ME 311: Microprocessors and Automatic Control

Basics of digital logic design: Combinational

P.S. Gandhi
Mechanical Engineering
IIT Bombay

Life Skill

How to understand?

- I read and I forget
- I see and I remember
- I do and I understand

- So how to study? Not just by reading books, web, or slides or seeing someone do! Do it yourself ☺

Recap

- CMOS circuits, brief history of microprocessor development
- Intro to combinatorial logic design steps
  - Given general problem statement
  - Perform digital abstraction
  - Express input output in truth table
  - Write boolean expressions for each output in terms of inputs
  - Simplify using K maps or identities

Encoding/Mapping

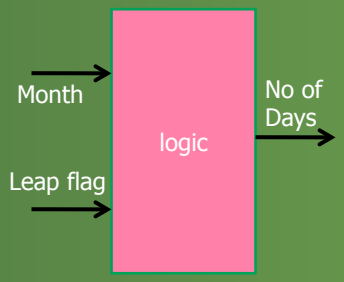
- Physical world to binary: Towards Mathematical abstraction of a given problem

Technology	State 0	State 1
Relay logic	Circuit Open	Circuit Closed
CMOS logic	0.0-1.0 volts	2.0-3.0 volts
Transistor transistor logic (TTL)	0.0-0.8 volts	2.0-5.0 volts
Fiber Optics	Light off	Light on
Dynamic RAM	Discharged capacitor	Charged capacitor
Nonvolatile memory (erasable)	Trapped electrons	No trapped electrons
Programmable ROM	Fuse blown	Fuse intact
Bubble memory	No magnetic bubble	Bubble present
Magnetic disk	No flux reversal	Flux reversal
Compact disc	No pit	Pit

Combinational logic design: Example

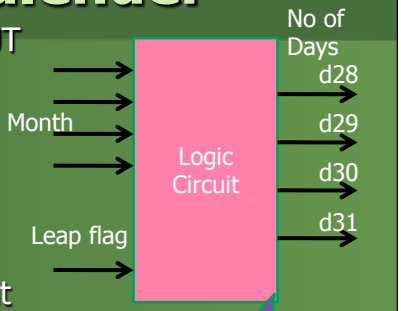
- Calender:
- Problem statement

  Given a month and a leap year flag as inputs, determine the number of days in month as output
- Application: digital watch display



Combinational logic design: Calender

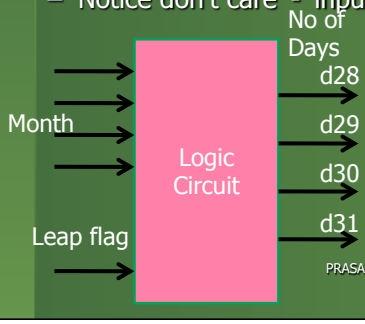
- ENCODING FOR DIGITAL CIRCIUT DESIGN/IMPLEMENTATION
  - How to encode the input of month and leap flag?
  - Total number of inputs =12
  - How many minimum inputs in terms of 0 and 1 can represent all the months? : ans 4 think why?
  - Leap year flag can be 1 if leap year and 0 if not leap year
  - Outputs are either 28, 29, 30 or 31. so 4 one-hot encoding



Alternatively we can have only two outputs saving some circuits

Combinational logic design: Calender

- TRUTH TABLE (combinational)
  - Develop truth table for considering the encoding in previous case.
  - Notice don't care '-' input



month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

Combinational logic design: Calender

- LOGIC → GATES

d28 = 1 when month is 0010 AND leap flag is 0

d28 = m8'•m4'•m2•m1'•leap'

AND NOT

d29 = 1 when month is 0010 AND leap flag is 1

d29 = m8'•m4'•m2•m1' •leap

d30 = (m8'•m4•m2'•m1') + (m8'•m4•m2•m1') + (m8•m4'•m2'•m1) + (m8•m4'•m2•m1)

