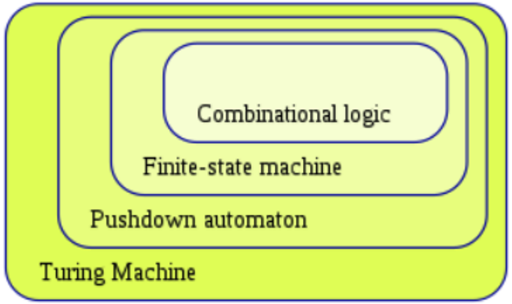
Lecture

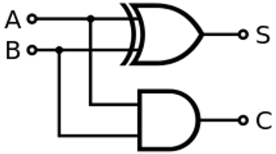


- Last time:
  - Boolean Algebra ⇔ Digital Circuits
    - Point: We can do a lot with just Combinational logic -- all true functions can be evaluated
    - Point: Digital Circuits can be built to evaluate all of these functions.
  - All we need is And (*), Or (+) and Not (')
  - Truth Table → Boolean Algebra → Truth Tables
  - Boolean Algebra → Circuits → Boolean Algebra
  - Minimization of Circuits
    - Algebraic Transformations
    - Karnaugh Maps

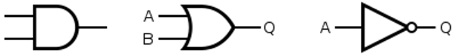
- Today: More Combinational Circuits

Combinational Logic

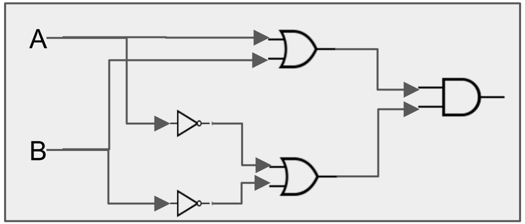
- Using just is tedious: AND (*), OR (+), NOT (')
- Solution: Build components and reuse!
  - XOR:  A ⊕ B is equivalent to (A + B) * (A' + B')
  - Half-Adder: S = A ⊕ B, C = A * B



- Bigger components and with more bits!
  - Binary Addition
  - Binary Subtraction
  - BCD Addition
  - Decoder
  - Multiplexer



XOR:

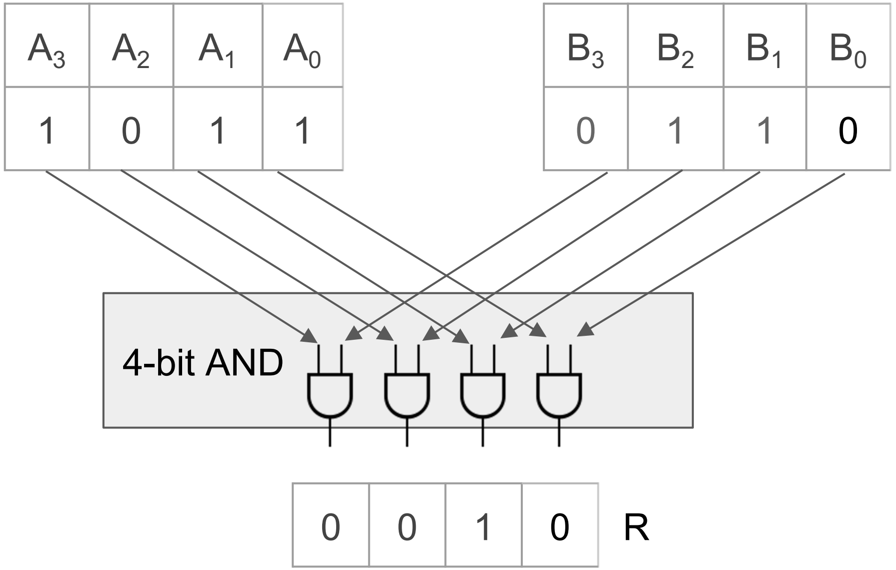


A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

4-bit Bitwise AND

- R = A & B

	1	0	1	1	A
&	0	1	1	0	B
	0	0	1	0	R

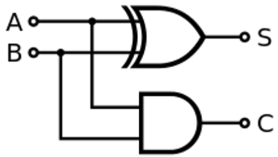


- For a n-bit operation,
  - create n-duplicates of the base circuitry
  - layout duplicates in parallel
  - package it up

