# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

$a' \cdot b \cdot c \cdot d'$

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

$a' \cdot b \cdot c \cdot d'$

$a \cdot b' \cdot c' \cdot d'$

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

a' · b · c · d'

a · b' · c' · d'

a · b' · c' · d

# Optimization (Karnaugh maps)

| a | b | c | d | e | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 0 | |
| 0 | 1 | 1 | 0 | 1 | a’ · b · c · d’ |
| 0 | 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | 1 | a · b’ · c’ · d’ |
| 1 | 0 | 0 | 1 | 1 | a · b’ · c’ · d |
| 1 | 0 | 1 | 0 | 1 | a · b’ · c · d’ |
| 1 | 0 | 1 | 1 | 1 | |
| 1 | 1 | 0 | 0 | 1 | |
| 1 | 1 | 0 | 1 | 1 | |
| 1 | 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 1 | 0 | |

# Optimization (Karnaugh maps)

| a | b | c | d | e | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 0 | |
| 0 | 1 | 1 | 0 | 1 | a’ · b · c · d’ |
| 0 | 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | 1 | a · b’ · c’ · d’ |
| 1 | 0 | 0 | 1 | 1 | a · b’ · c’ · d |
| 1 | 0 | 1 | 0 | 1 | a · b’ · c · d’ |
| 1 | 0 | 1 | 1 | 1 | a · b’ · c · d |
| 1 | 1 | 0 | 0 | 1 | a · b · c’ · d’ |
| 1 | 1 | 0 | 1 | 1 | a · b · c’ · d |
| 1 | 1 | 1 | 0 | 1 | a · b · c · d’ |
| 1 | 1 | 1 | 1 | 0 | |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

$a' \cdot b \cdot c \cdot d'$

$a \cdot b' \cdot c' \cdot d'$

$a \cdot b' \cdot c' \cdot d$

$a \cdot b' \cdot c \cdot d'$

$a \cdot b' \cdot c \cdot d$

$a \cdot b \cdot c' \cdot d'$

$a \cdot b \cdot c' \cdot d$

$a \cdot b \cdot c \cdot d'$

} or

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | | | 1 |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | | | 1 |
| 01 | | | | |
| 11 | | | 0 | |
| 10 | | | | |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | | | 1 |
| 01 | | | | |
| 11 | | | 0 | |
| 10 | | 1 | | |

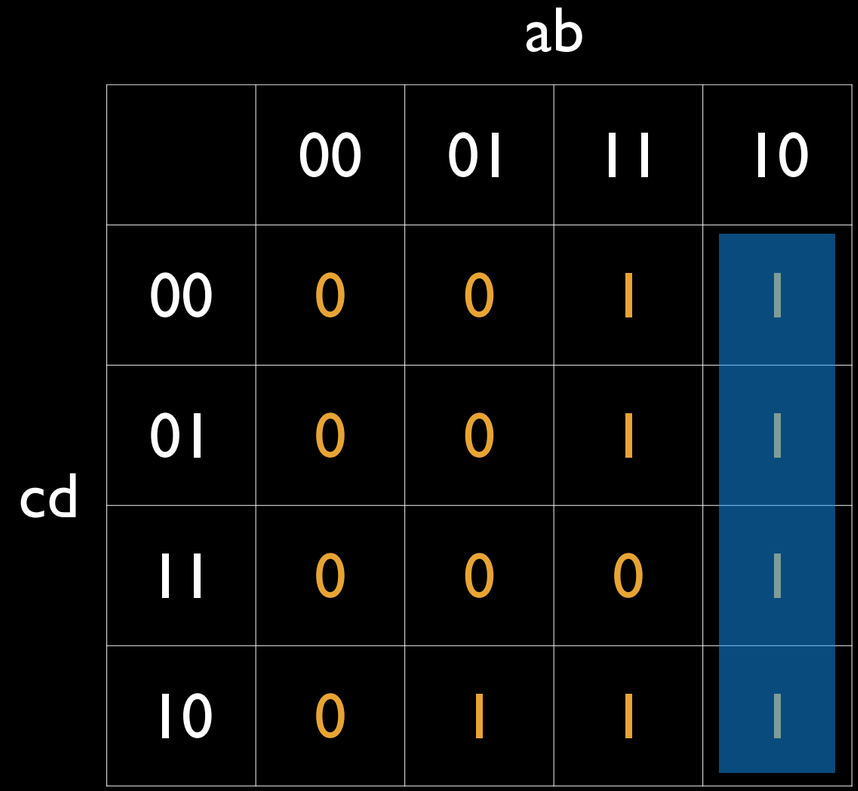
# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

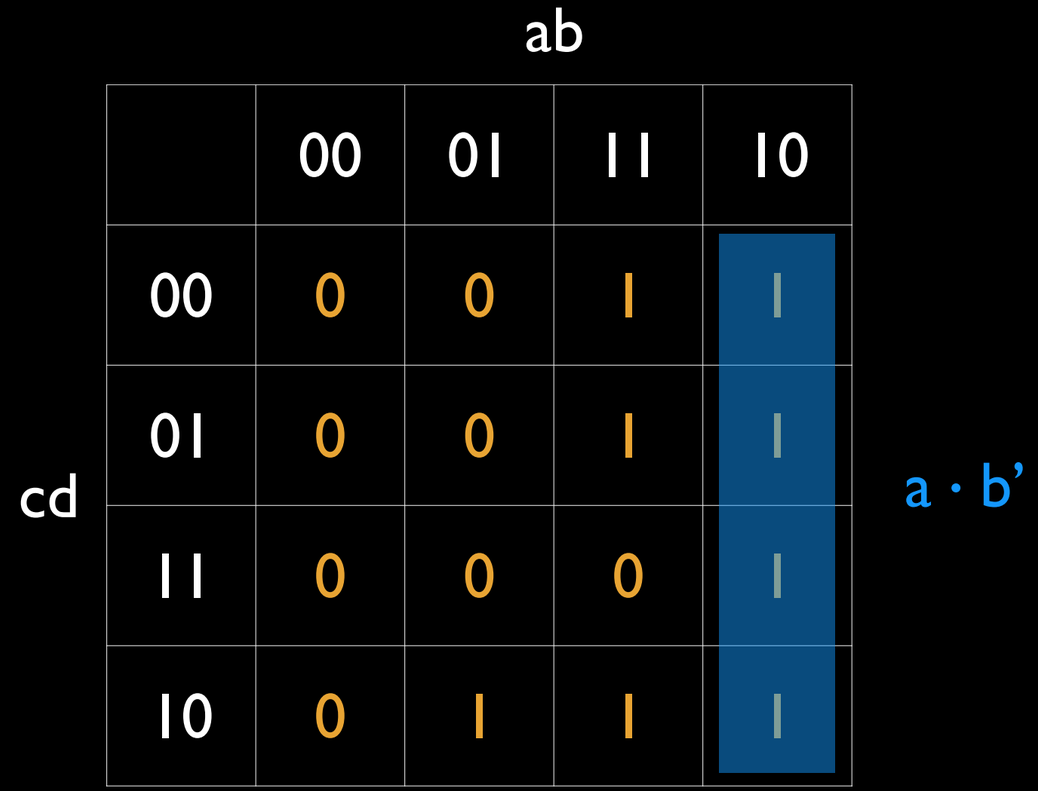
# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |



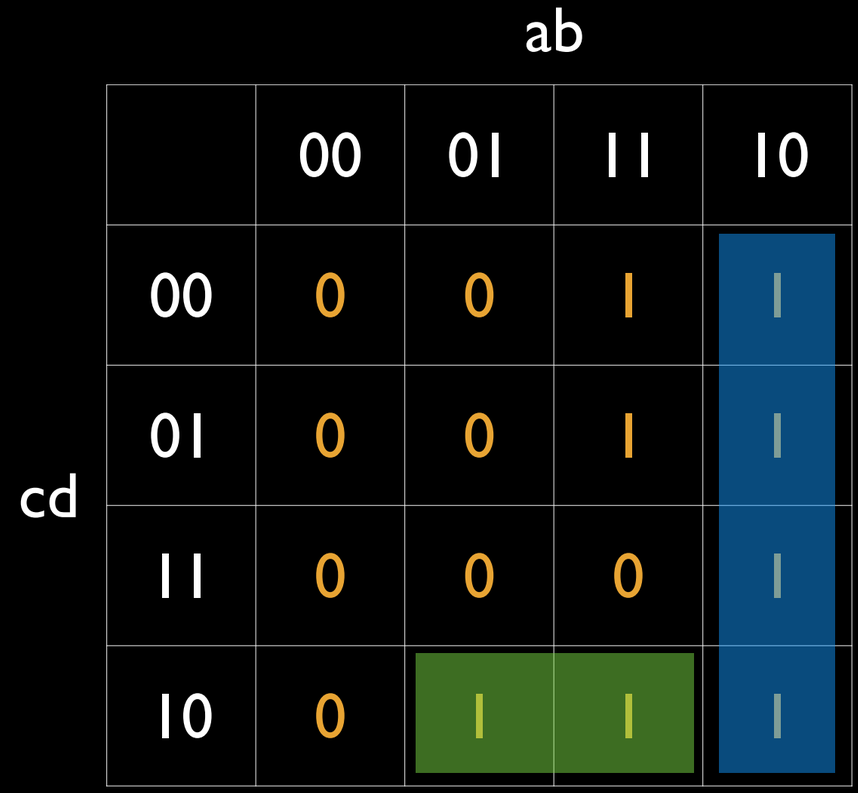
# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |



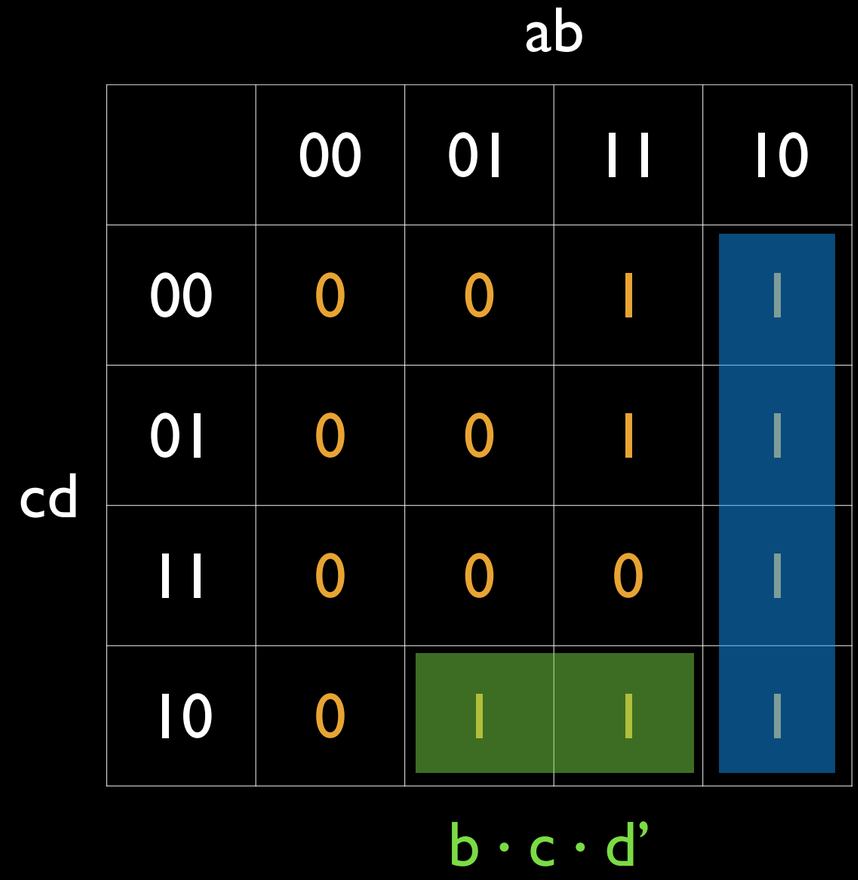
# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |



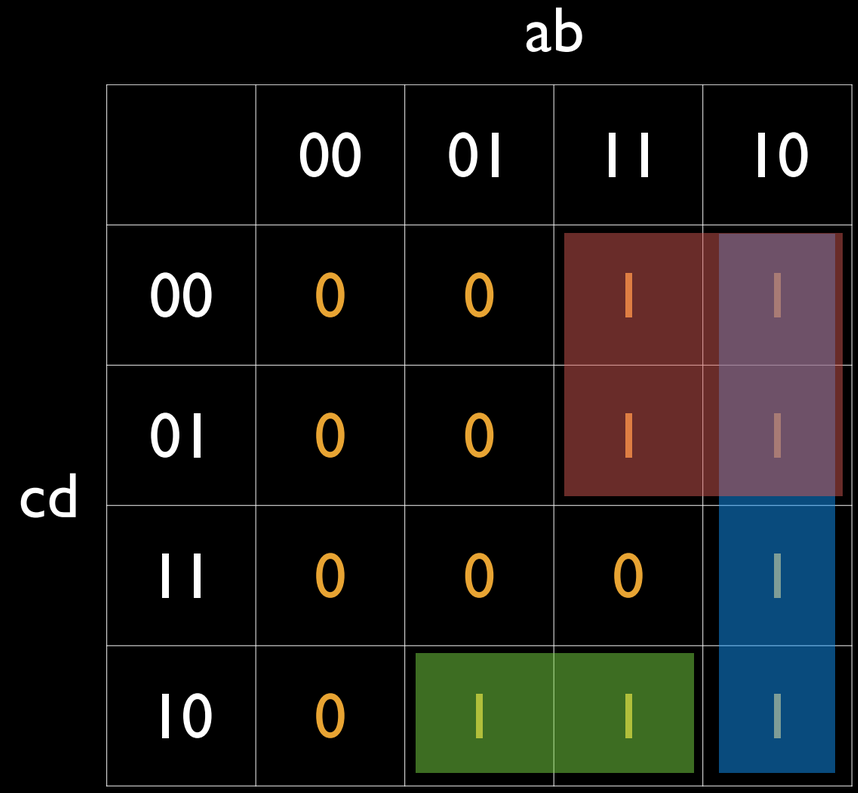
# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |



# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

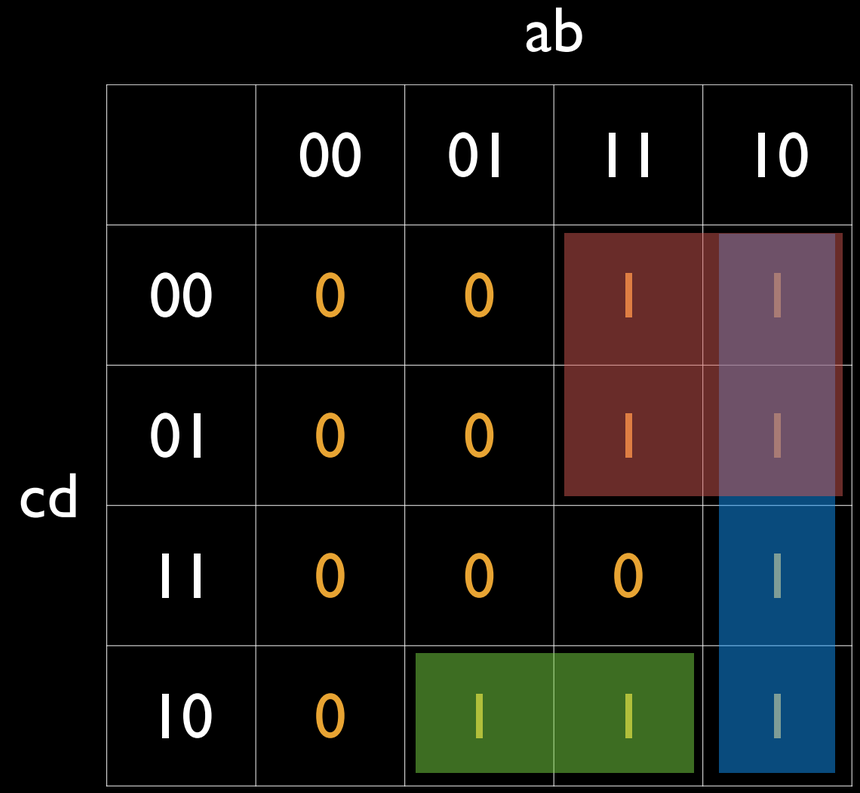


# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

# Optimization (Karnaugh maps)

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |




$e = a \cdot b' + b \cdot c \cdot d' + a \cdot c'$

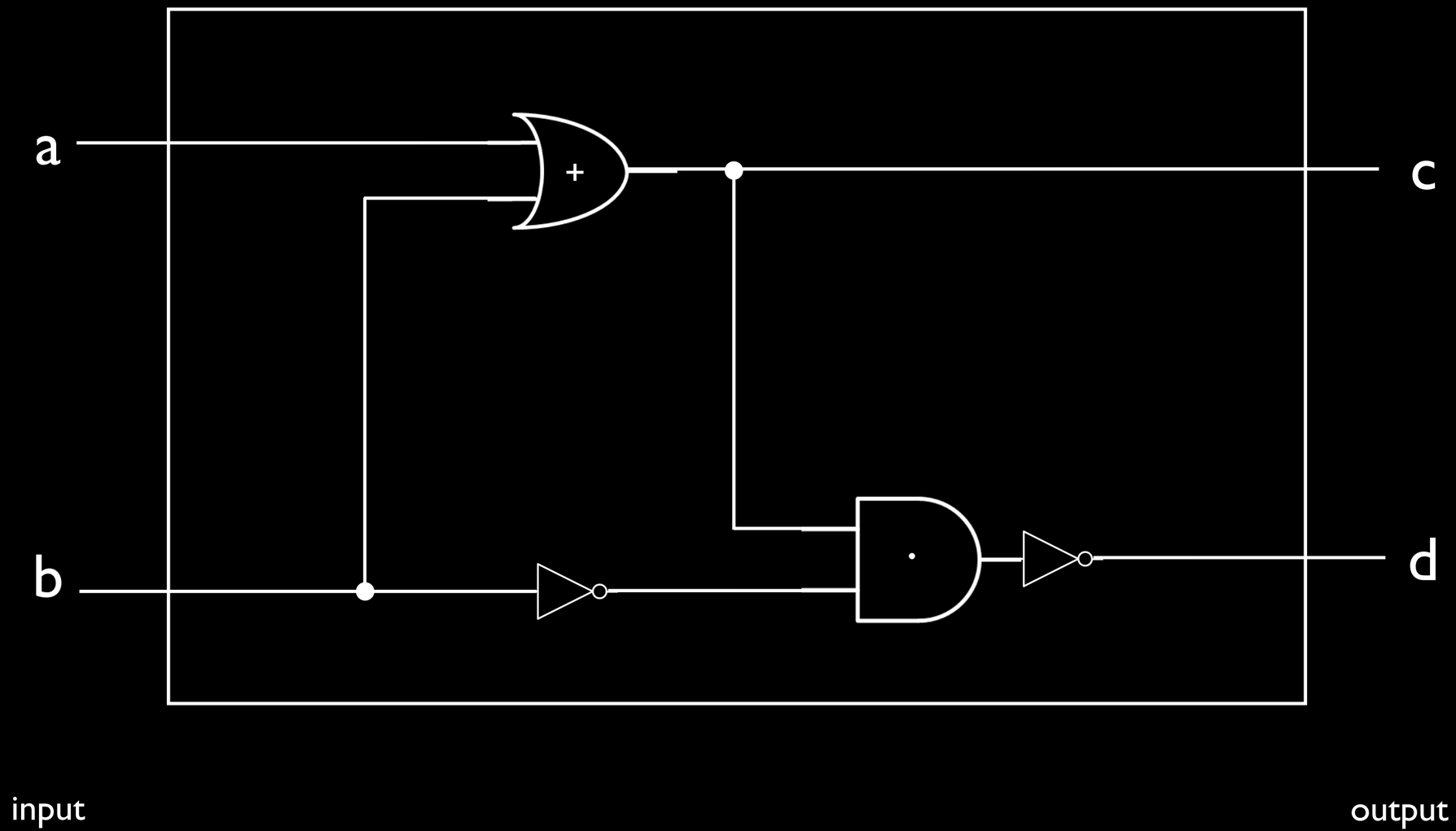
# Optimization (Karnaugh maps)

| a | b | c | d | e | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 0 | |
| 0 | 1 | 1 | 0 | 1 | a’ · b · c · d’ |
| 0 | 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | 1 | a · b’ · c’ · d’ |
| 1 | 0 | 0 | 1 | 1 | a · b’ · c’ · d |
| 1 | 0 | 1 | 0 | 1 | a · b’ · c · d’ |
| 1 | 0 | 1 | 1 | 1 | a · b’ · c · d |
| 1 | 1 | 0 | 0 | 1 | a · b · c’ · d’ |
| 1 | 1 | 0 | 1 | 1 | a · b · c’ · d |
| 1 | 1 | 1 | 0 | 1 | a · b · c · d’ |
| 1 | 1 | 1 | 1 | 0 | |

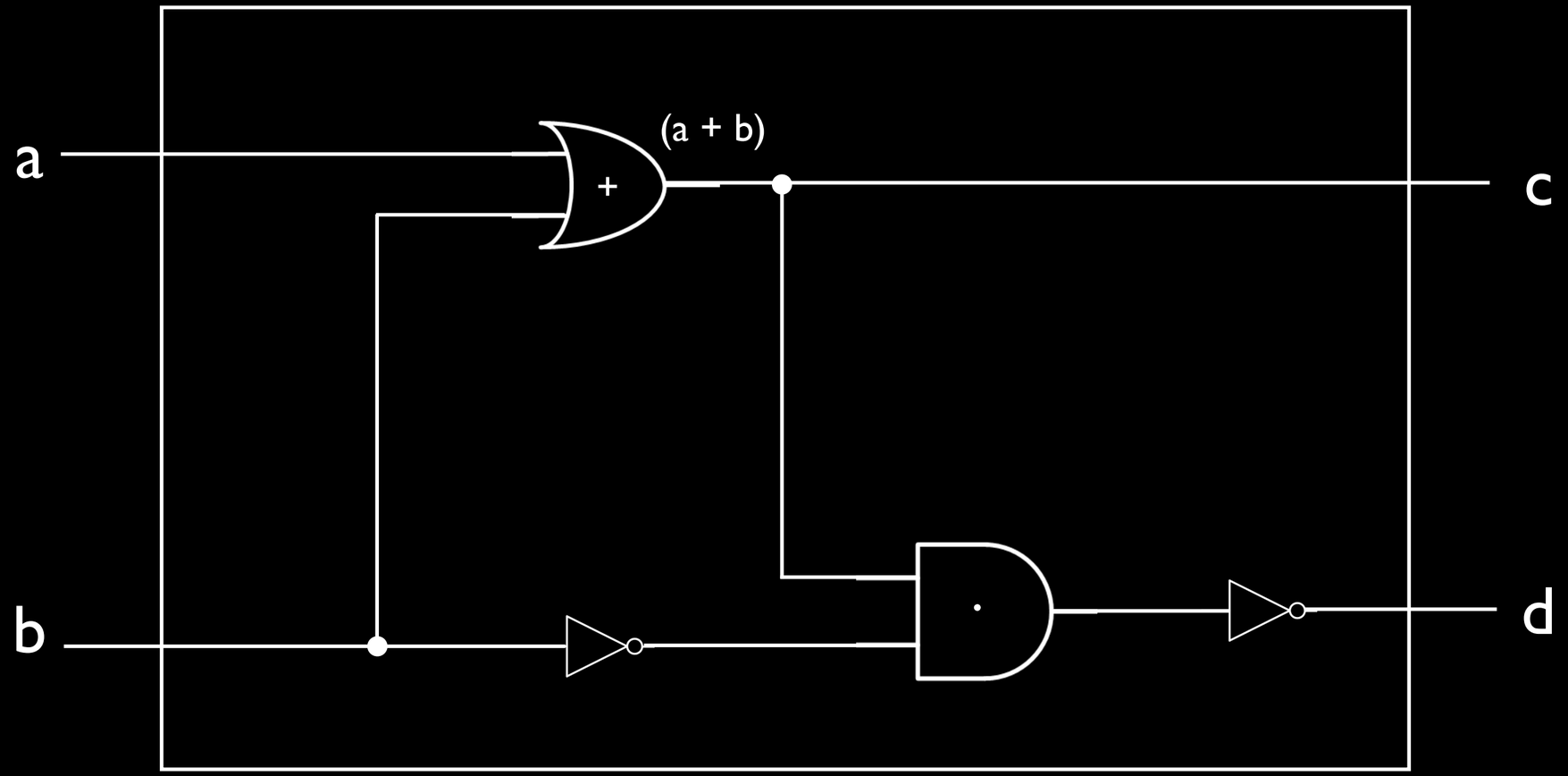
# Circuits

A circuit is a collection of logical gates that transforms a set of binary inputs into a set of binary outputs.

# Circuits

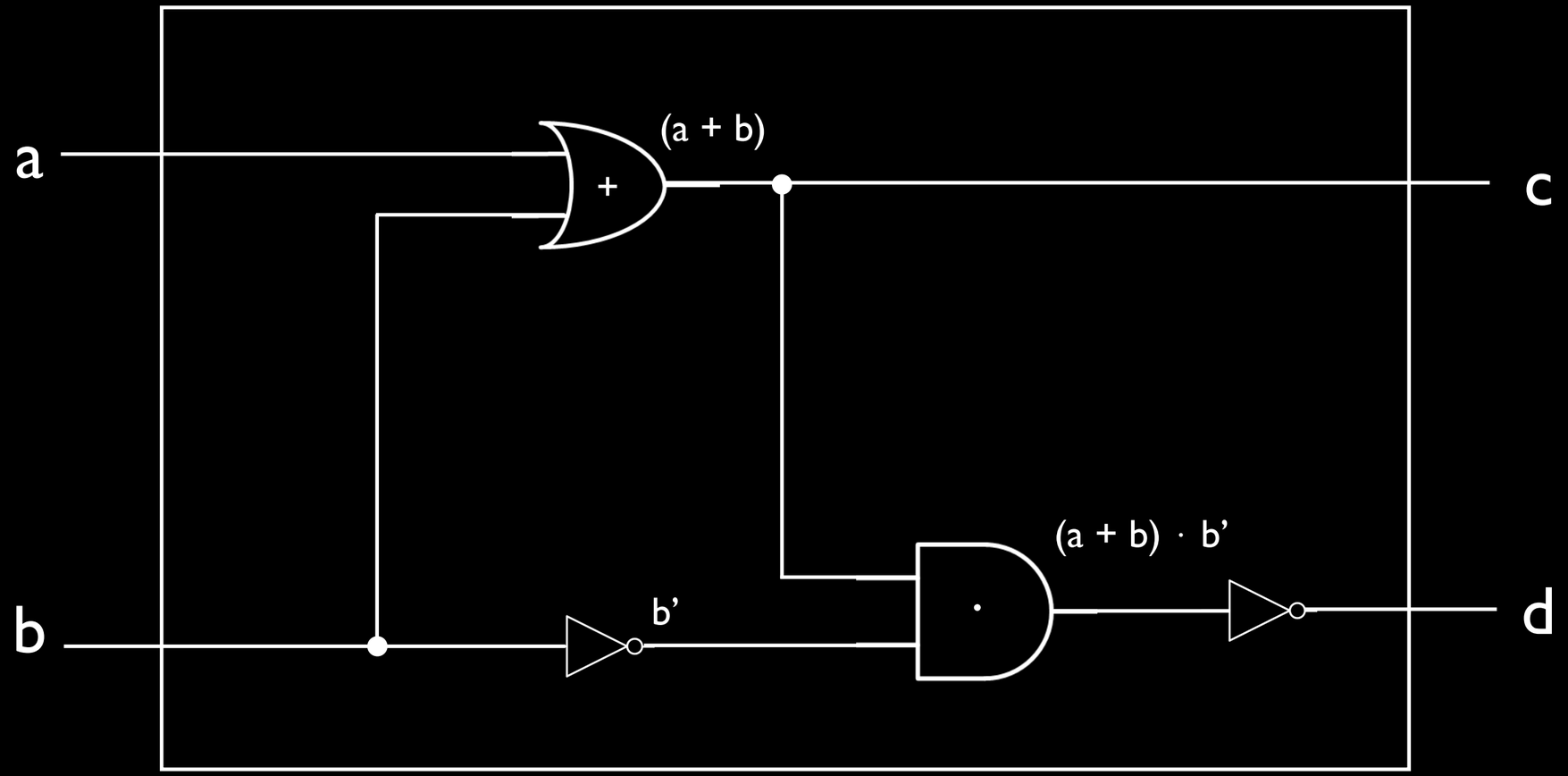


# Circuits

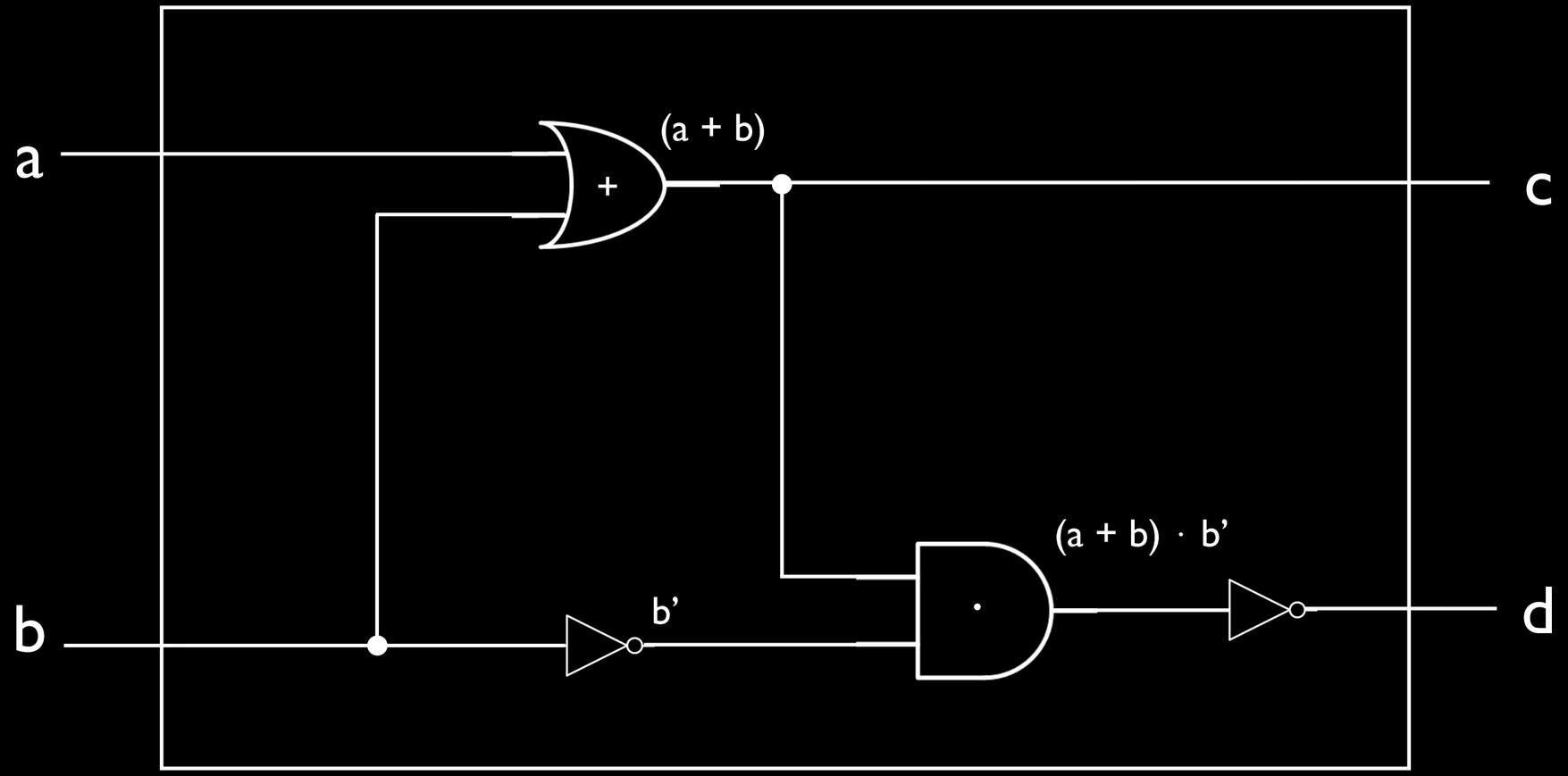



$$c = (a + b)$$

# Circuits



# Circuits




$$d = ((a + b) \cdot b')'$$

# Designing Circuits

| input | | output | |
|---|---|---|---|
| a | b | c | d |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

step 1. build truth-table for all possible input/output values

# Designing Circuits

input output

| a | b | c | d |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

arbitrary, for now

step 1. build truth-table for all possible input/output values

# Designing Circuits

input output

| a | b | c | d |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

arbitrary, for now

step 2. build sub-expressions with *and*/*not* for each output column

# Designing Circuits

| input | | output | |
|---|---|---|---|
| a | b | c | d |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$d = a \cdot b$

step 2. build sub-expressions with *and*/*not* for each output column

# Designing Circuits

| input | | output | |
|---|---|---|---|
| a | b | c | d |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$c = a' \cdot b$

step 2. build sub-expressions with *and*/*not* for each output column

# Designing Circuits

| input | | output | |
|---|---|---|---|
| a | b | c | d |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$c = a' \cdot b$

