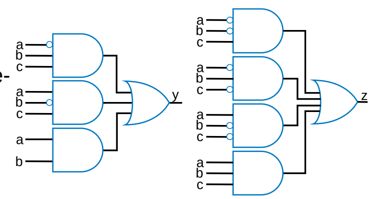


Digital Logic – Combinational Logic

Combinational Logic Design Process Example: Number of 1s Count

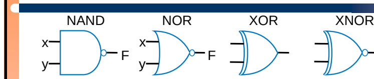
- Problem: Output in binary on two outputs yz the number of 1s on three inputs
    - 010 → 01   101 → 10   000 → 00
  - **Step 1: Capture** the function
    - Truth table or equation?
      - Truth table is straightforward
  - **Step 2: Convert** to equation
    - y = a'bc + ab'c + abc' + abc
    - z = a'b'c + a'bc' + ab'c' + abc
  - **Step 3: Implement** as a gate-based circuit

Inputs			(# of 1s)	Outputs	
a	b	c		y	z
0	0	0	(0)	0	0
0	0	1	(1)	0	1
0	1	0	(1)	0	1
0	1	1	(2)	1	0
1	0	0	(1)	0	1
1	0	1	(2)	1	0
1	1	0	(2)	1	0
1	1	1	(3)	1	1



Digital Logic – Combinational Logic

More Gates

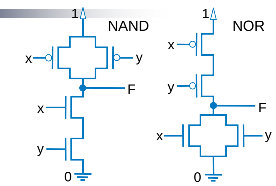


x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1



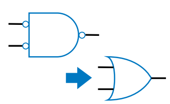
- NAND: Opposite of AND ("NOT AND")
- NOR: Opposite of OR ("NOT OR")
- XOR: Exactly 1 input is 1, for 2-input XOR. (For more inputs -- odd number of 1s)
- XNOR: Opposite of XOR ("NOT XOR")

- NAND same as AND with power & ground switched
  - Why? nMOS conducts 0s well, but not 1s (reasons beyond our scope) -- so NAND more efficient
- Likewise, NOR same as OR with power/ground switched
- AND in CMOS: NAND with NOT
- OR in CMOS: NOR with NOT
- So NAND/NOR more common

Digital Logic – Combinational Logic

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
- Likewise for NOR



Digital Logic – Combinational Logic

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

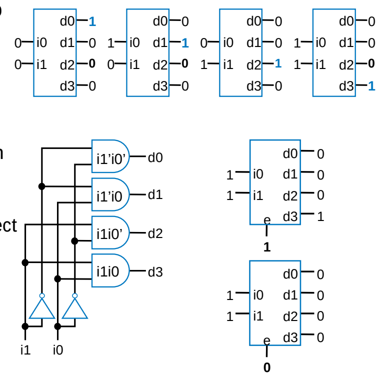
a	b	F	
0	0	0 or 1	2 choices
0	1	0 or 1	2 choices
1	0	0 or 1	2 choices
1	1	0 or 1	2 choices

$2^4 = 16$
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**
		0	a AND b		a		b	a XOR b	a OR b	a NOR b	a XNOR b	b'		a'		a NAND b	1

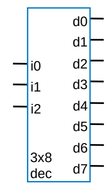
Digital Logic – Combinational Logic
Decoders and Muxes

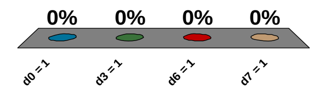
- **Decoder**: Popular combinational logic building block, in addition to logic gates
  - Converts input binary number to one high output
- 2-input decoder: four possible input binary numbers
  - So has four outputs, one for each possible input binary number
- Internal design
  - AND gate for each output 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



Digital Logic – Combinational Logic
N-bit Mux

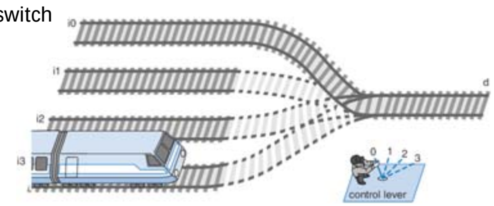
- What output of a 3x8 decoder will be asserted if i2i1i0 = 110?
  1. d0 = 1
  2. d3 = 1
  3. d6 = 1
  4. d7 = 1