1-bit Binary Addition

- Recall:
  - A + B → S, C

				1	A
+				1	B
					S



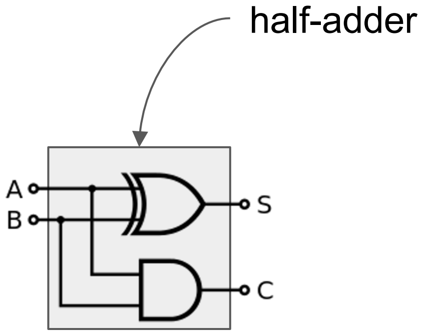
A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

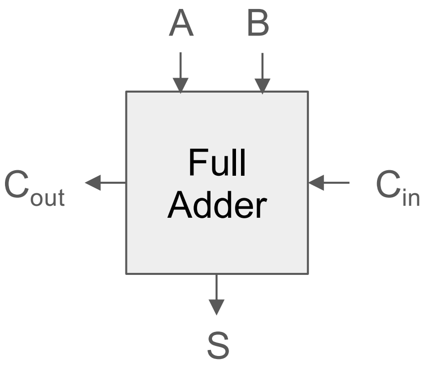
4-bit Binary Addition

- Recall:
  - C_{in} + A_x + B_x → C_{out}, S_x

	1	0	1	1	A
+	0	1	1	0	B
					S

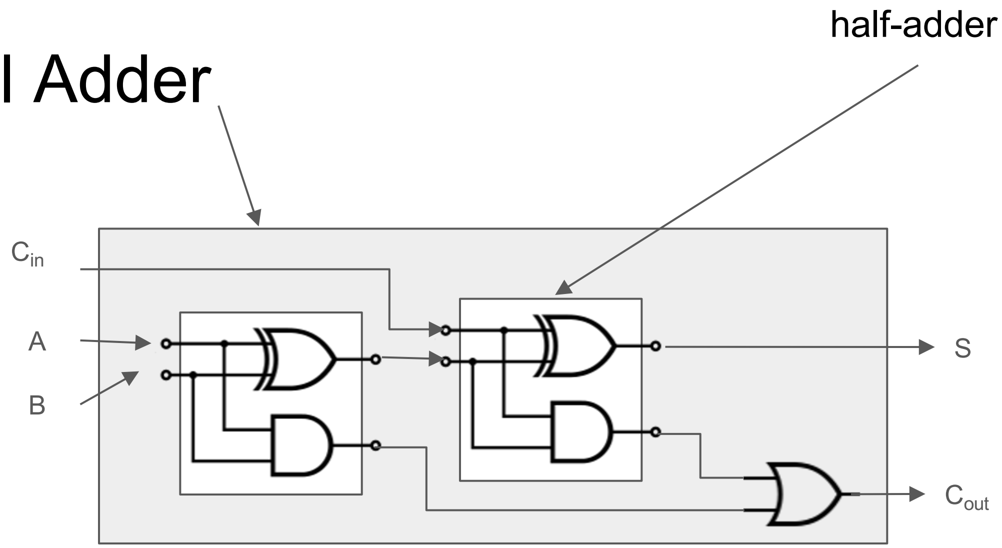
- Half-Adder is not sufficient!
  We need a Full-Adder





