CSSE132
Introduction to Computer Systems

9 : Combinational logic

March 18, 2013

Today: Combinational Logic

- Voting logic and circuit
- Karnaugh maps

- Adding
  - 1-bit adder
  - 32-bit adder
  - Issues with carry

Karnaugh Maps

- Representation of truth table
  - Usually 2D
  - Axes are input values

- 2 input AND gate example

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

		B	
		0	1
A	0	0	0
	1	0	1

Karnaugh Maps

- Can be extended to 3 and 4 inputs
  - Used to quickly simplify logical expressions
  - Adjacent cells differ by one inversion in product
  - Nearby cells can be grouped into logical expression

		C	
		0	1
AB	00	0	0
	01	0	1
	11	1	1
	10	0	1

		CD			
		00	01	11	10
AB	00	1	0	1	1
	01	0	0	1	1
	11	0	1	1	0
	10	0	1	1	0

K-Map Example

- Logical expression for this truth table
- SOP:
  A'B'C'D + A'BC'D' + A'BC'D + A'BCD'+
  A'BCD + ABCD'
- Simplify with Theorems

A	B	C	D	R
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

K-Map Example

- Build K-map from truth table
  - 2D grid of AB, CD
  - Only change one bit per row/col

A	B	C	D	R
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

K-Map Example

- Build K-map from truth table
  - 2D grid of AB, CD
  - Only change one bit per row/col

		CD			
		00	01	11	10
AB	00	0	1	0	0
	01	1	1	1	1
	11	0	0	0	1
	10	0	0	0	0

A	B	C	D	R
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

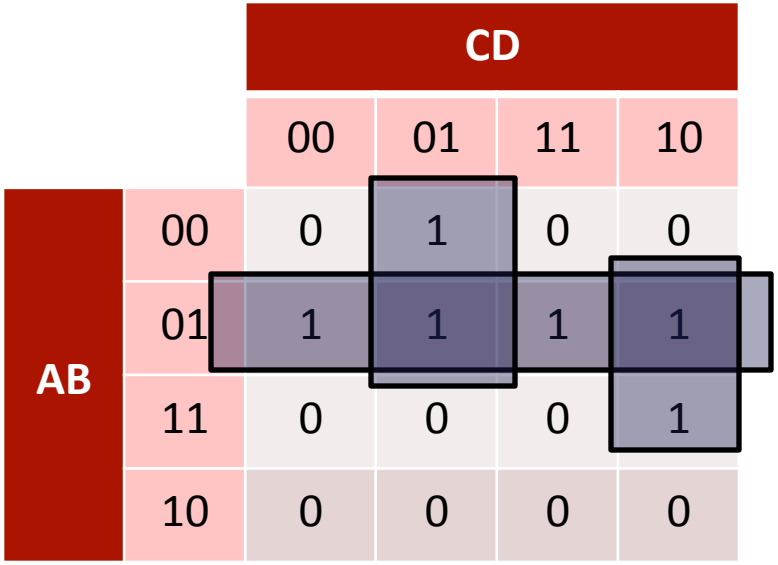
K-Map Example

- Group nearby ‘1’s
  - Rectangle shape
  - No 0s inside groups
  - Number of groups cells must be power of two
  - Edges are adjacent
  - Each group must have at least 1 unique cell
  - Each group must be as large as possible
  - Every 1 must be in a group
- Goal: fewest groups possible