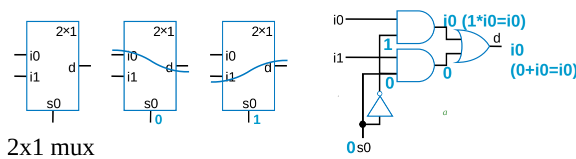


Digital Logic – Combinational Logic
Multiplexor (Mux)

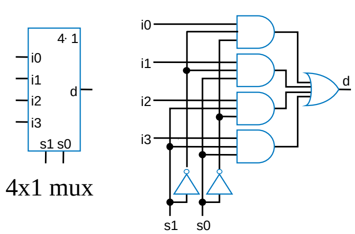
- Mux: Another popular combinational building block
  - Routes one of its N data inputs to its one output, based on binary value of select inputs
    - 4 input mux → needs 2 select inputs to indicate which input to route through
    - 8 input mux → 3 select inputs
    - N inputs → $\log_2(N)$ selects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



Digital Logic – Combinational Logic
Mux Internal Design



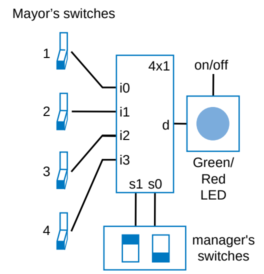
2x1 mux



4x1 mux

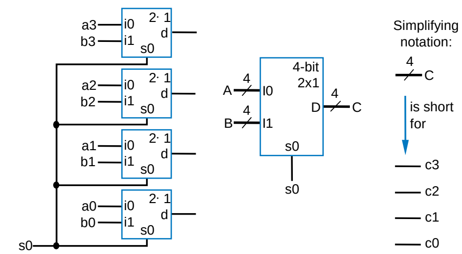
Digital Logic – Combinational Logic
Mux Example

- City mayor *(with no budget for good voting system)* 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



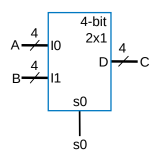
Digital Logic – Combinational Logic
N-bit Mux

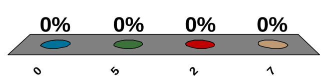
- Example: Two 4-bit inputs, A (a3 a2 a1 a0), and B (b3 b2 b1 b0)
  - 4-bit 2x1 mux (just four 2x1 muxes sharing a select line) can select between A or B



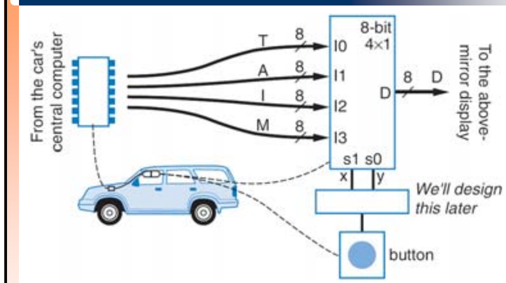
Digital Logic – Combinational Logic
N-bit Mux

- If A=5, B=2, what is the output of the 4-bit 2x1 mux if s0 = 1?
  1. 0
  2. 5
  3. 2
  4. 7





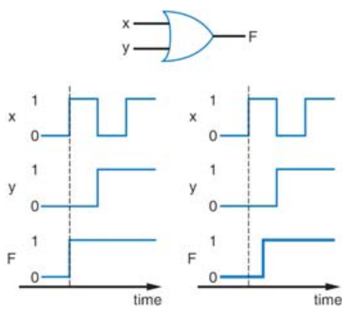
Digital Logic – Combinational Logic
N-bit Mux Example



- 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 using two inputs x and y
  - Use 8-bit 4x1 mux

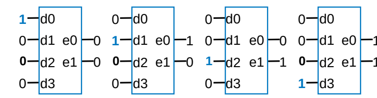
Digital Logic – Combinational Logic
Non-Ideal Gate Behavior -- Delay

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

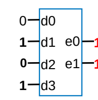


Digital Logic – Combinational Logic
Encoders

- **Encoder**: Combinational logic building block with opposite functionality of decoder
  - Outputs binary encoding for input signal that is 1
  - 4x2 encoder would have four inputs and 2 outputs



- What if two inputs are 1?
  - Can use a **priority encoder**
  - Gives priority to the highest input that is 1, and outputs binary encoding for that input
  - Example: If d3=1 and d1=1, will output e0=1 and e1=1 because d3 has priority



Digital Logic – Combinational Logic
In Class Exercise

- Design a 4x2 encoder using AND, OR, and NOT gates.