C_{in}	A	B	C_{out}	S
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

Full Adder

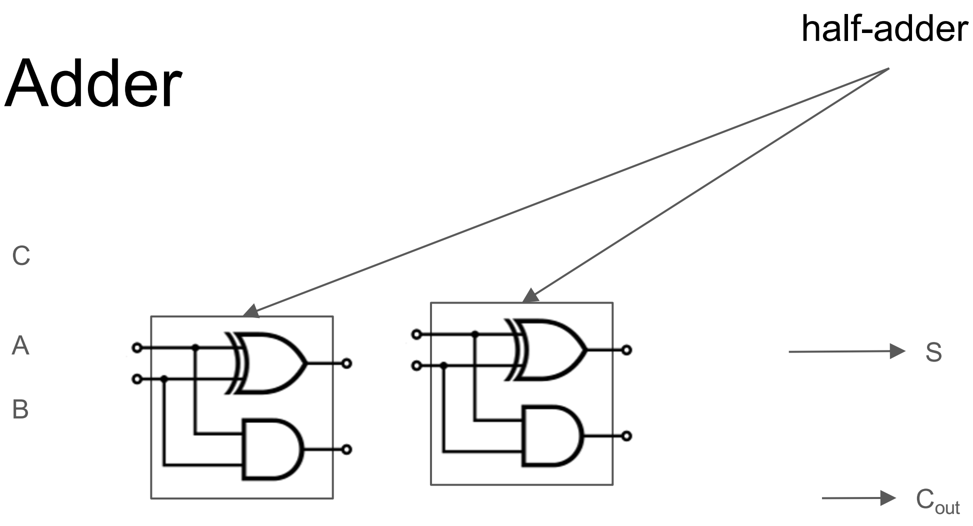


Got It? Any Questions

C_{in}	A	B	C_{out}	S
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

- $C_{out} = AB + C_{in}(A \oplus B)$
- $S = C_{in} \oplus A \oplus B$

Full Adder



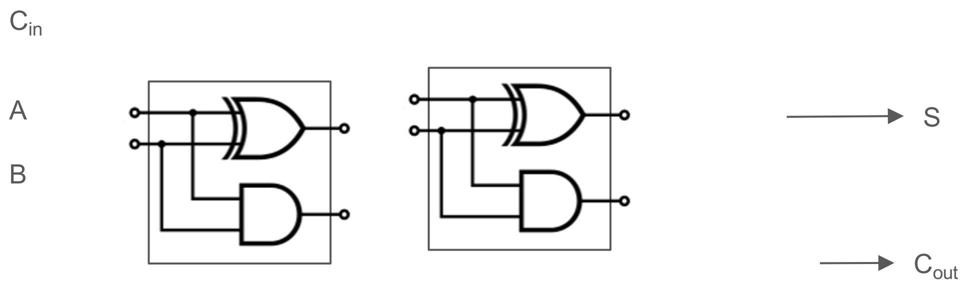
Note: Renamed C_{in} to be C

C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	

Use Sum of Products

- C_{out} = C'AB +

Full Adder

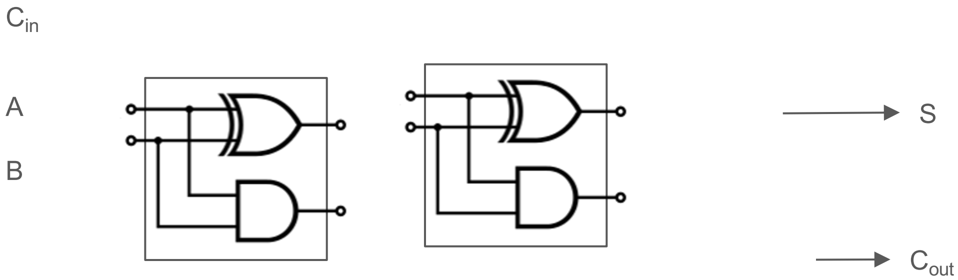


C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	CA'B
1	1	0	1	0	CAB'
1	1	1	1	1	CAB

- C_{out} = C'AB + CA'B + CAB' + CAB

Use Sum of Products

Full Adder

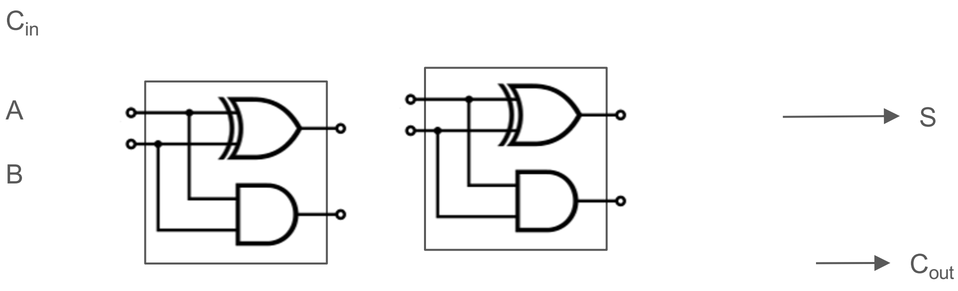


C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	CA'B
1	1	0	1	0	CAB'
1	1	1	1	1	CAB