		CD			
		00	01	11	10
AB	00	0	1	0	0
	01	1	1	1	1
	11	0	0	0	1
	10	0	0	0	0

K-Map Example

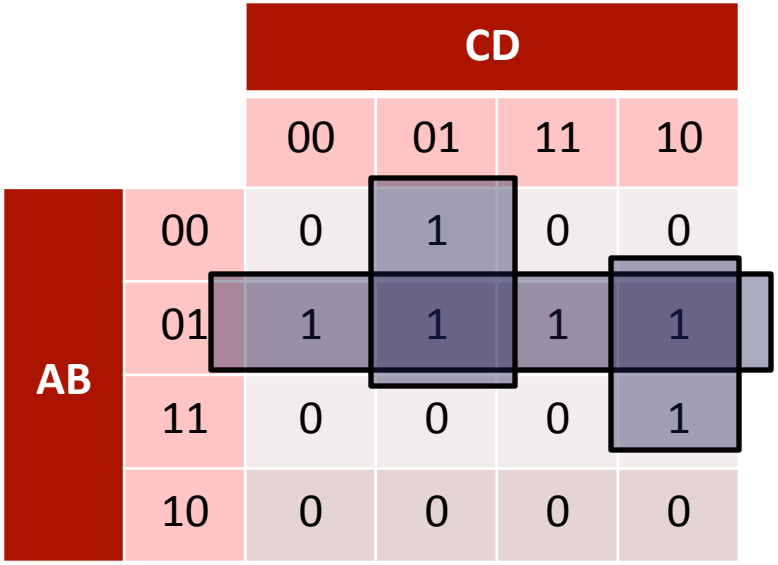
- Group nearby ‘1’s
  - Rectangle shape
  - No 0s inside groups
  - Number of groups cells must be power of two
  - Edges are adjacent
  - Each group must have at least 1 unique cell
  - Each group must be as large as possible
  - Every 1 must be in a group
- Goal: fewest groups possible



K-Map Example

- Form expression from groups
  - SOP expression
- Biggest group:
  - Contains all of CD
  - When A'B
  - = A'B
- Top-left
  - Contains part of C'D
  - Doesn't contain any A
  - = A'C'D
- Rightmost
  - Contains part of CD'
  - Does not contain any B'
  - =BCD'

- SOP:
  A'B'C'D + A'BC'D' + A'BC'D + A'BCD'+ A'BCD + ABCD'



Majority voting decision

- Returns true when the majority of inputs are true
- What is the K-map?
- What are the groups?
- What is the simplified expression?
- What is the circuit?

A	B	C	R
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Adding

- Binary adder
  - Inputs A and B
  - Outputs sum and carry
- Cannot accept carry
  - Called ‘half-adder’
- Expression
  - S = A’B + AB’ = A^B
  - C = AB

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full-Adder

- Binary adder
  - Inputs A, B, carry
  - Outputs sum and carry
- Expression
  - S = ?
  - C = ?

A	B	C	S	C
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Full-Adder

- Binary adder
  - Inputs A, B, carry
  - Outputs sum and carry
- Expression
  - S = ?
  - C = ?

A	B	C	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

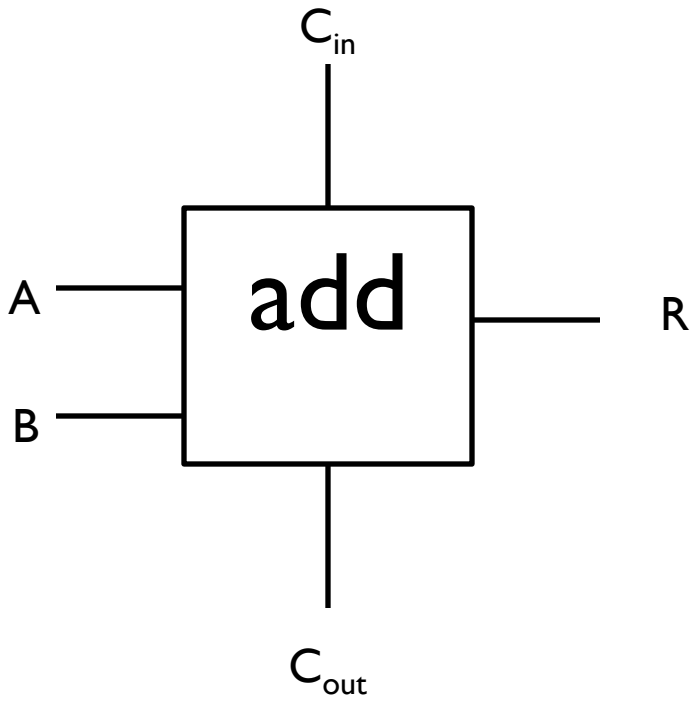
Full-Adder

- Binary adder
  - Inputs A, B, C_{in}
  - Outputs sum and C_{out}
- Expression
  - $S = AB'C_{in}' + A'BC_{in}' + A'B'C_{in} + ABC_{in}$
  - $C_{out} = AB + BC_{in} + AC_{in}$