OR

month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

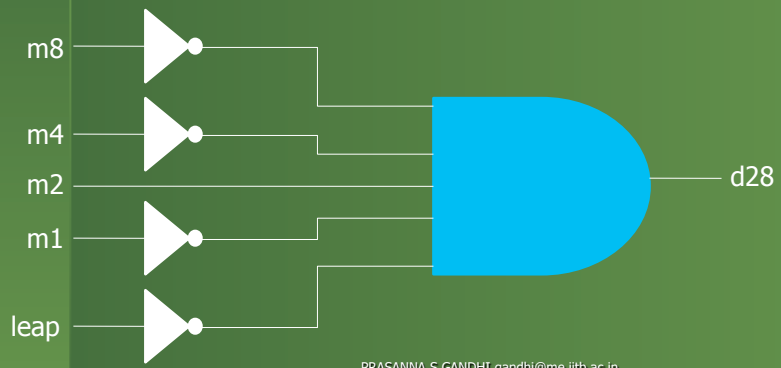
Combinational logic design: Calender

- LOGIC → GATES
- d31 = (m8'•m4'•m2'•m1)+
  (m8'•m4'•m2•m1) +
  (m8'•m4•m2'•m1) +
  (m8'•m4•m2•m1) +
  (m8•m4'•m2'•m4') +
  (m8•m4'•m2•m1') +
  (m8•m4•m2'•m1')

month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

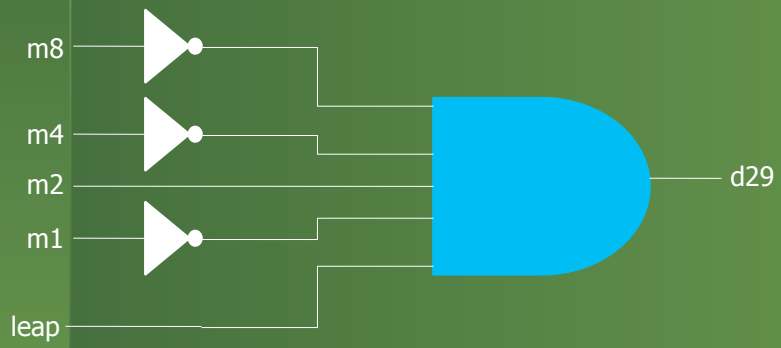
Combinational logic design: Calender

- LOGIC → GATES → logic circuit
- d28 = m8'•m4'•m2•m1'•leap'