$c = a \cdot b'$

step 2. build sub-expressions with *and*/*not* for each output column

# Designing Circuits

| input | | | output |
|---|---|---|---|
| a | b | c | d |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

step 3. combine, two at a time, sub-expressions with an *or*

# Designing Circuits

| input | | output | |
|---|---|---|---|
| a | b | c | d |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$$c = (a' \cdot b) + (a \cdot b')$$

step 3. combine, two at a time, sub-expressions with an *or*

# Designing Circuits

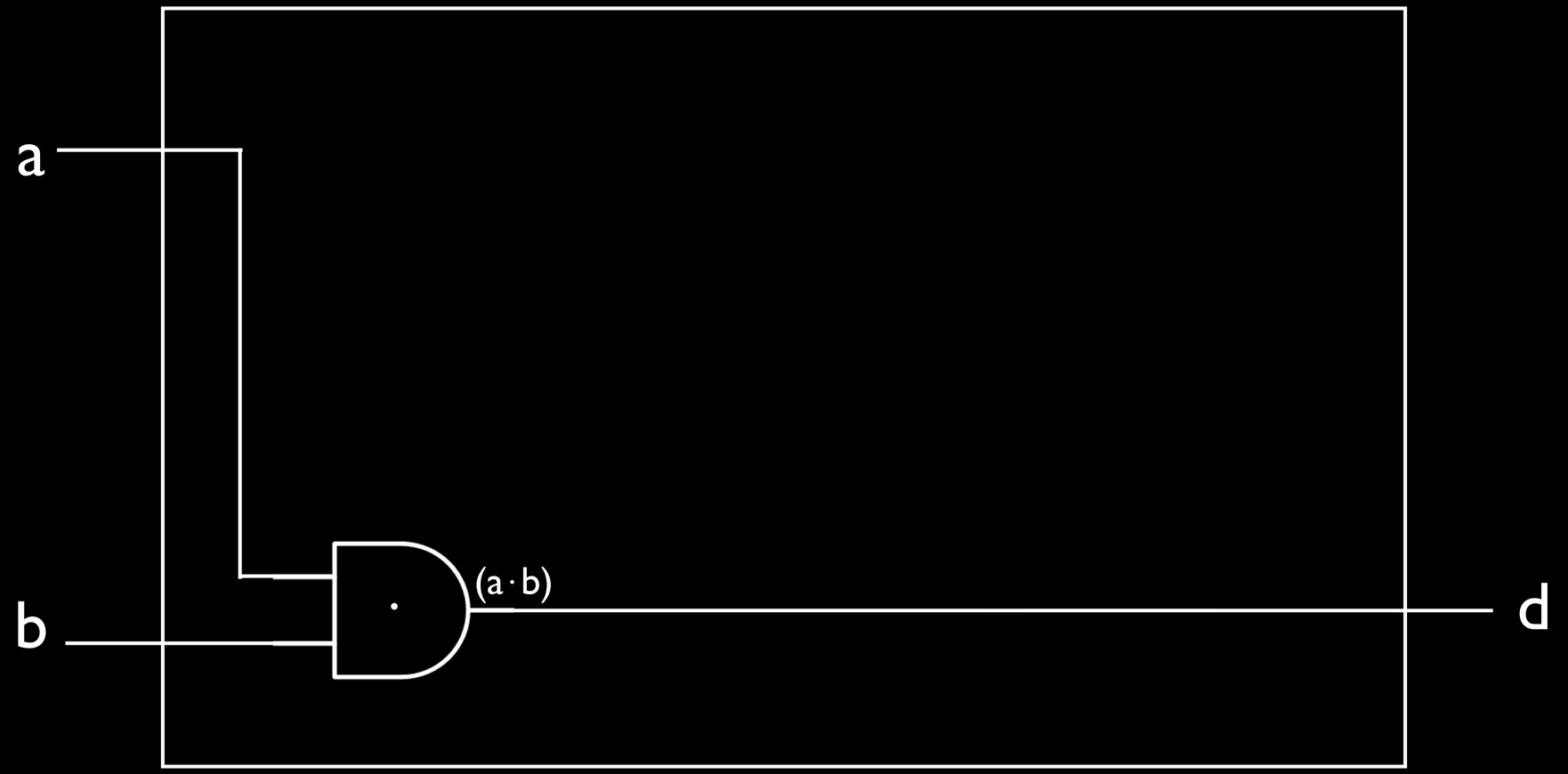
| input | | output | |
|---|---|---|---|
| a | b | c | d |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$c = (a' \cdot b) + (a \cdot b')$

$d = a \cdot b$

step 3. combine, two at a time, sub-expressions with an *or*

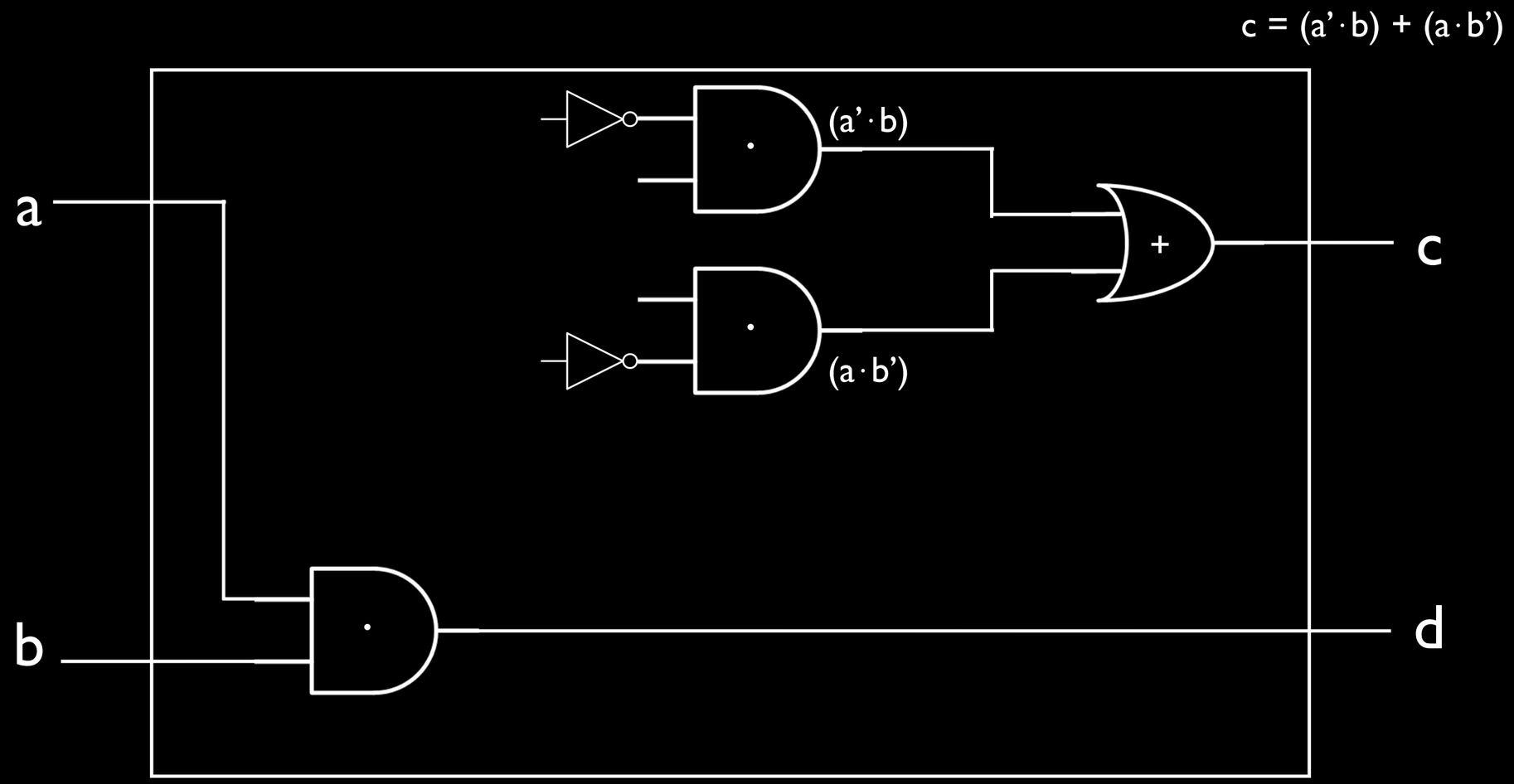
# Designing Circuits




$d = a \cdot b$

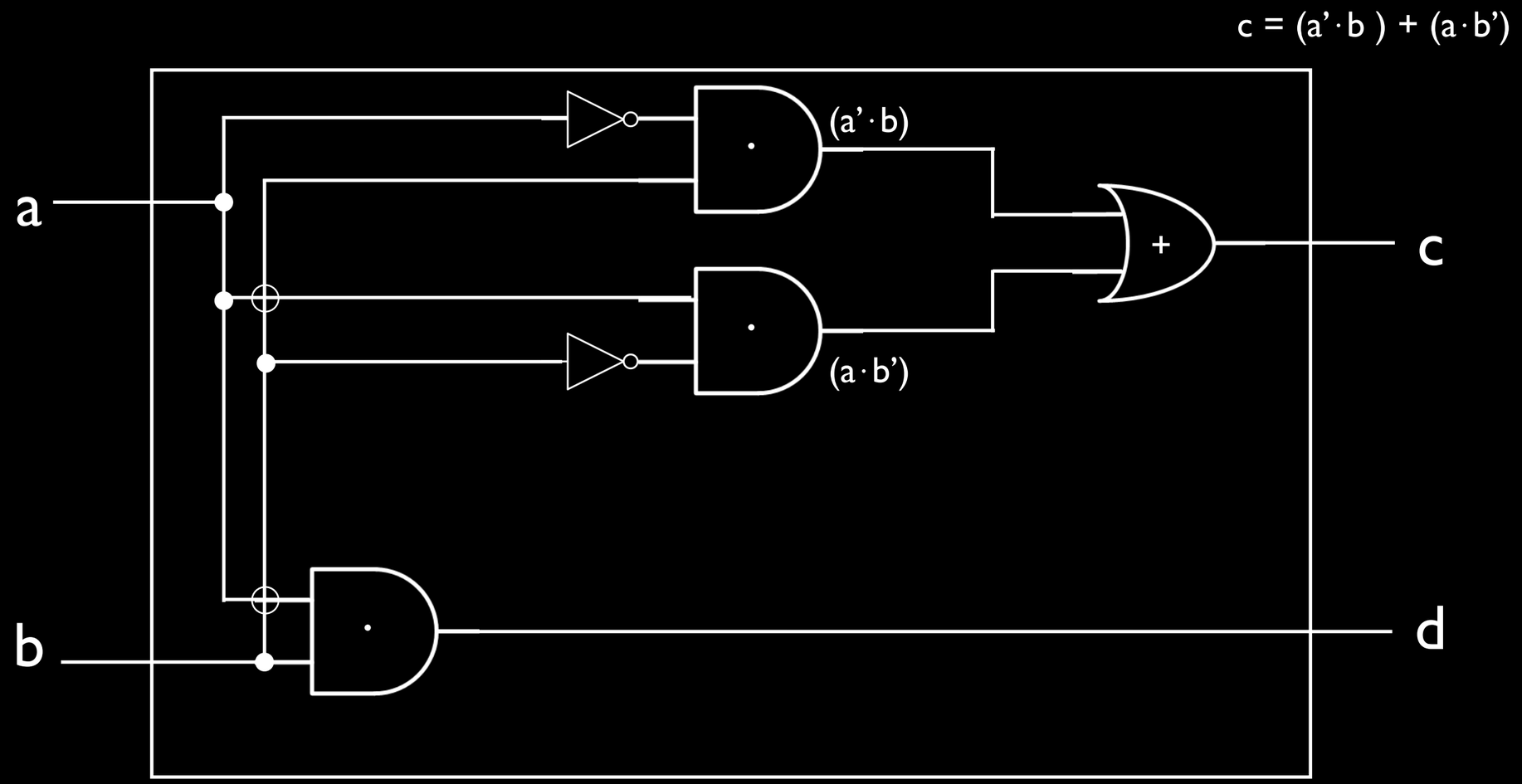
step 4. draw circuit diagram

# Designing Circuits




step 4. draw circuit diagram

# Designing Circuits




step 4. draw circuit diagram

# Designing Circuits

step 1. build truth-table for all possible input/output values

step 2. build sub-expressions with *and*/*not* for each output column

step 3. combine, two at a time, sub-expressions with an *or*

step 4. draw circuit diagram

# 1-Bit Compare for Equality (CE)

If two bits *a, b* are equal then
    return 1
else
    return 0

# 1-Bit Compare for Equality (CE)

| input | | output |
|---|---|---|
| a | b | c |
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

# 1-Bit Compare for Equality (CE)

| input | | output |
|---|---|---|
| a | b | c |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# 1-Bit Compare for Equality (CE)

| input | | output | |
|---|---|---|---|
| a | b | c | sub-expression |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 1 | 0 | 0 | |
| 1 | 1 | 1 | |

# 1-Bit Compare for Equality (CE)

| input | | output | |
|---|---|---|---|
| a | b | c | sub-expression |
| 0 | 0 | 1 | a’ · b’ |
| 0 | 1 | 0 | |
| 1 | 0 | 0 | |
| 1 | 1 | 1 | a · b |

# 1-Bit Compare for Equality (CE)

| input | | output | |
|---|---|---|---|
| a | b | c | sub-expression |
| 0 | 0 | 1 | a’ · b’ |
| 0 | 1 | 0 | |
| 1 | 0 | 0 | |
| 1 | 1 | 1 | a · b |

$$c = (a' \cdot b') + (a \cdot b)$$

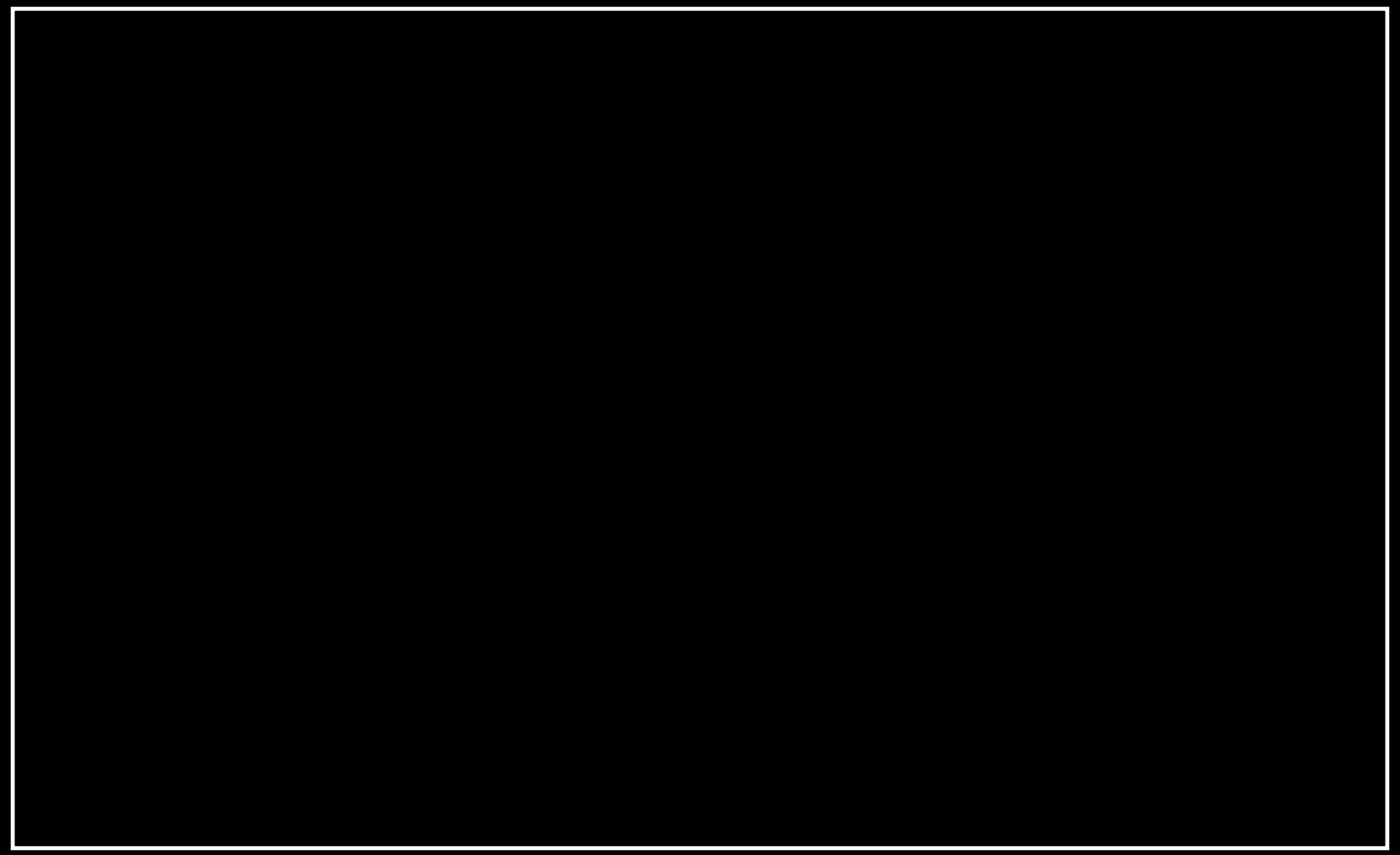
# 1-Bit Compare for Equality (CE)