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

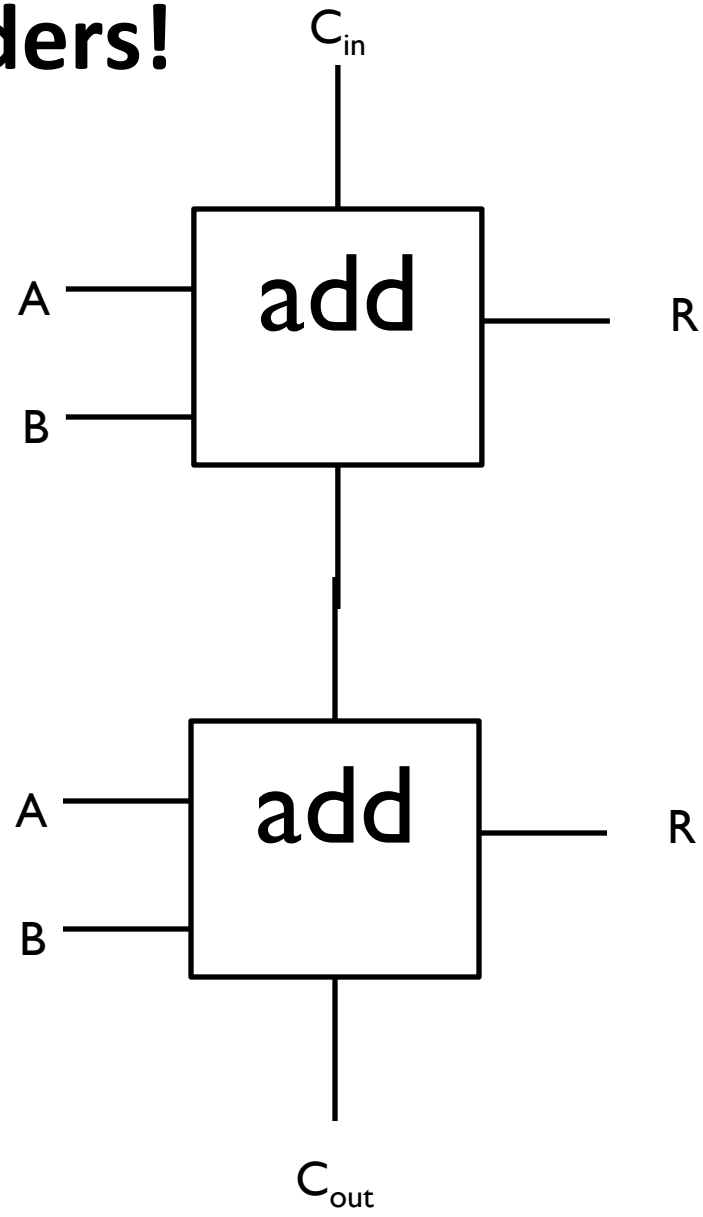
32 bit adder

- Abstract 1bit adder to single unit
- Use 32 1bit adders to create 32 bit adder



2 bit adder : 2 x 1 bit adders!

- Use 2 x 1 bit adders
- Wire C_{out} into next C_{in}
  - Relies on output from previous adder
- Can duplicate for any number of bits