Use Commutative Property

- C_{out} = C'AB + $\boxed{\text{CA'B + CAB'}}$ + $\boxed{\text{CAB}}$
- ➢ C_{out} = C'AB + CAB + CA'B + CAB'

Full Adder

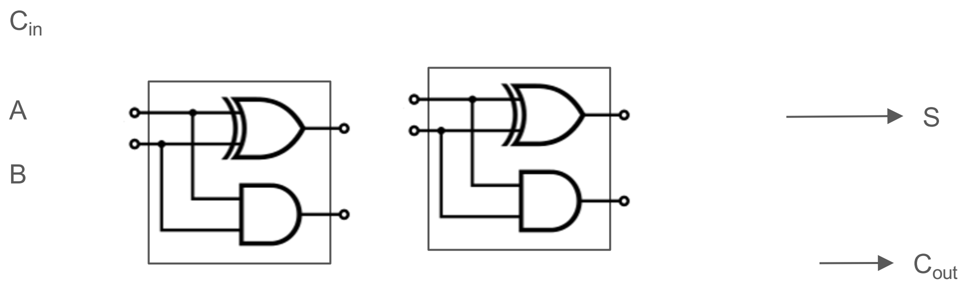


C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	CA'B
1	1	0	1	0	CAB'
1	1	1	1	1	CAB

Use Distributive Property

- $C_{out} = \boxed{C'AB} + \boxed{CAB} + CA'B + CAB'$
  - ➢ $C_{out} = (C' + C)AB + CA'B + CAB'$

Full Adder

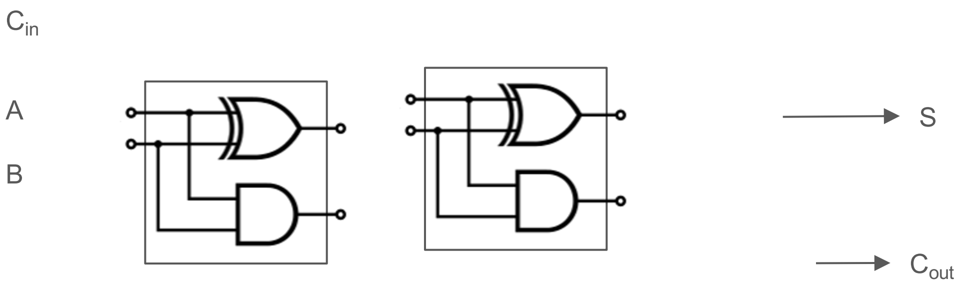


C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	CA'B
1	1	0	1	0	CAB'
1	1	1	1	1	CAB

Use Complement Property

- $C_{out} = (\boxed{C' + C})AB + CA'B + CAB'$
  - ➢ $C_{out} = (\text{true})AB + CA'B + CAB'$

Full Adder

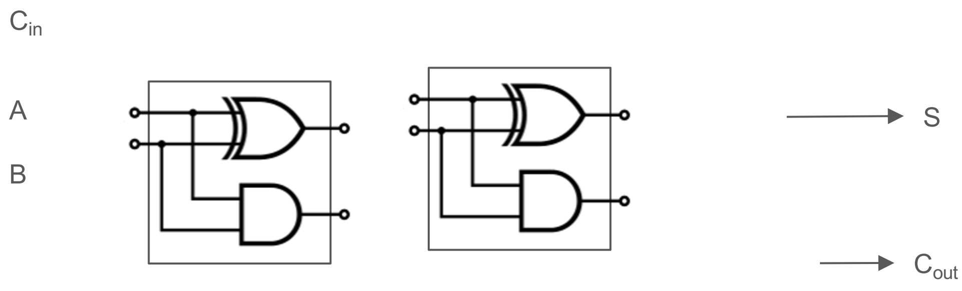


C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	CA'B
1	1	0	1	0	CAB'
1	1	1	1	1	CAB

- C_{out} = $\boxed{\text{(true)AB}}$ + CA'B + CAB'
  - ➢ C_{out} = AB + CA'B + CAB'