input

output

a

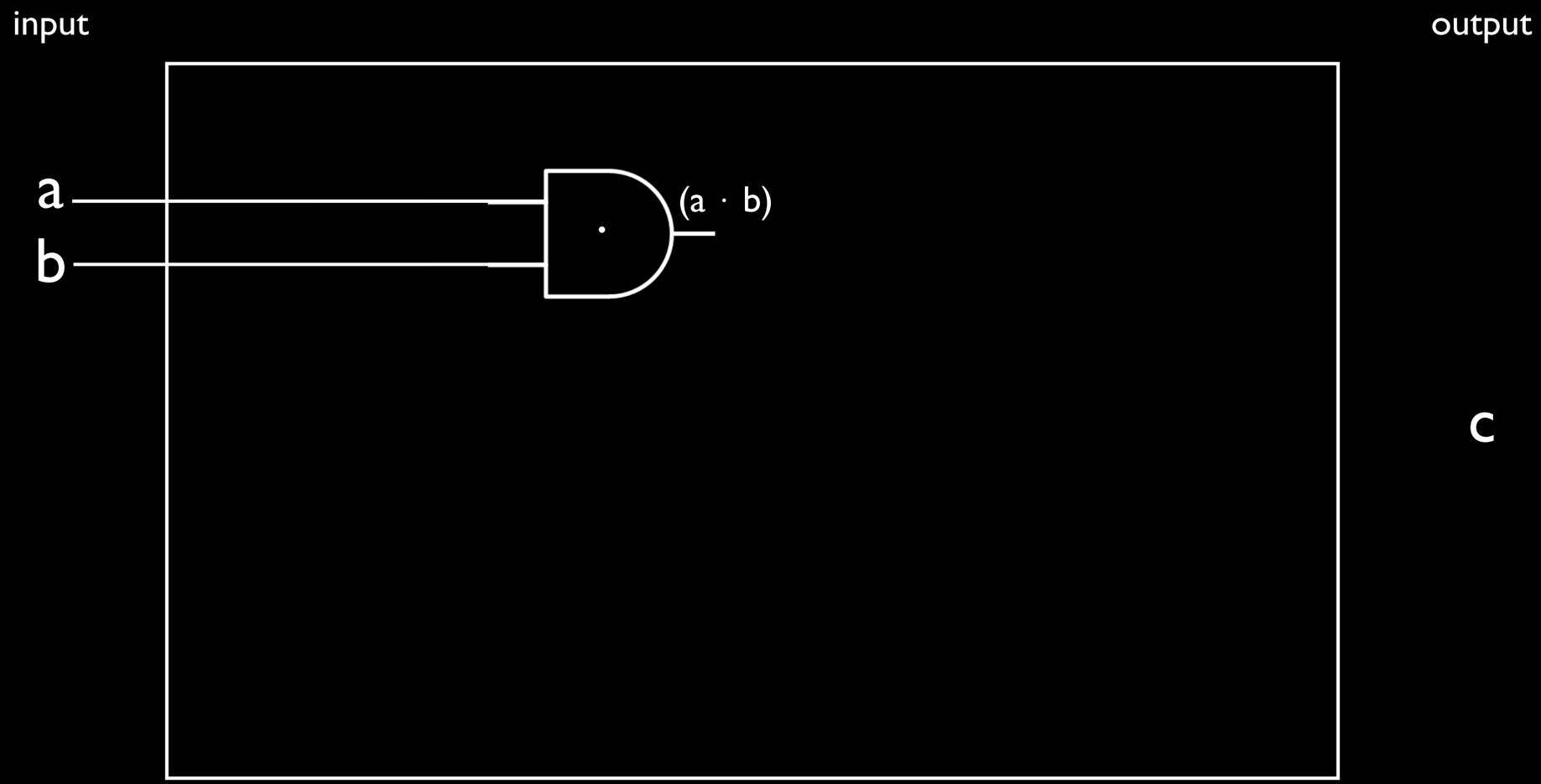
b



c

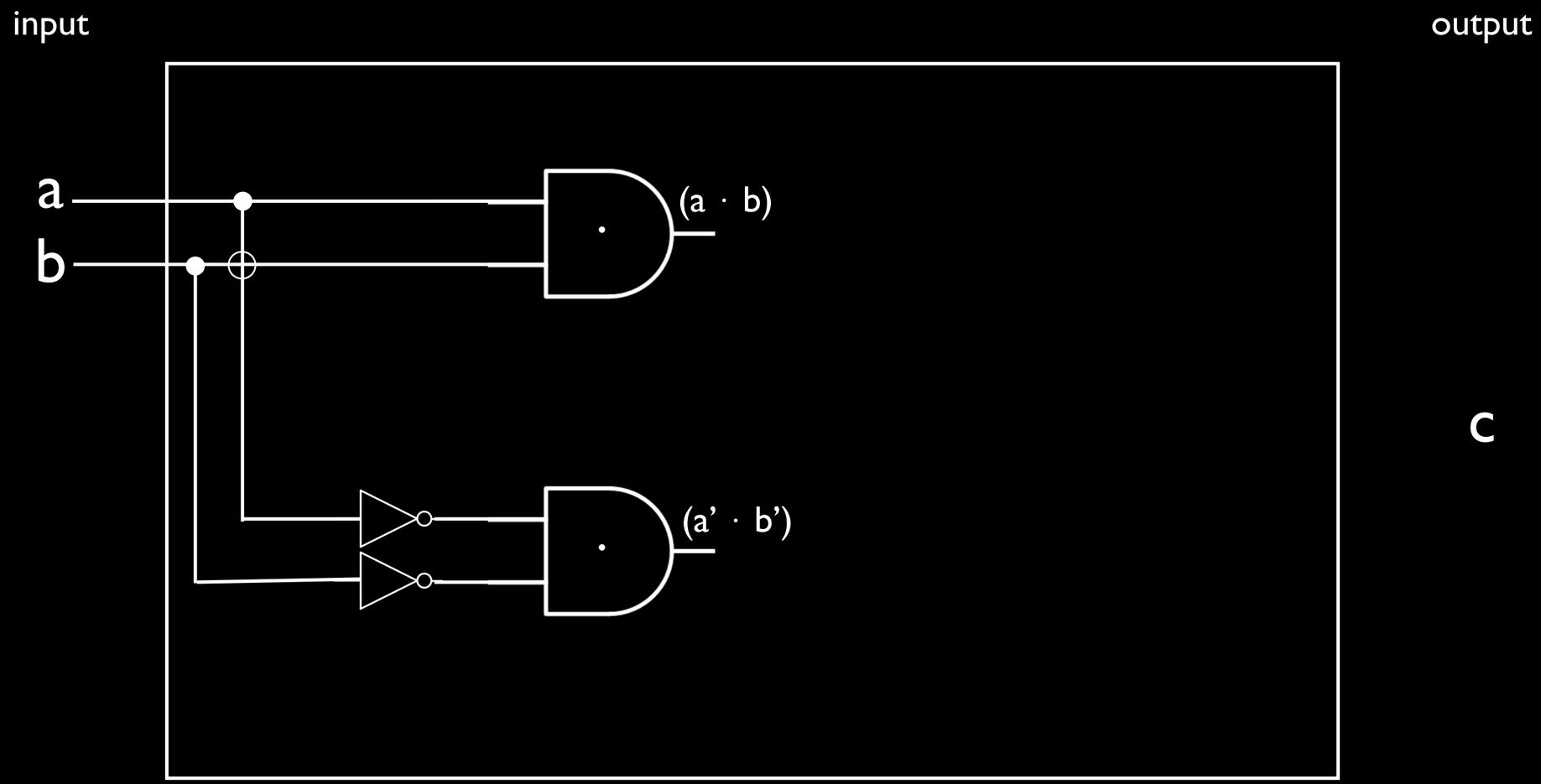
$$c = (a' \cdot b') + (a \cdot b)$$

# 1-Bit Compare for Equality (CE)



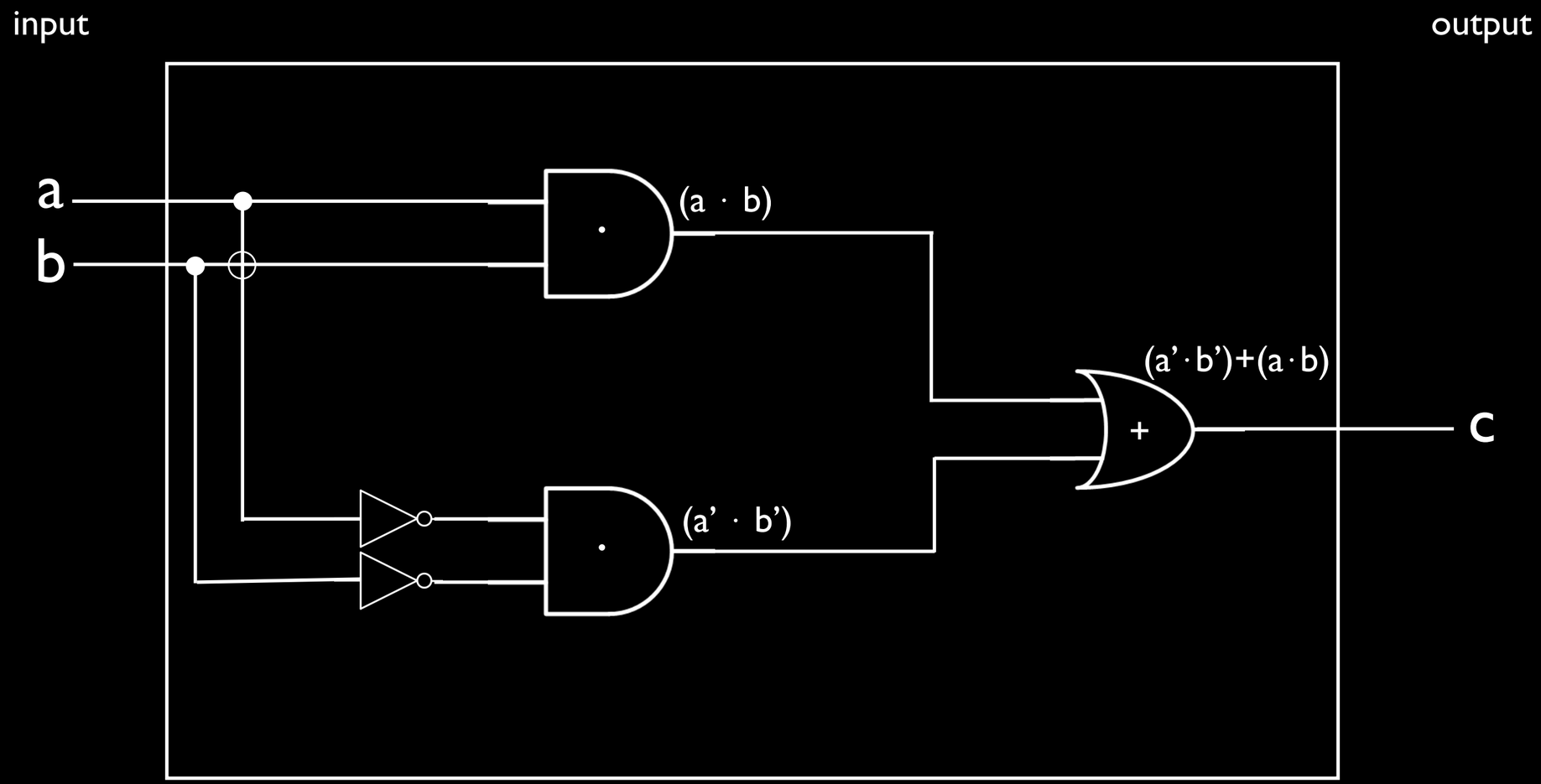

$$c = (a' \cdot b') + (a \cdot b)$$

# 1-Bit Compare for Equality (CE)




$$c = (a' \cdot b') + (a \cdot b)$$

# 1-Bit Compare for Equality (CE)




$$c = (a' \cdot b') + (a \cdot b)$$

# 4-Bit Compare for Equality (CE)

```
If two 4-bit numbers are equal then
      return  1
else
      return 0
```

# 4-Bit Compare for Equality (CE)

If two 4-bit numbers are equal then
    return 1
else
    return 0

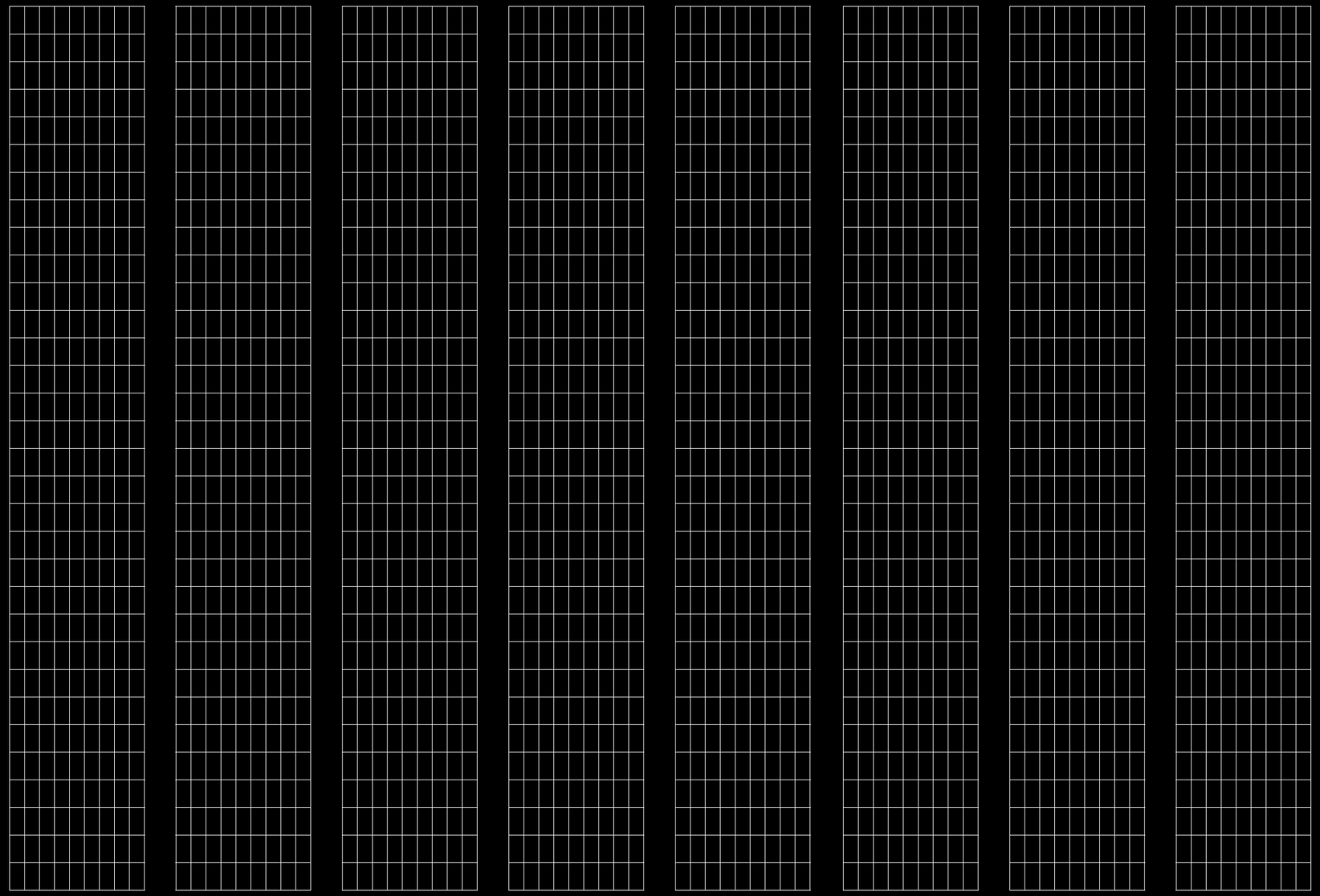
$a_3\ a_2\ a_1\ a_0 == b_3\ b_2\ b_1\ b_0$

# 4-Bit Compare for Equality (CE)

| input | | | | | | | | output |
|---|---|---|---|---|---|---|---|---|
| $a_3$ | $a_2$ | $a_1$ | $a_0$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ | c |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

how many rows?