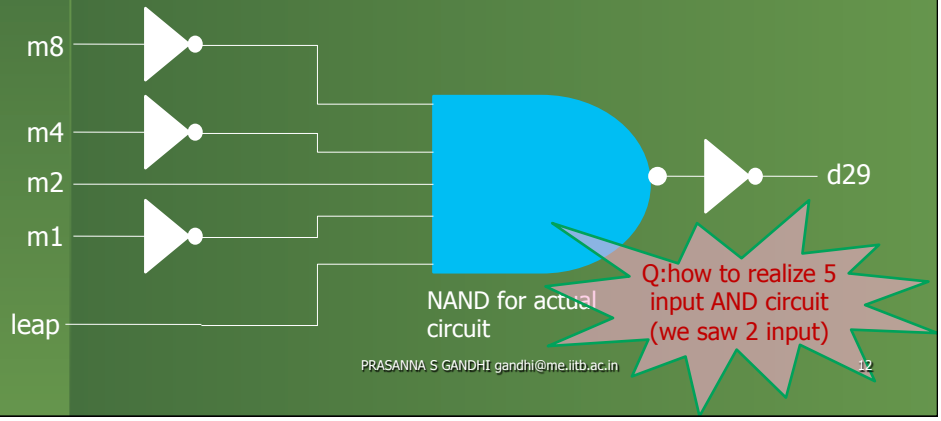
Combinational logic design: Calender

- LOGIC → GATES → logic circuit
- d29 = m8'•m4'•m2•m1'•leap



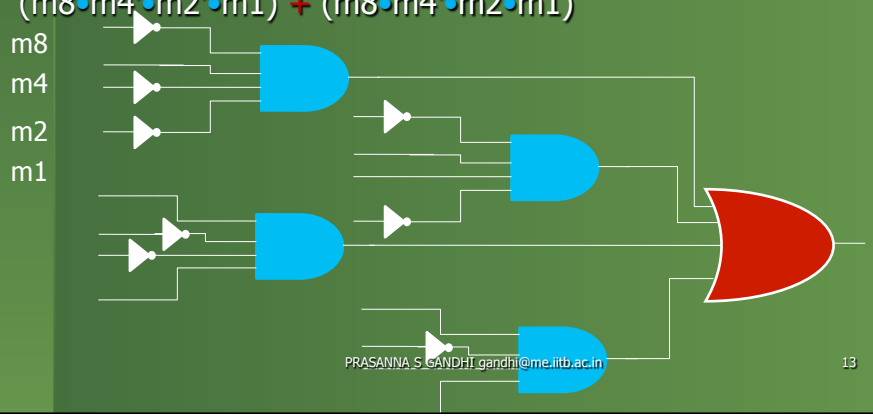
Combinational logic design: Calender

- LOGIC → GATES → logic circuit → CMOS circuit
- d29 = m8'•m4'•m2•m1'•leap



Combinational logic design: Calender

- LOGIC → GATES → logic circuit → CMOS circuit
- d30 = (m8'•m4•m2'•m1') + (m8'•m4•m2•m1') + (m8•m4'•m2'•m1) + (m8•m4'•m2•m1)



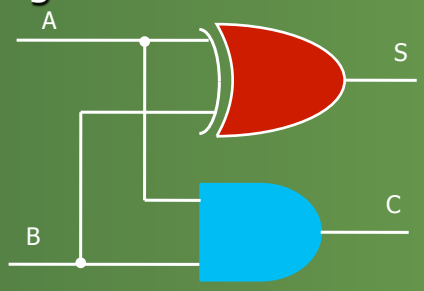
Example: Adder

Half Adder

- Truth table

A	B	Carry	Sum
0	0		
0	1		
1	0		
1	1		

Circuit: Notice XOR gate



Boolean expression

$$S = A'B + B'A = A \oplus B$$

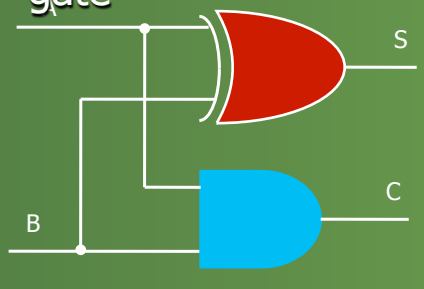
$$C = AB$$

Half Adder

- Truth table

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Circuit: Notice XOR gate



Boolean expression

$$S = A'B + B'A = A \oplus B$$

$$C = AB$$

Full Adder

- Truth table

A	B	C_{in}	C_{out}	Sum
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Boolean expression

$$Sum = AB'C'_{in} + A'BC'_{in} + A'B'C_{in} + ABC_{in}$$

$$C_{out} = A'BC_{in} + AB'C_{in} + ABC'_{in} + ABC_{in}$$

$$= BC_{in} + AC_{in} + AB$$

Full Adder

- Truth table

A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Boolean expression

$$Sum = AB'C'_{in} + A'BC'_{in} + A'B'C_{in} + ABC_{in}$$

$$C_{out} = A'BC_{in} + AB'C_{in} + ABC'_{in} + ABC_{in}$$

$$= BC_{in} + AC_{in} + AB$$

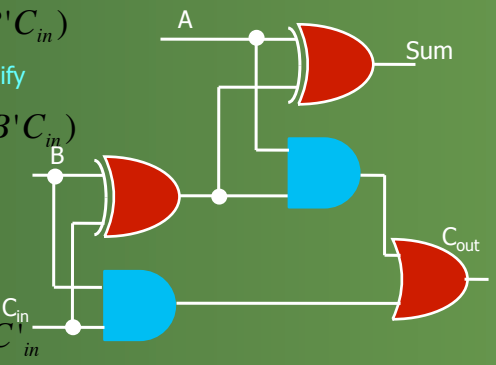
Full Adder

- Implementation 1:using XOR

$$Sum = AB'C'_{in} + A'BC'_{in} + A'B'C_{in} + ABC_{in}$$
$$= A(B'C'_{in} + BC_{in}) + A'(BC'_{in} + B'C_{in})$$

DeMorgan's law+ simplify

$$= A(B'C_{in} + BC'_{in})' + A'(BC'_{in} + B'C_{in})$$
$$Sum = A \oplus (B \oplus C_{in})$$
$$C_{out} = A'BC_{in} + AB'C_{in}$$
$$+ ABC'_{in} + ABC_{in}$$
$$= BC_{in}(A + A') + AB'C_{in} + ABC'_{in}$$
$$= BC_{in} + A(B'C_{in} + BC'_{in})$$
$$C_{out} = BC_{in} + A(B \oplus C_{in})$$



How many 'levels' this implementation has?