Use Identity Property

Full Adder

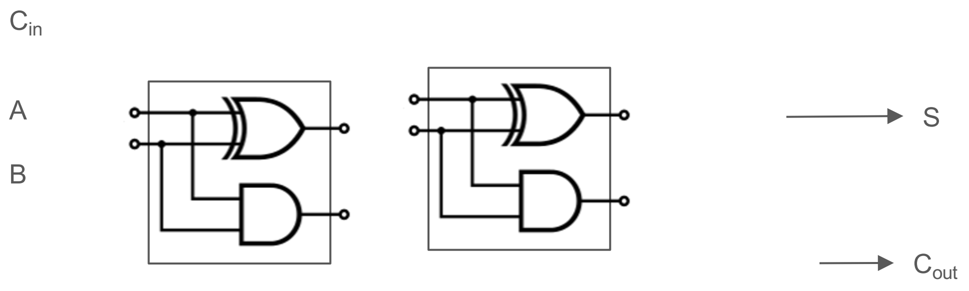


C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	CA'B
1	1	0	1	0	CAB'
1	1	1	1	1	CAB

- $C_{out} = AB + \boxed{CA'B + CAB'}$
   - $C_{out} = AB + C(A'B + AB')$

Use Distributive Property

Full Adder



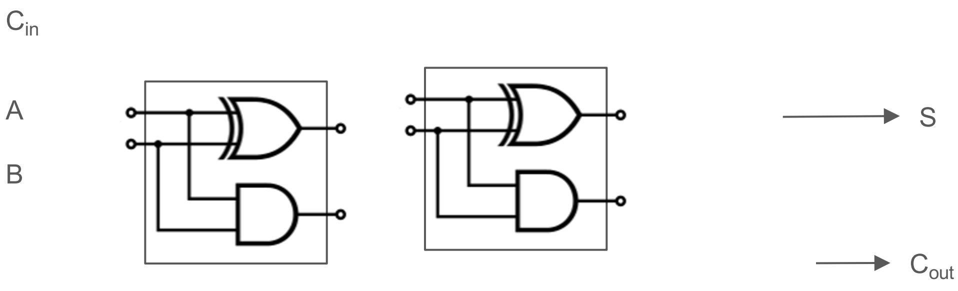
C_{in}	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	C'AB
1	0	0	0	1	
1	0	1	1	0	CA'B
1	1	0	1	0	CAB'
1	1	1	1	1	CAB

- $C_{out} = AB + \boxed{C(A'B + AB')}$

✓ $C_{out} = AB + C(A \oplus B)$

$A \oplus B \Leftrightarrow A'B + AB'$

Full Adder



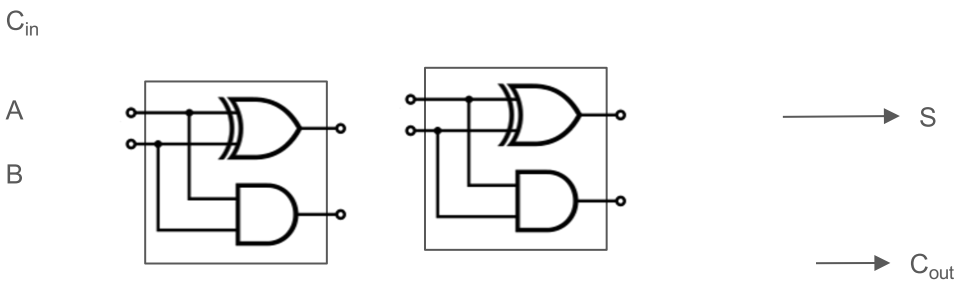
C	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	C'A'B
0	1	0	0	1	C'AB'
0	1	1	1	0	
1	0	0	0	1	CA'B'
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	CAB

Sum of Products:	C'A'B + C'AB' + CA'B' + CAB

- ✓ C_{out} = AB + C(A⊕B)
- S = C'A'B + C'AB' + CA'B' + CAB

A ⊕ B ⇔ A'B + AB'

Full Adder



C	A	B	C_{out}	S	
0	0	0	0	0	
0	0	1	0	1	C'A'B
0	1	0	0	1	C'AB'
0	1	1	1	0	
1	0	0	0	1	CA'B'
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	CAB