# 4-Bit Compare for Equality (CE)



256

# 4-Bit Compare for Equality (CE)

two 4-bit numbers are equal if:

$a_3 == b_3$ and

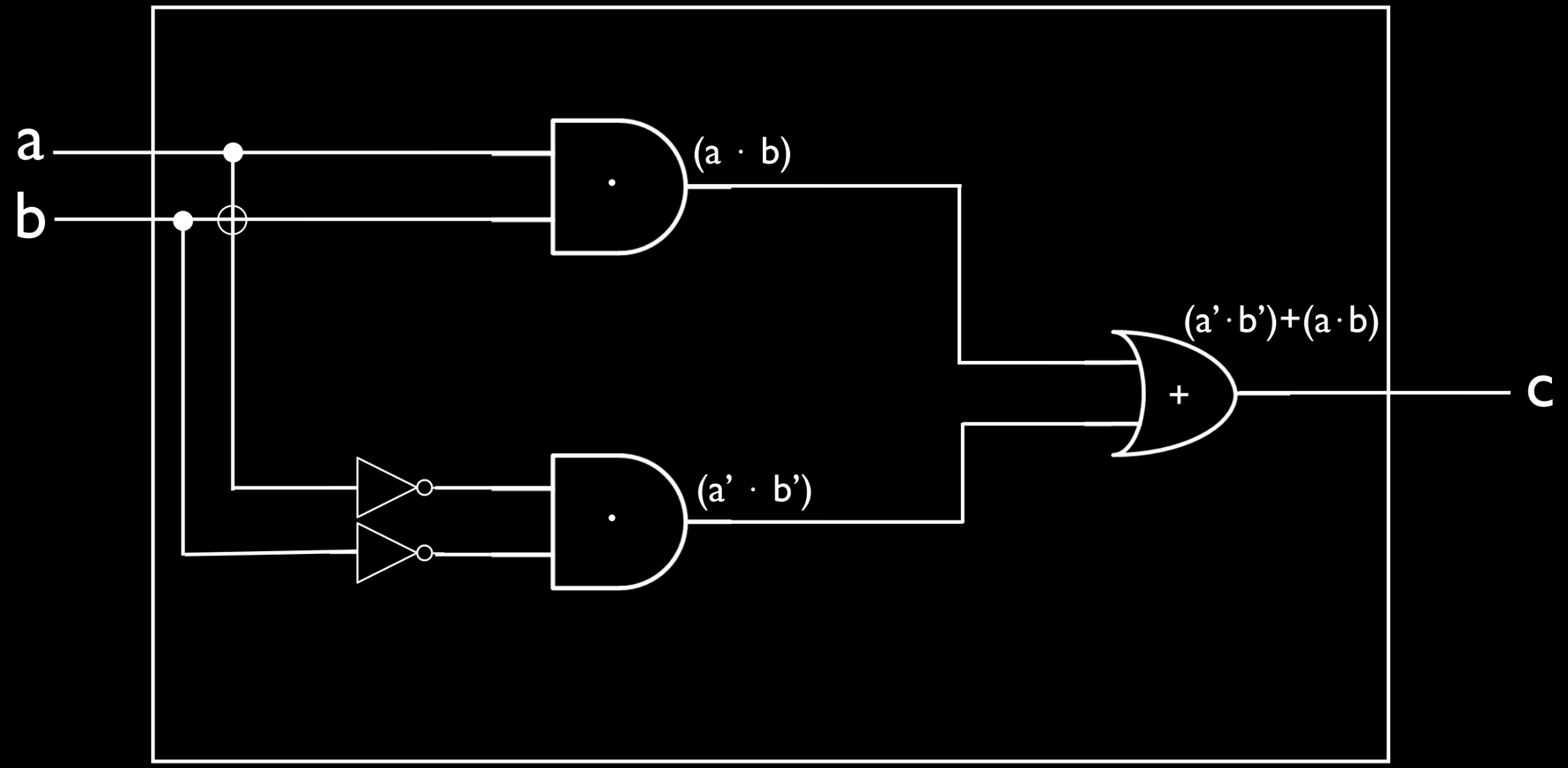
$a_2 == b_2$ and

$a_1 == b_1$ and

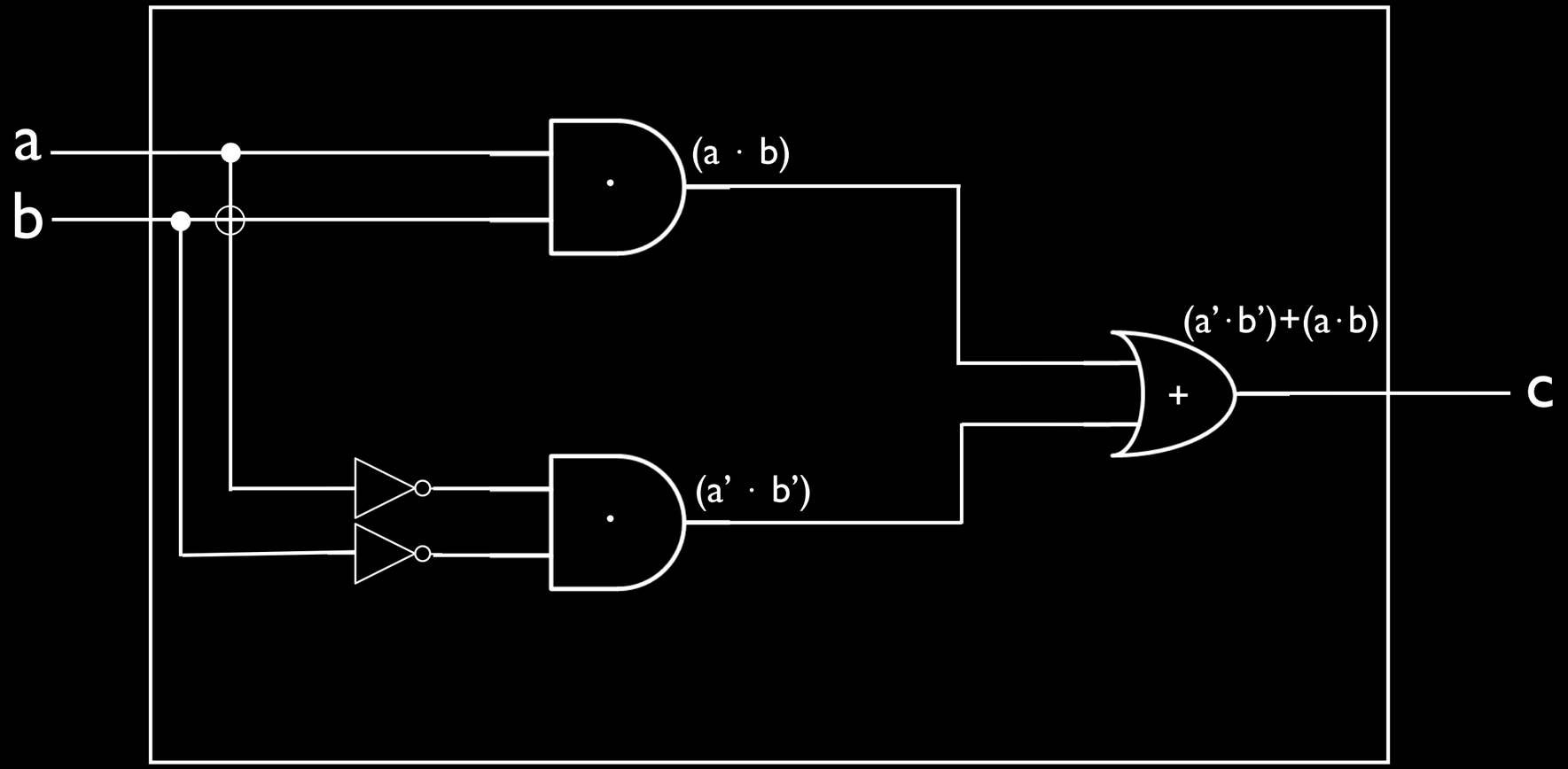
$a_0 == b_0$

$a_3\ a_2\ a_1\ a_0 == b_3\ b_2\ b_1\ b_0$

# 4-Bit Compare for Equality (CE)



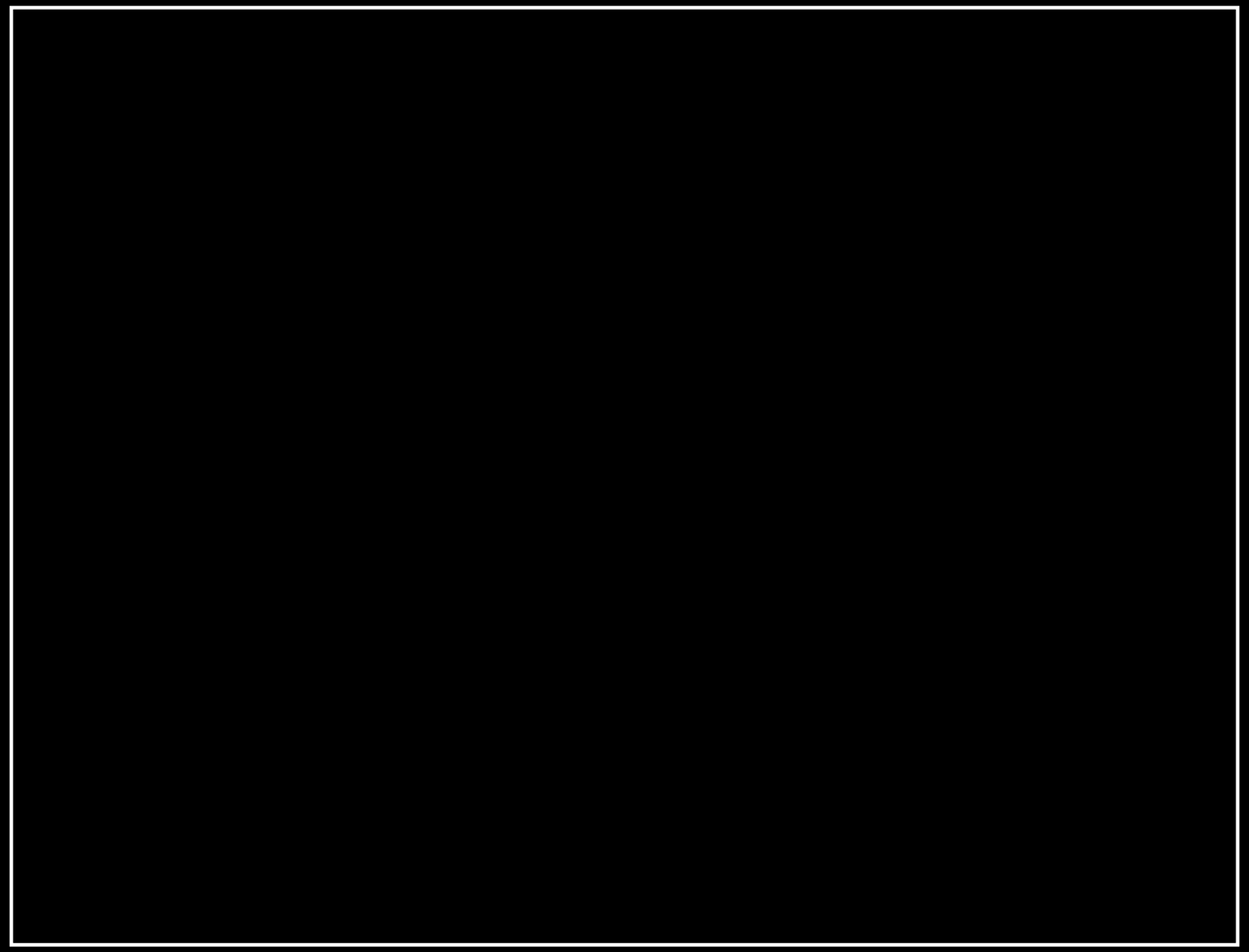
# 4-Bit Compare for Equality (CE)