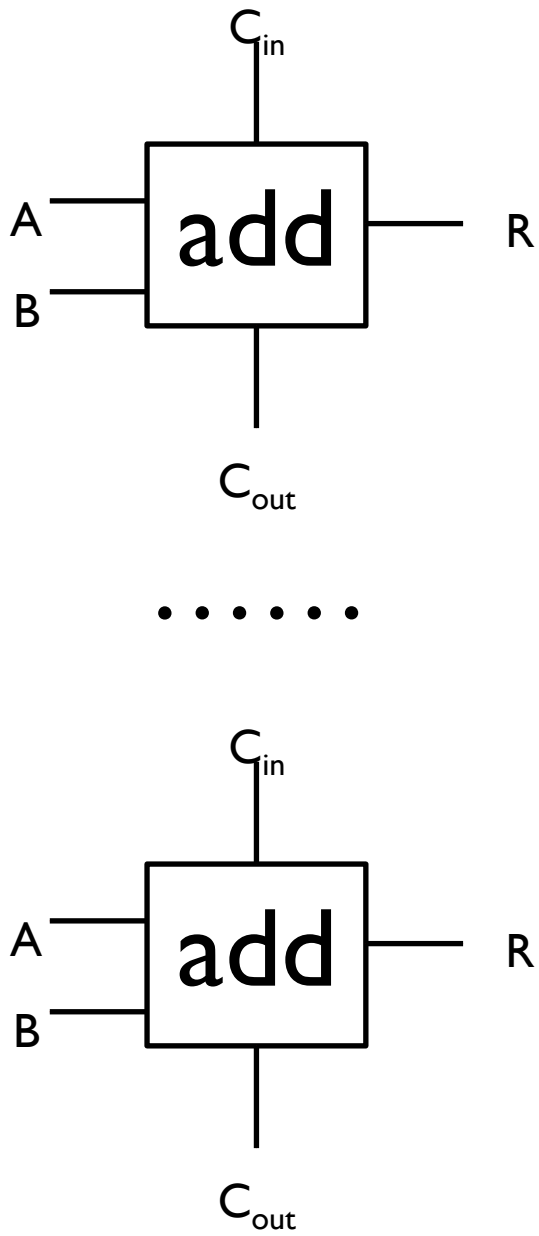
Carry issues

- Consider a 32 bit adder
- Final adder needs input from previous adder (carry result)
- Must wait for 31 other adders!
  - Very slow

C1 = (B0*C0)+(A0*C0)+(A0*B0)

C2 = (B1*C1)+(A1*C1)+(A1*B1)

…



Carry Expansion

- Can represent any function in two levels of logic
  - Build full truth table
  - Construct POS/SOP
  - Implement with gates
- Let's do this for carry

  c1 = (b0*c0)+(a0*c0)+(a0*b0)

  c2 = (b1*c1)+(a1*c1)+(a1*b1)

  So…

  d2 = (b1*(b0*c0)+(a0*c0)+(a0*b0))

  + (a1 * ((b0*c0)+(a0*c0)+(a0*b0))

  + (a1*b1)

  c3 = … , c4 = …
- Logic becomes very large
  - Gate/space cost too much

Carry lookahead

- Refactor carry equation
  - Input that generates carry
  - Input that propagates carry
- In general

$$ci + 1 = (bi * ci) + (ai * ci) + (ai * bi)$$
$$= (ai * bi) + (ai + bi) * ci$$

- So, second carry:

$$c2 = (a1 * b1) + (a1 + b1) * ((a0 * b0) + (a0 + b0) * c0)$$

- In these examples
  - Generate carry: $gi = (ai * bi)$
  - Propagate carry: $pi = (ai + bi) * ci$

Carry lookahead

- 4-bit carry lookahead

  $c1 = g0 + (p0 \cdot c0)$

  $c2 = g1 + (p1 \cdot g0) + (p1 \cdot p0 \cdot c0)$

  $c3 = g2 + (p2 \cdot g1) + (p2 \cdot p1 \cdot g0) + (p2 \cdot p1 \cdot p0 \cdot c0)$

  $c4 = g3 + (p3 \cdot g2) + (p3 \cdot p2 \cdot g1) + (p3 \cdot p2 \cdot p1 \cdot g0) + (p3 \cdot p2 \cdot p1 \cdot p0 \cdot c0)$

- Build unit that computes carries
  - Inputs: a0, a1, a2, a3, b0, b1, b2, b3, carryIn0
  - Output: c1, c2, c3, c4
  - Couple with 4-bit adder
- Each 4-bit adder
  - Gets lookahead unit for fast carries
- To expand beyond 4 bits
  - Build carry-lookahead unit for the 4 bit carry-lookahead unit!