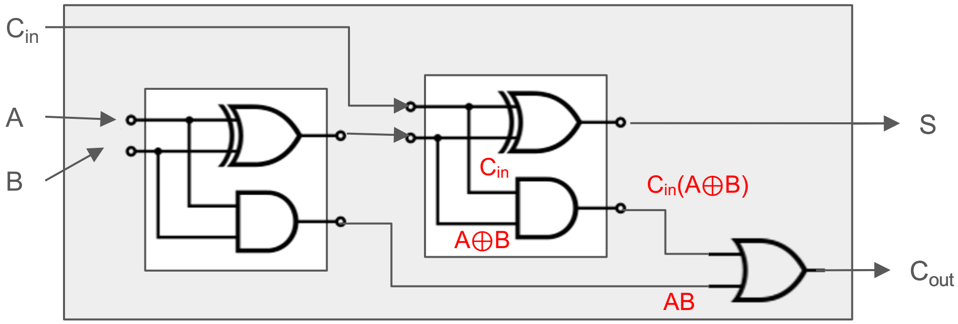
Sum of Products:

$$
\begin{aligned}
& C'A'B + C'AB' + CA'B' + CAB \\
&= C'(A'B + AB') + C(A'B' + AB) \\
&= C'(A \oplus B) + C(A \oplus B)' \\
&= C \oplus (A \oplus B) \\
&= C \oplus A \oplus B
\end{aligned}
$$

- ✓ $C_{out} = AB + C(A \oplus B)$
- $S = C'A'B + C'AB' + CA'B' + CAB$
- ✓ $S = C \oplus A \oplus B$