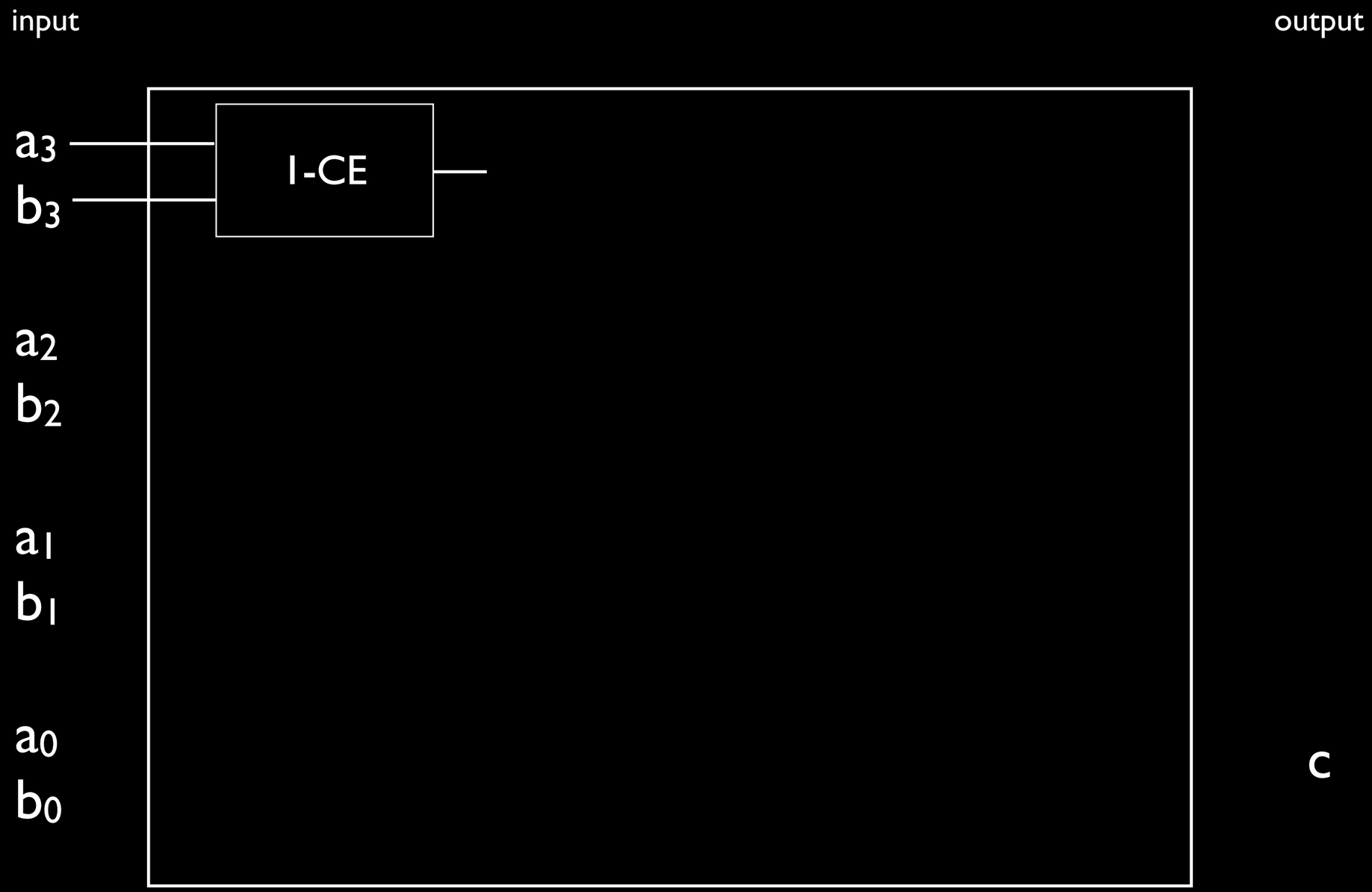

# 4-Bit Compare for Equality (CE)

input

output

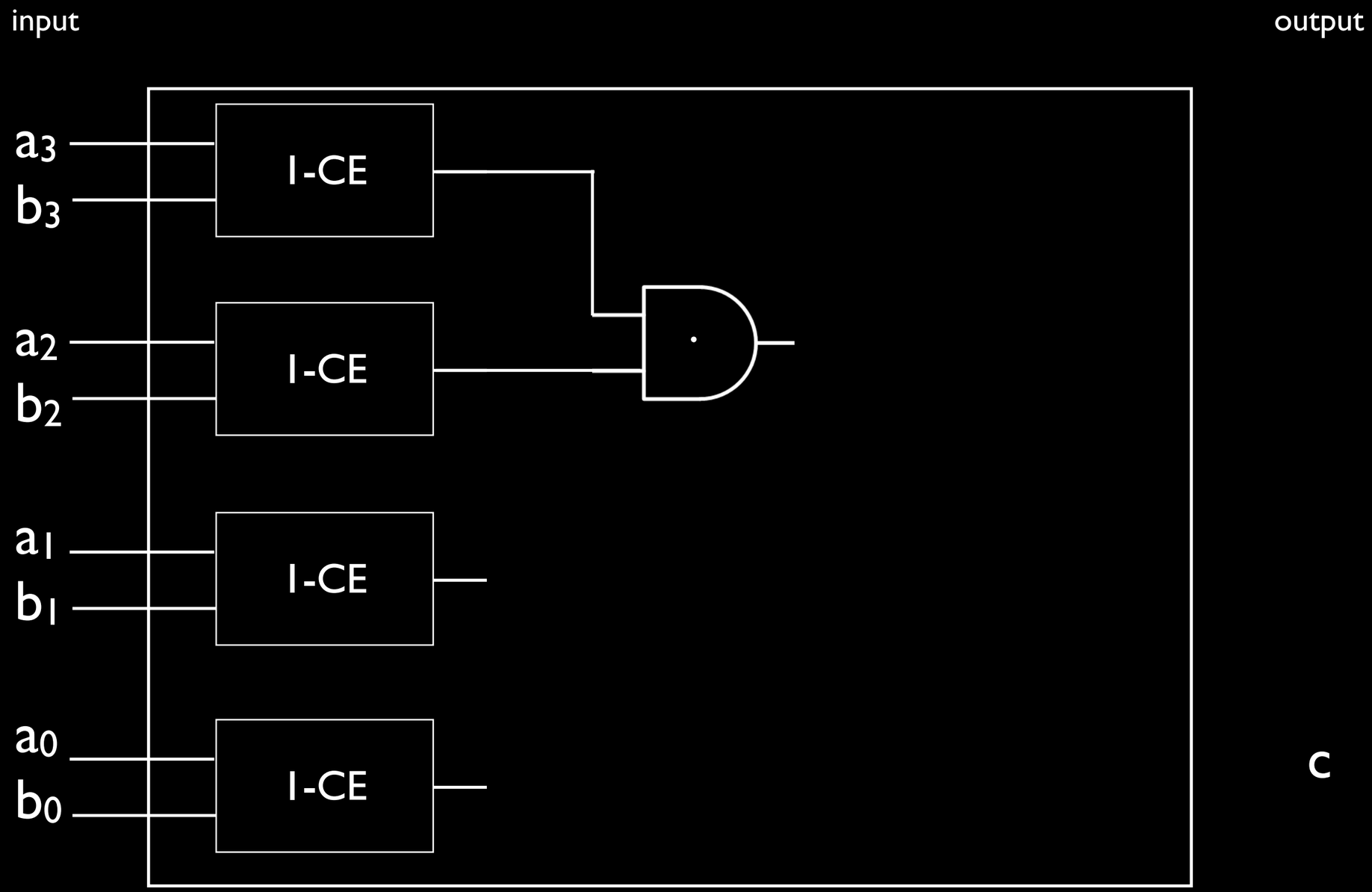


# 4-Bit Compare for Equality (CE)

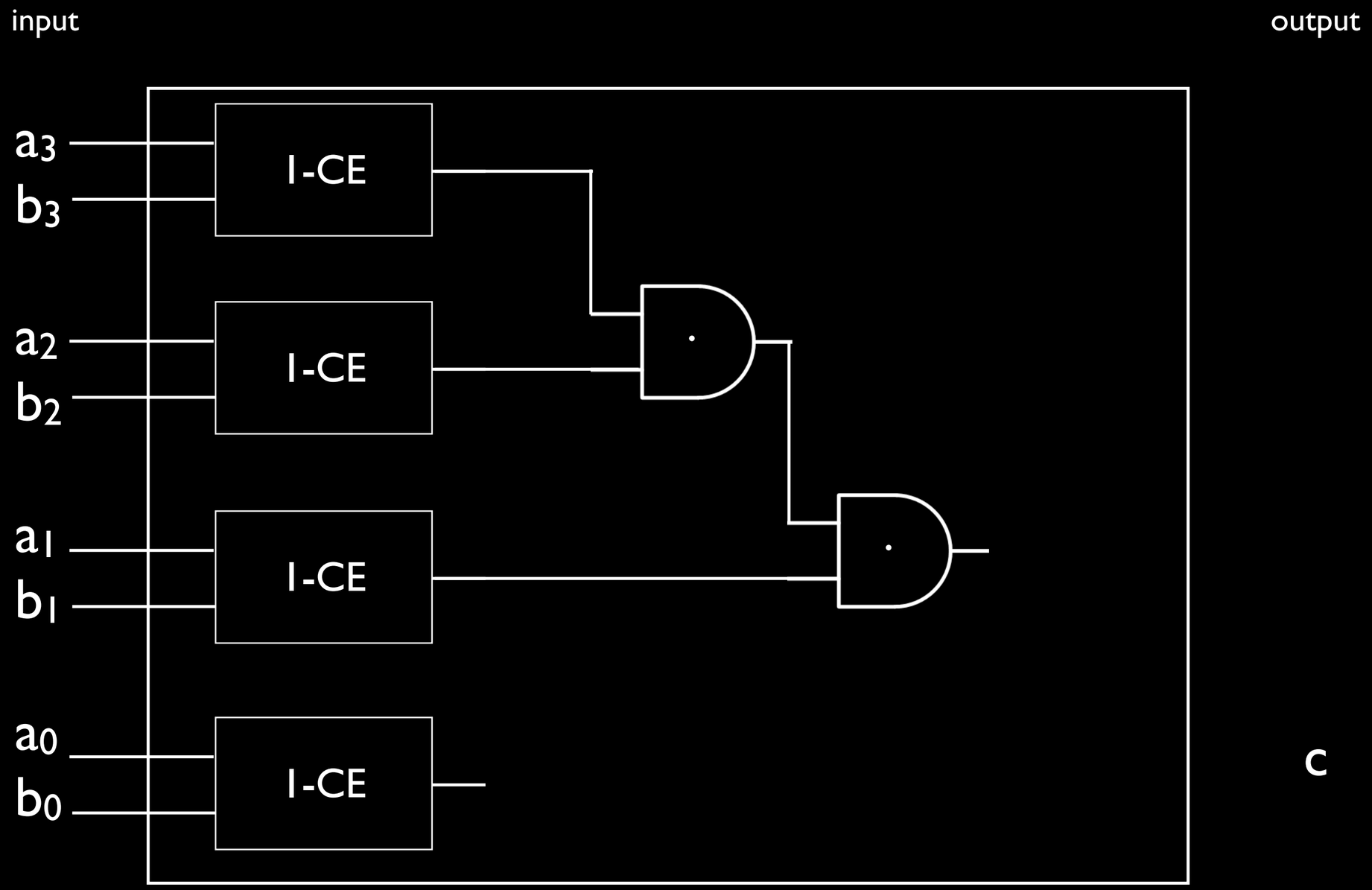


# 4-Bit Compare for Equality (CE)

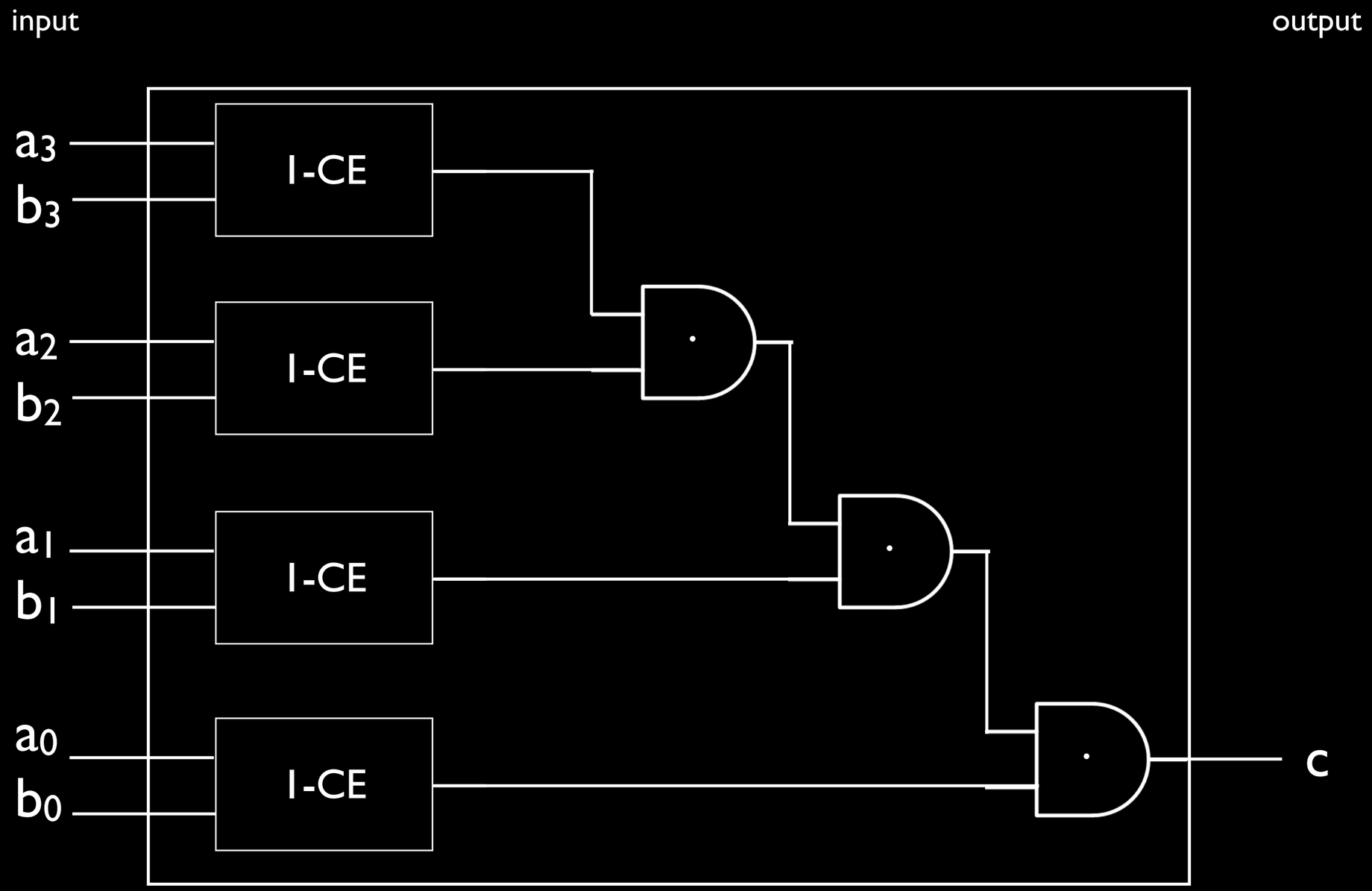
# 4-Bit Compare for Equality (CE)



# 4-Bit Compare for Equality (CE)



# 4-Bit Compare for Equality (CE)



# 1-Bit Adder

build a circuit that adds two 1-bit numbers

# 1-Bit Adder

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = ?

# 1-Bit Adder

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 *need to carry*

# 1-Bit Adder

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

input: two digits *a*, *b*

# 1-Bit Adder

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

input: two digits *a*, *b* and a carry *c*

# 1-Bit Adder

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

output: sum *d* and carry e

# 1-Bit Adder

| | | |
|---|---|---|
| 5 | 101 | $2^2+2^0$ |
| 1 | 001 | $2^0$ |
| --- | --- | |
| 6 | | |

# 1-Bit Adder

```
          1
  5      101
  1      001
 ---     ---
  6        0
```

# 1-Bit Adder

```
    1
  5       101
  1       001
---       ---
  6        10
```

# 1-Bit Adder

```
      1
  5   101
  1   001
---   ---
  6   110
```

# 1-Bit Adder

```
          1
  5      101
  1      001
 ---     ---
  6      110      2²+2¹
```

# 1-Bit Adder

**input**: digits *a*, *b* and carry *c*

**output**: sum *d* and carry *e*

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 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 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 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 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 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 | | |
| 1 | 1 | 1 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 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 | | |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e |
|---|---|---|---|---|
| 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 |

**input**: digits *a*, *b* and carry *c*
**output**: sum *d* and carry e

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 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 | | |

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 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 | | |

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | a’ · b’ · c | |
| 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 | | |

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | a’·b’·c | |
| 0 | 1 | 0 | 1 | 0 | a’·b·c’ | |
| 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 | | |

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | a’·b’·c | |
| 0 | 1 | 0 | 1 | 0 | a’·b·c’ | |
| 0 | 1 | 1 | 0 | 1 | | |
| 1 | 0 | 0 | 1 | 0 | a·b’·c’ | |
| 1 | 0 | 1 | 0 | 1 | | |
| 1 | 1 | 0 | 0 | 1 | | |
| 1 | 1 | 1 | 1 | 1 | | |

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | a’·b’·c | |
| 0 | 1 | 0 | 1 | 0 | a’·b·c’ | |
| 0 | 1 | 1 | 0 | 1 | | |
| 1 | 0 | 0 | 1 | 0 | a·b’·c’ | |
| 1 | 0 | 1 | 0 | 1 | | |
| 1 | 1 | 0 | 0 | 1 | | |
| 1 | 1 | 1 | 1 | 1 | a·b·c | |

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | a’·b’·c | |
| 0 | 1 | 0 | 1 | 0 | a’·b·c’ | |
| 0 | 1 | 1 | 0 | 1 | | a’·b·c |
| 1 | 0 | 0 | 1 | 0 | a·b’·c’ | |
| 1 | 0 | 1 | 0 | 1 | | a·b’·c |
| 1 | 1 | 0 | 0 | 1 | | a·b·c’ |
| 1 | 1 | 1 | 1 | 1 | a·b·c | a·b·c |

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | a’·b’·c | |
| 0 | 1 | 0 | 1 | 0 | a’·b·c’ | |
| 0 | 1 | 1 | 0 | 1 | | a’·b·c |
| 1 | 0 | 0 | 1 | 0 | a·b’·c’ | |
| 1 | 0 | 1 | 0 | 1 | | a·b’·c |
| 1 | 1 | 0 | 0 | 1 | | a·b·c’ |
| 1 | 1 | 1 | 1 | 1 | a·b·c | a·b·c |

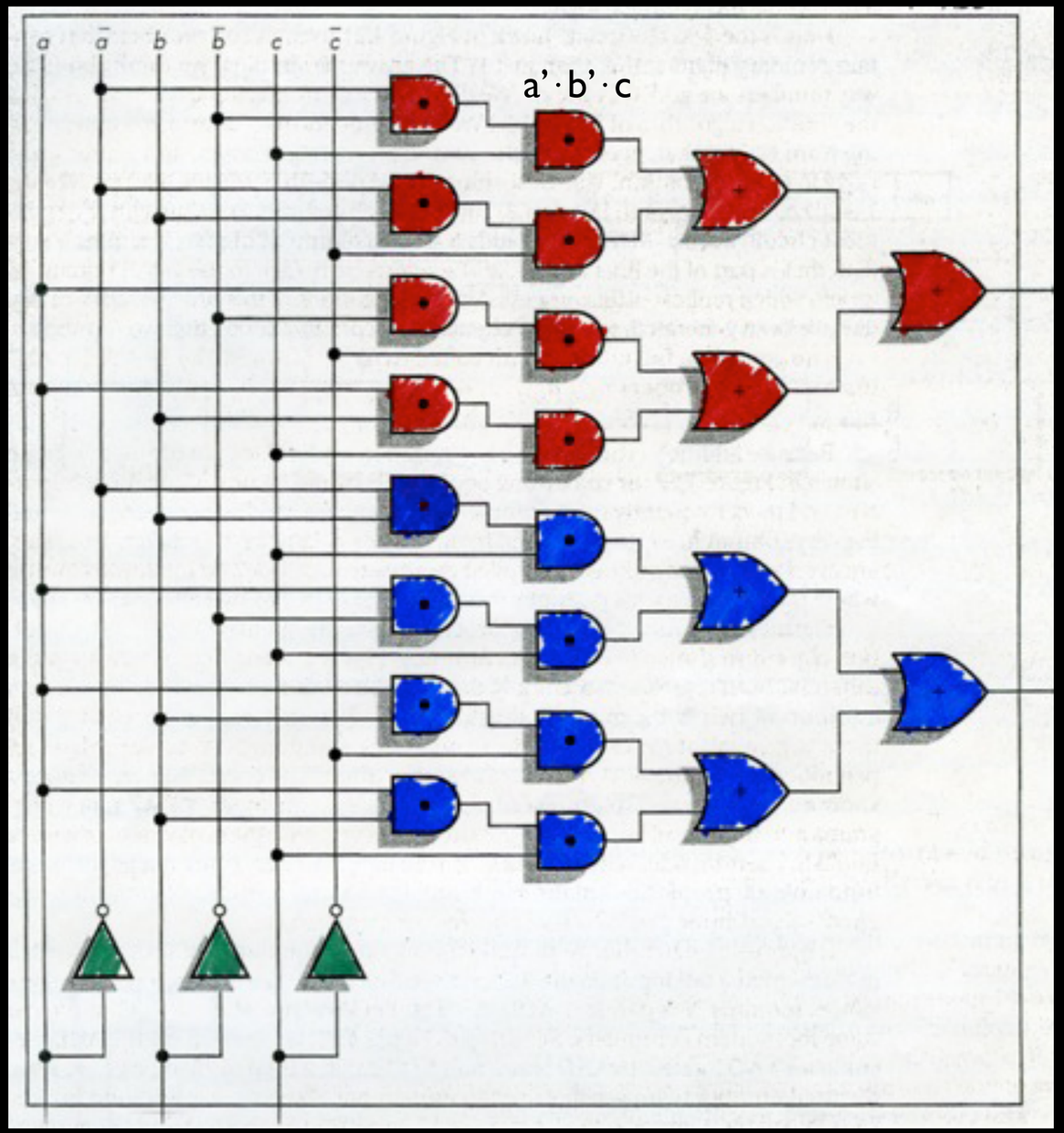
d = (a’·b’·c) + (a’·b·c’) + (a·b’·c’) + (a·b·c)

# 1-Bit Adder

| a | b | c | d | e | sub-expressions (d) | sub-expressions (e) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 1 | 1 | 0 | a’·b’·c | |
| 0 | 1 | 0 | 1 | 0 | a’·b·c’ | |
| 0 | 1 | 1 | 0 | 1 | | a’·b·c |
| 1 | 0 | 0 | 1 | 0 | a·b’·c’ | |
| 1 | 0 | 1 | 0 | 1 | | a·b’·c |
| 1 | 1 | 0 | 0 | 1 | | a·b·c’ |
| 1 | 1 | 1 | 1 | 1 | a·b·c | a·b·c |

e = (a’·b·c) + (a·b’·c) + (a·b·c’) + (a·b·c)