Full Adder

- Implementation 2:No XOR

$$Sum = AB'C'_{in} + A'BC'_{in} + A'B'C_{in} + ABC_{in}$$
$$= A(B'C'_{in} + BC_{in}) + A'(BC'_{in} + B'C_{in})$$

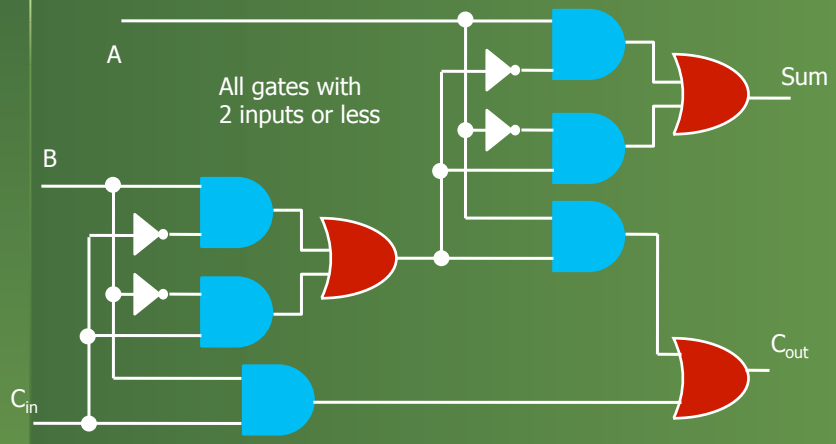
DeMorgan's law

$$= A(B'C_{in} + BC'_{in})' + A'(BC'_{in} + B'C_{in})$$

$$C_{out} = A'BC_{in} + AB'C_{in}$$
$$+ ABC'_{in} + ABC_{in}$$
$$= BC_{in}(A + A') + AB'C_{in} + ABC'_{in}$$
$$= BC_{in} + A(B'C_{in} + BC'_{in})$$

Full Adder

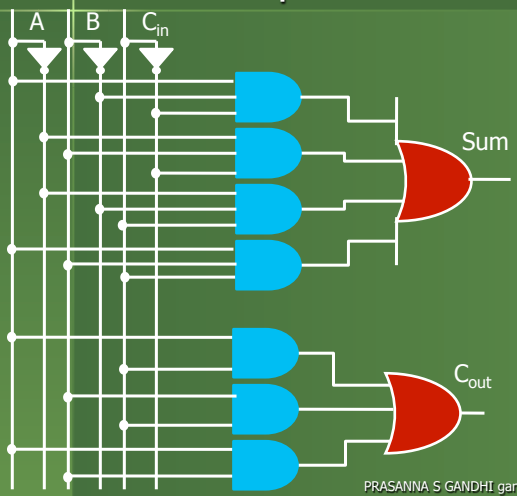
- Implementation 2:No XOR



How many 'levels' this implementation has?

Full Adder

- Implementation 3:Two level



- Q: Which of the implementation should be preferred?
- The one that has less levels of implementation or less no of inputs to gates used!!!!

Identities of Boolean algebra

- Revise standard identities used for simplification
- K maps: If you want to optimize the digital circuit

Full Adder

- Implementation: Fundes

- The larger the number of inputs to a gate slower it will be
- Also the propagation delay increases more than linearly with number of inputs to a gate
- Good rule-of-thumb: DO NOT use gates with more than four inputs

THANK YOU

Combinational logic design: Calender

- What is logic in c program you can come up with?

```
integer  number_of_days ( month,
leap_year_flag) {
switch (month) {
case 1: return (31);
case 2: if (leap_year_flag == 1)
          then return (29)
          else return (28);
case 3: return (31);
...
case 12: return (31);
default: return (0);
}
}
```