$A \oplus B \Leftrightarrow A'B + AB'$

Full Adder



C	A	B	C_{out}	S
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

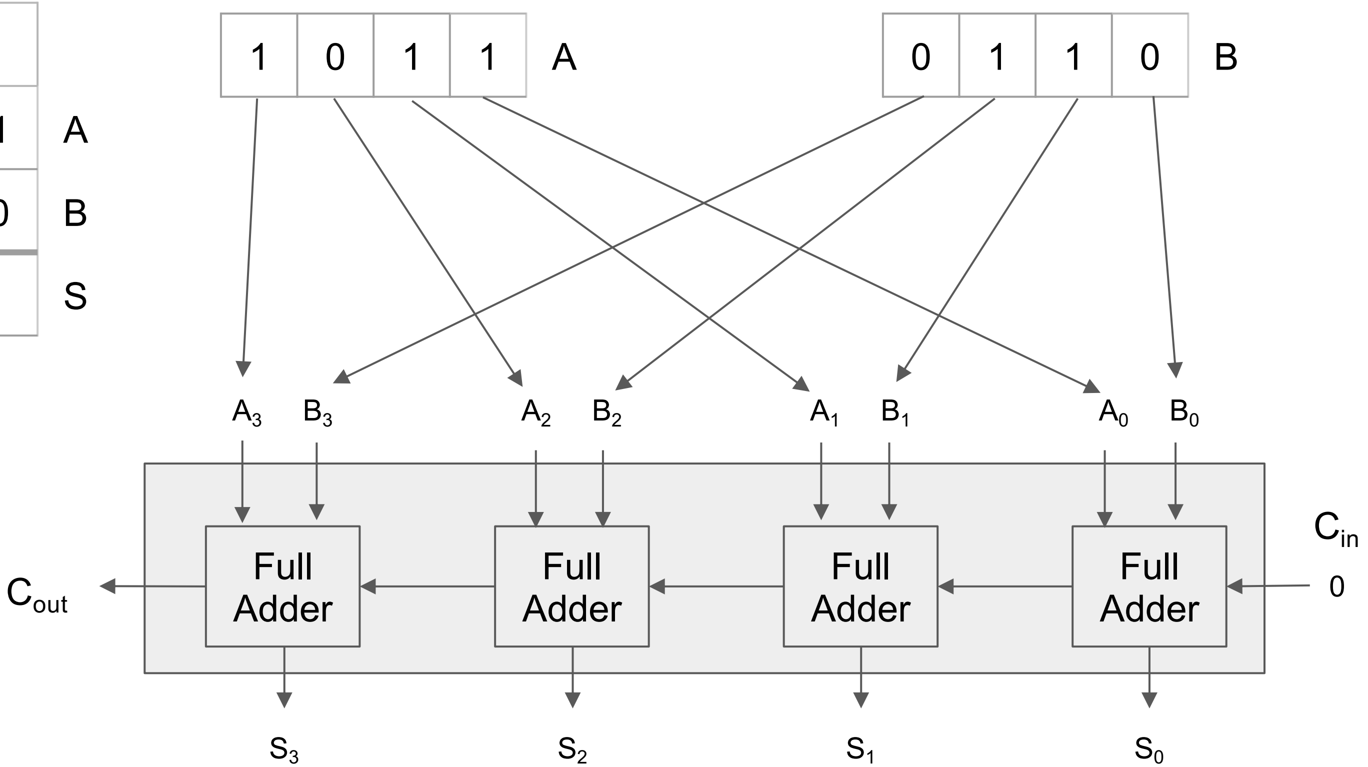
Note: Renamed C to be C_{in}

✓ $C_{out} = AB + C_{in}(A\oplus B)$

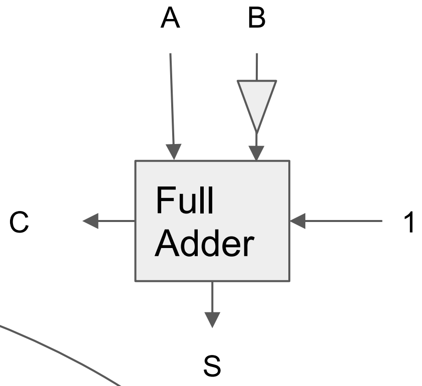
✓ $S = C_{in}\oplus A\oplus B$

4-bit Binary Addition (aka: 4-bit Full Adder)

	1	0	1	1	A
	+ 0	1	1	0	B
					S



Binary Subtractor

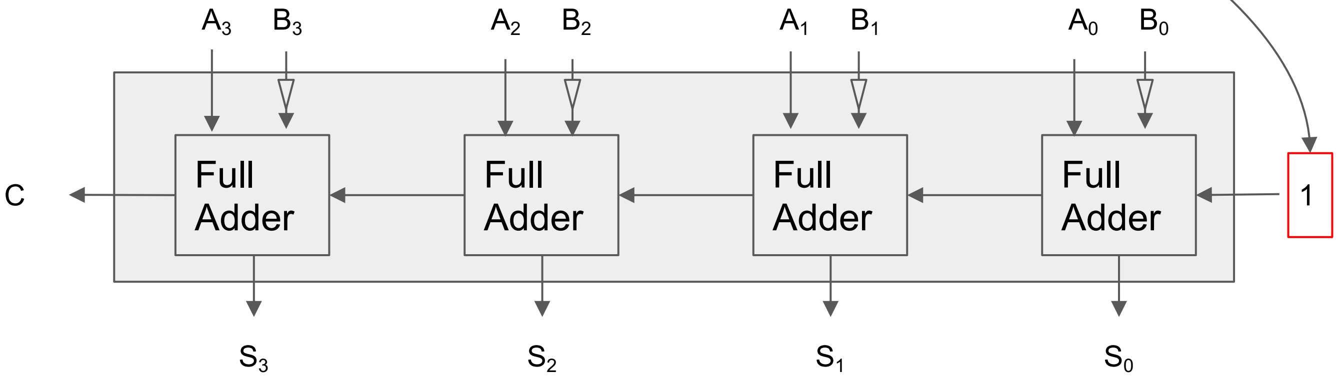


- Recall: A - B

  = A + (2's complement of B)

  = A + (1's complement of B) + 1

- 4-bit Binary Subtractor



4-bit BCD Addition

- BCD: 35 + 28

$$\begin{array}{r} 35 \\ +\ 28 \end{array}$$

	0	1	1	1	
	0	0	1	1	3
+	0	0	1	0	2

	0	0	0	0	
	0	1	0	1	5
+	1	0	0	0	8

#	Encoding $S_3S_2S_1S_0$		Encoding $S_3S_2S_1S_0$
0	0 0 0 0	8	1 0 0 0
1	0 0 0 1	9	1 0 0 1
2	0 0 1 0		1 0 1 0
3	0 0 1 1		1 0 1 1
4	0 1 0 0	invalid	1 1 0 0
5	0 1 0 1		1 1 0 1
6	0 1 1 0		1 1 1 0
7	0 1 1 1		1 1 1 1