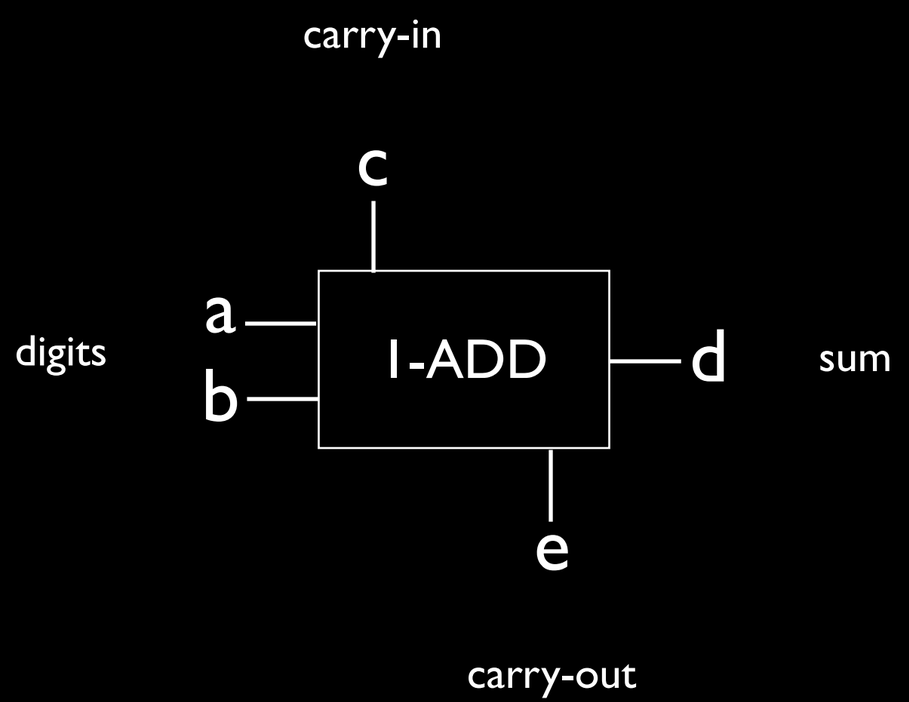
# 1-Bit Adder



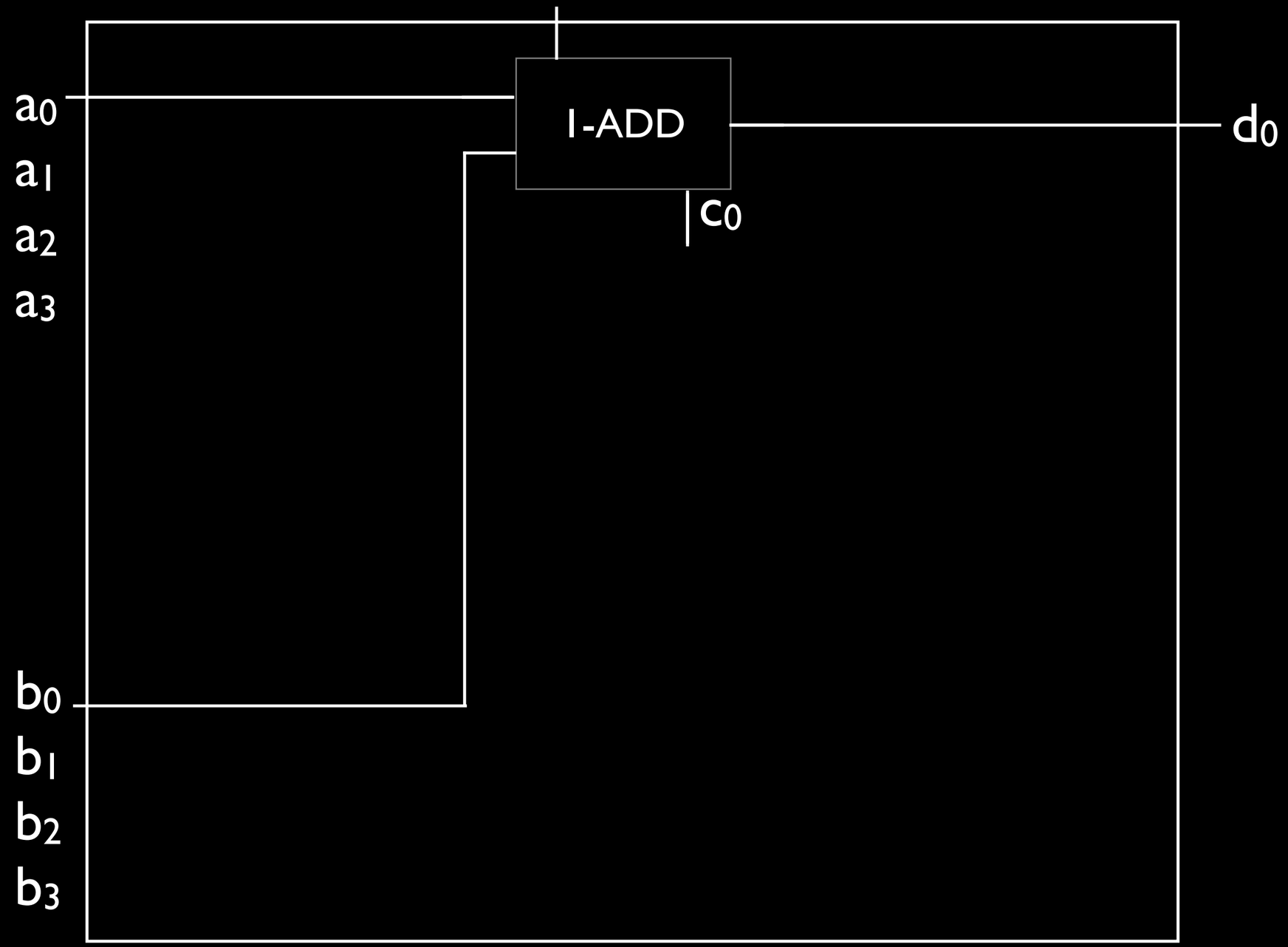
# 4-Bit Adder

build a circuit that adds two 4-bit numbers

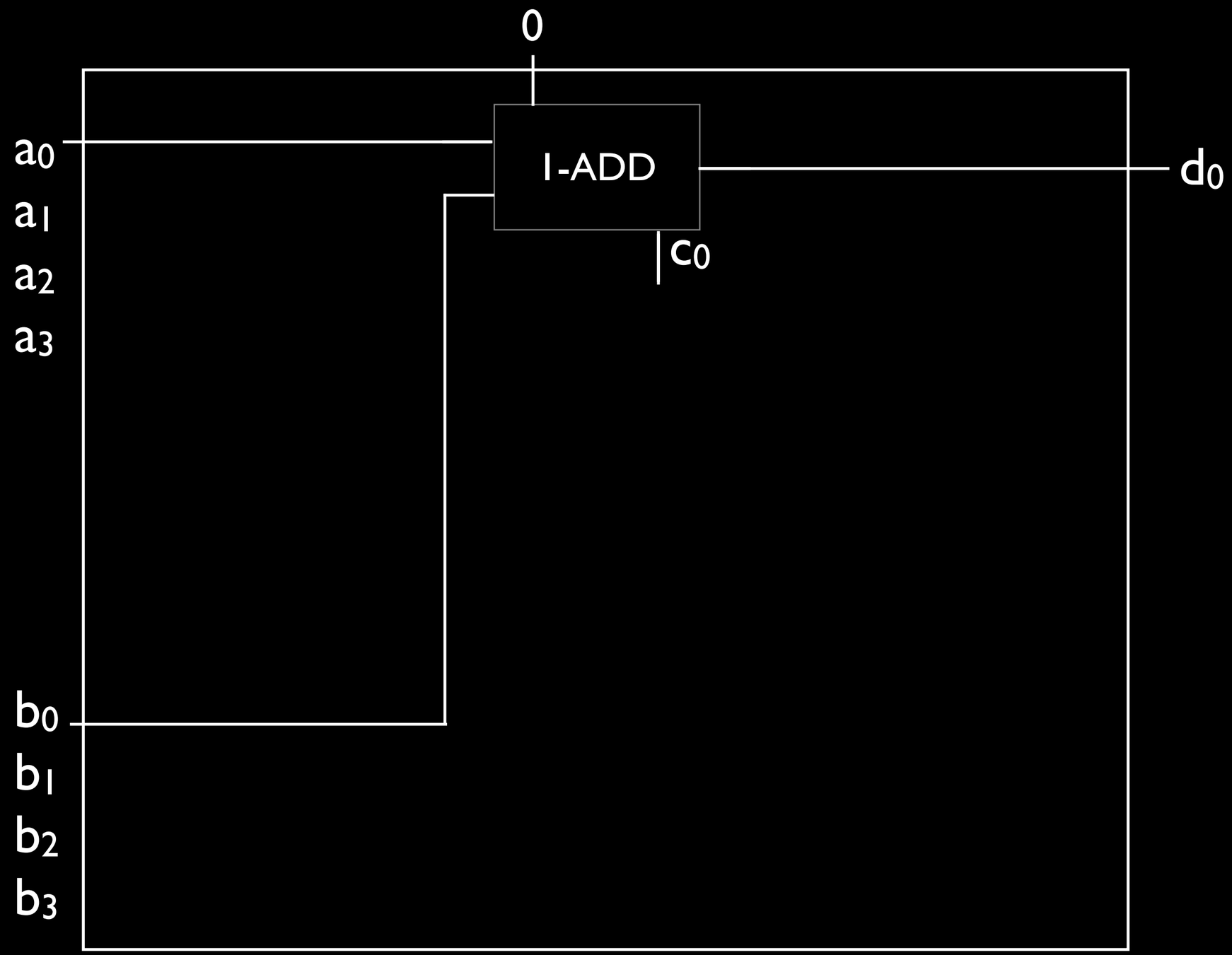
# 4-Bit Adder



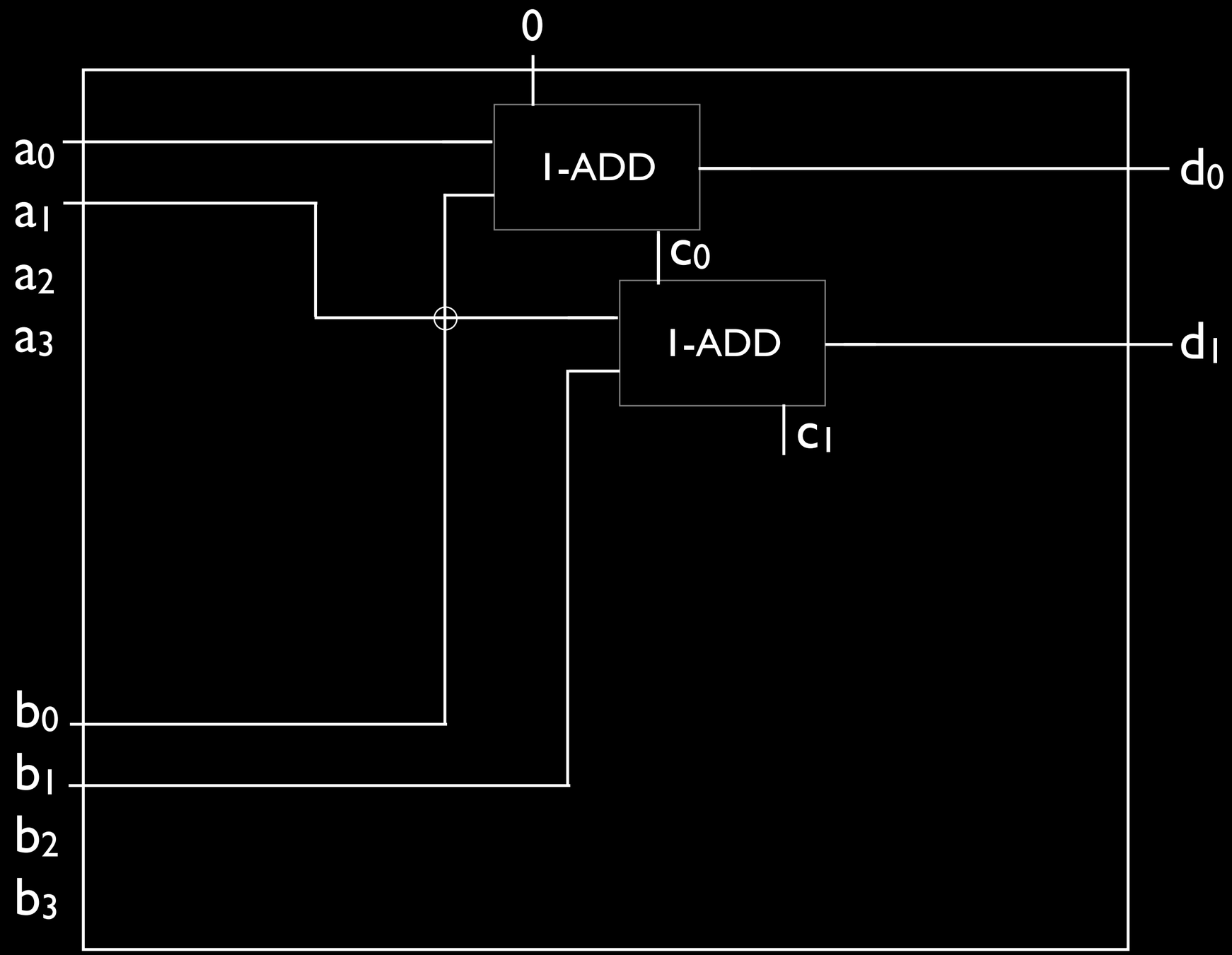
# 4-Bit Adder



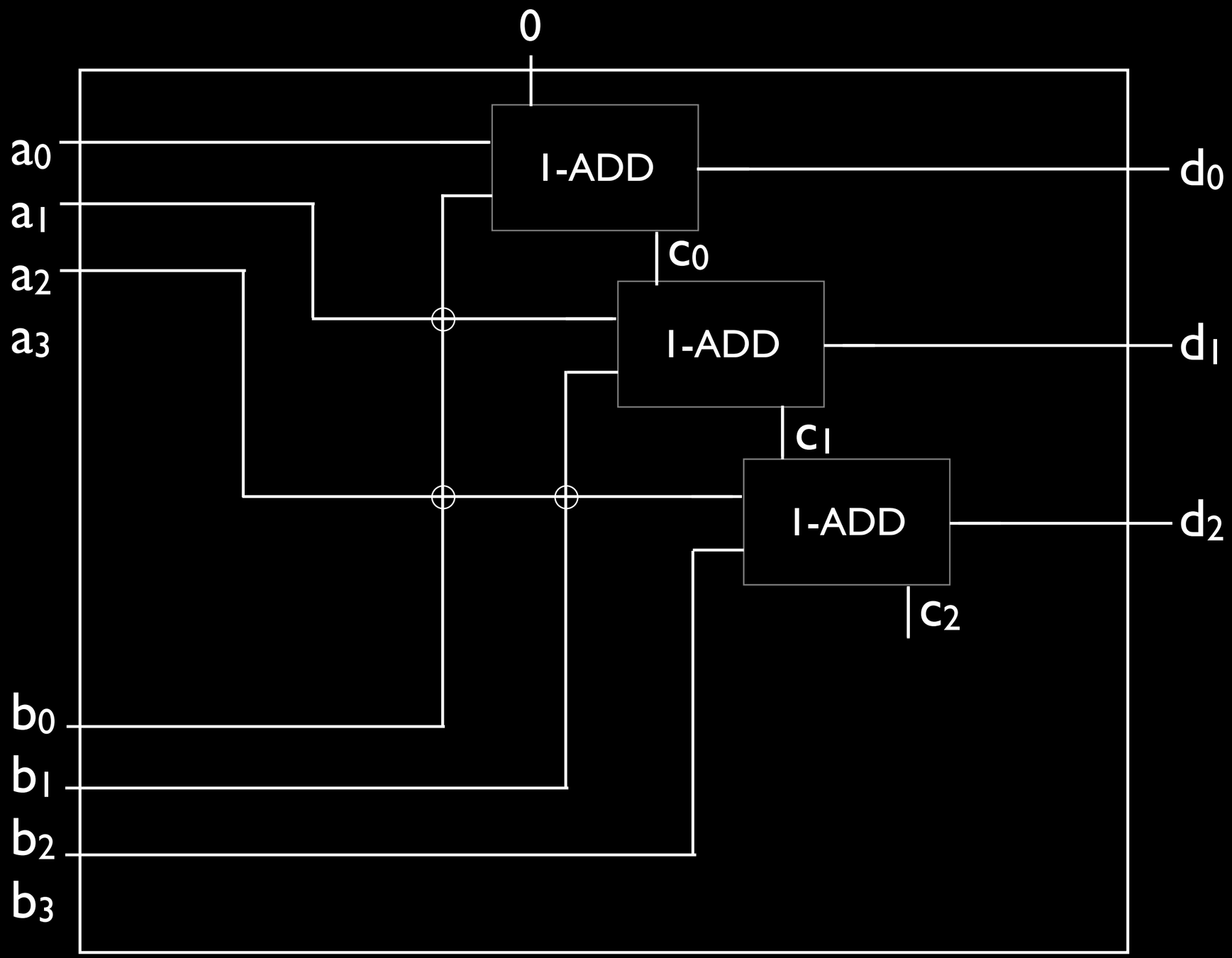
# 4-Bit Adder



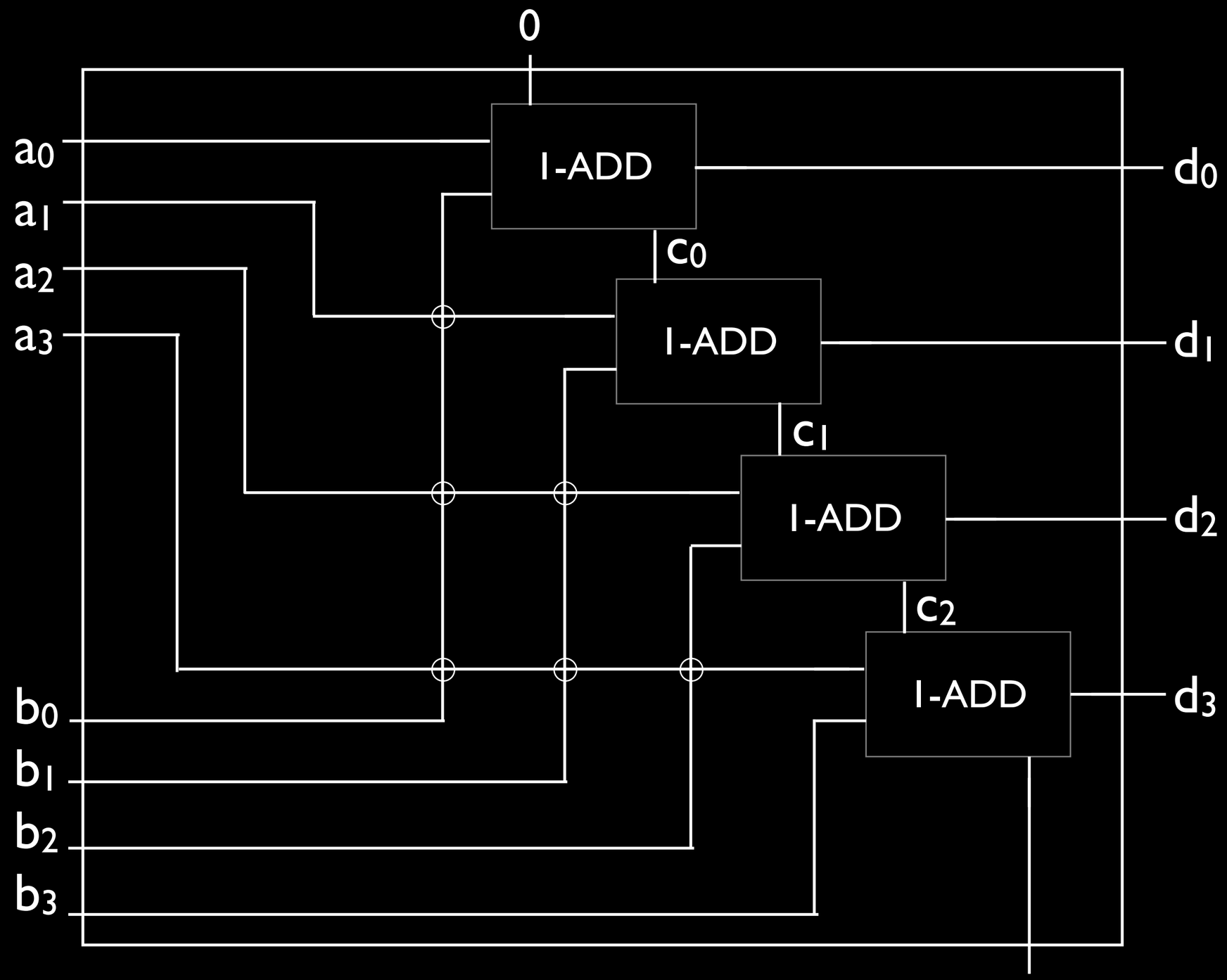
# 4-Bit Adder



# 4-Bit Adder



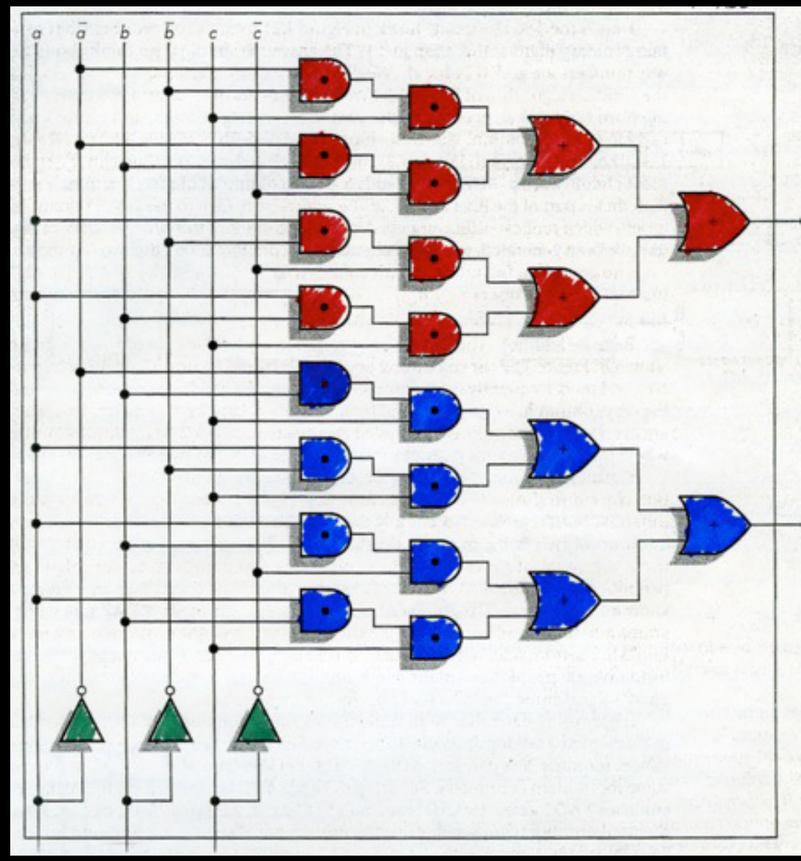
# 4-Bit Adder



# 32-bit Adder

96 = 32x3 *not* gates
512 = 32x16 *and* gates
192 = 32x6 *or* gates

800 gates

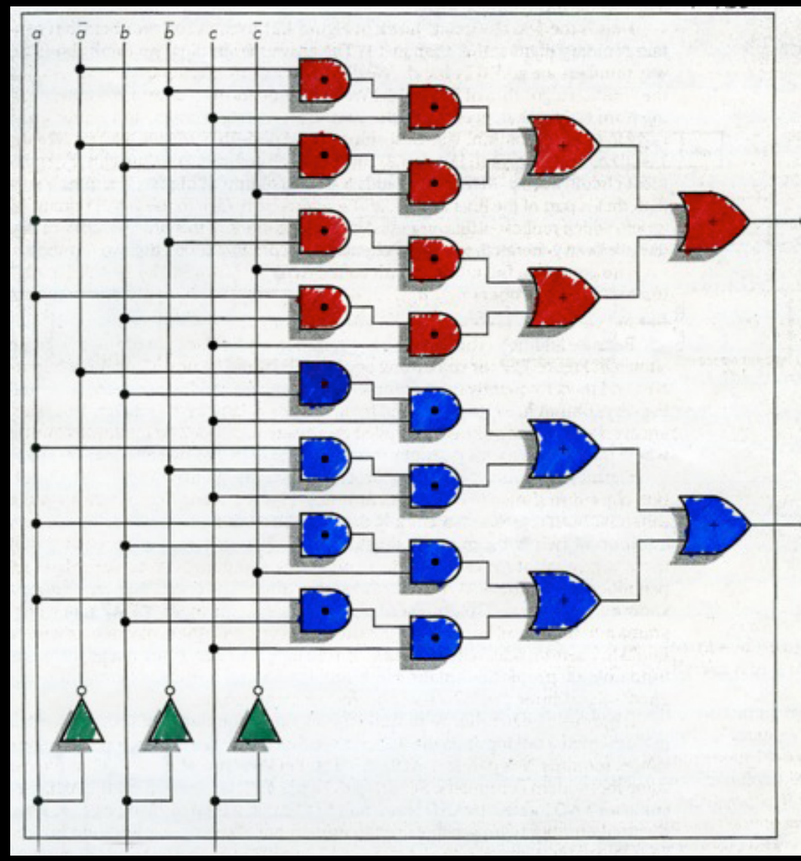


# 32-bit Adder

96 = 32x3 *not* gates
512 = 32x16 *and* gates
192 = 32x6 *or* gates

800 gates

1,504 = 96+1024+384 transistors



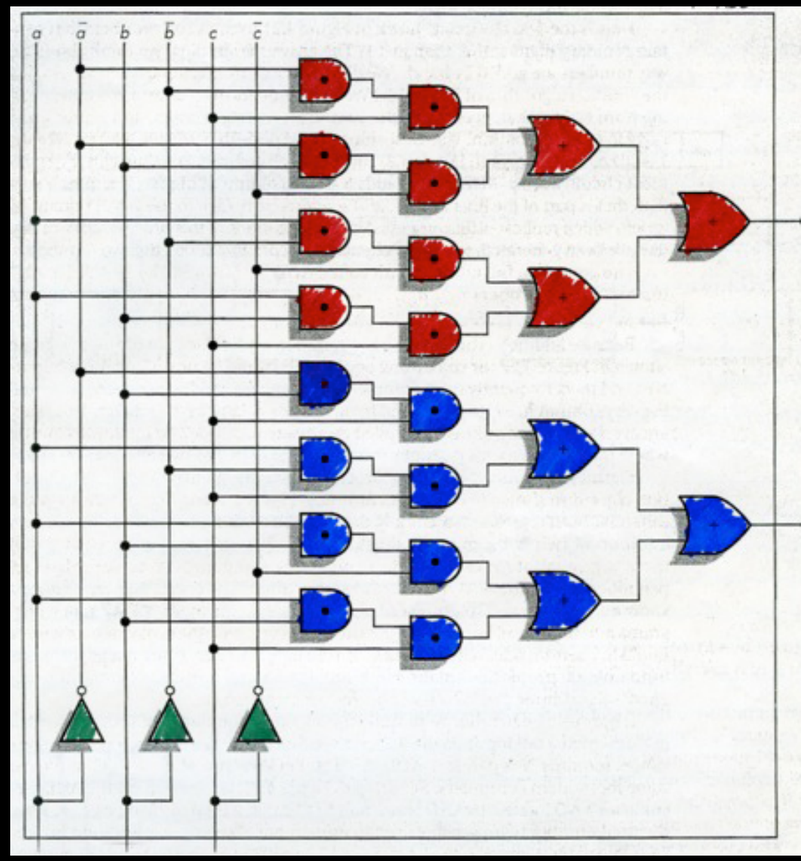