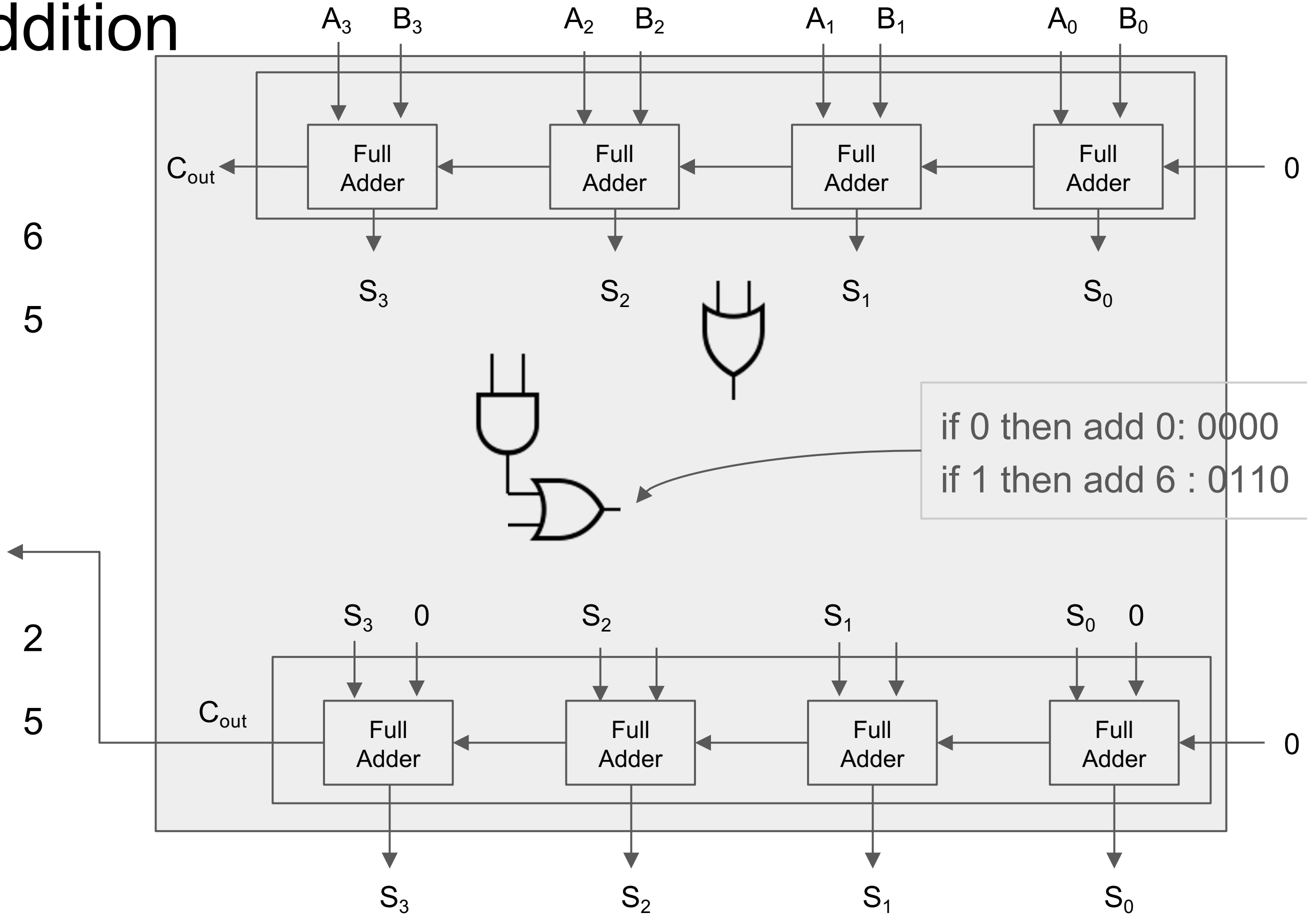
- Perform Regular Binary Addition, but account for the invalid patterns
- Add six upon whenever you are in the deadzone or there is overflow
  - Invalid $= S_3 * (S_2 + S_1)$
  - Overflow $= C_{out}$

Overflow = C_{out}

4-bit BCD Addition

	0	1	1	0	6
+	0	1	0	1	5

	0	0	1	0	2
+	0	1	0	1	5



if 0 then add 0: 0000
if 1 then add 6 : 0110