# 32-bit Adder

96 = 32x3 *not* gates
512 = 32x16 *and* gates
192 = 32x6 *or* gates

800 gates

1,504 = 96+1024+384 transistors

1945: refrigerator-sized computer
2015: .-sized computer



# Equals 0

build a circuit that determines if an 8-bit number is 0

Equals 0

$$a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0 = 00000000$$

# Equals 0

output


How many rows?

# Equals 0

| a | b | c | d | e | f | g | h | i |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

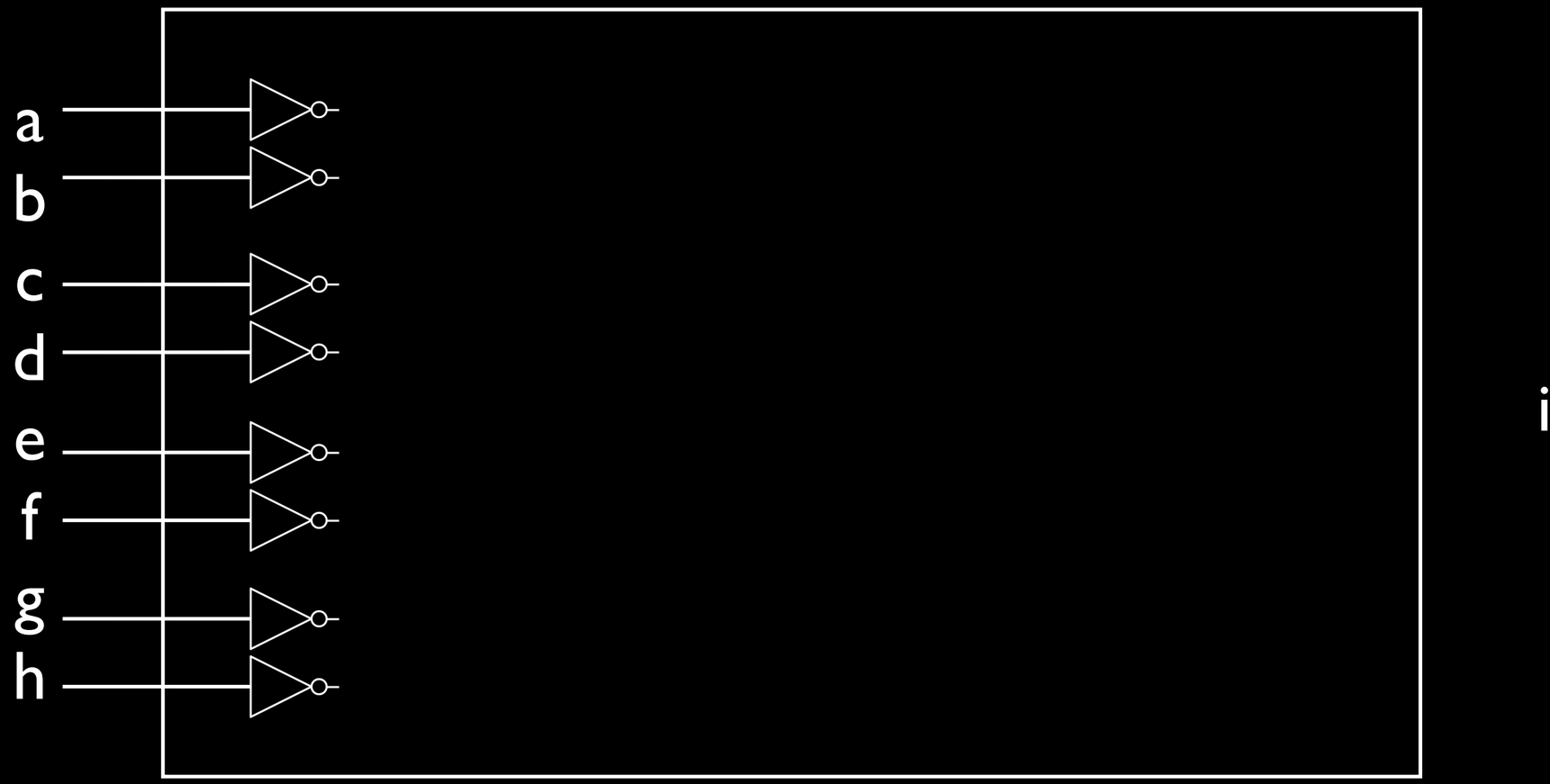
Only one row has an output of 1

# Equals 0

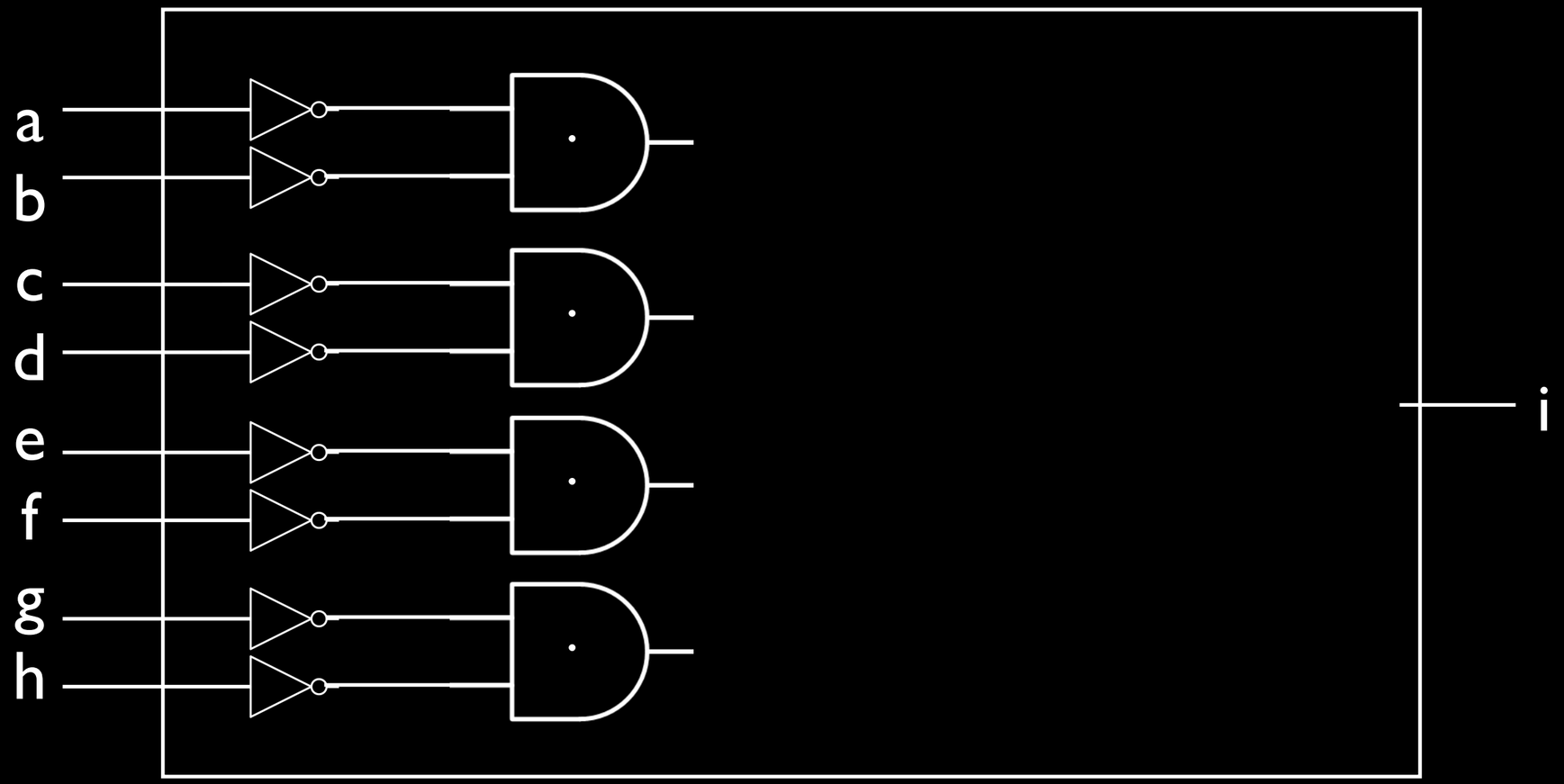
| a | b | c | d | e | f | g | h | i |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

$$i = a' \cdot b' \cdot c' \cdot d' \cdot e' \cdot f' \cdot g' \cdot h'$$

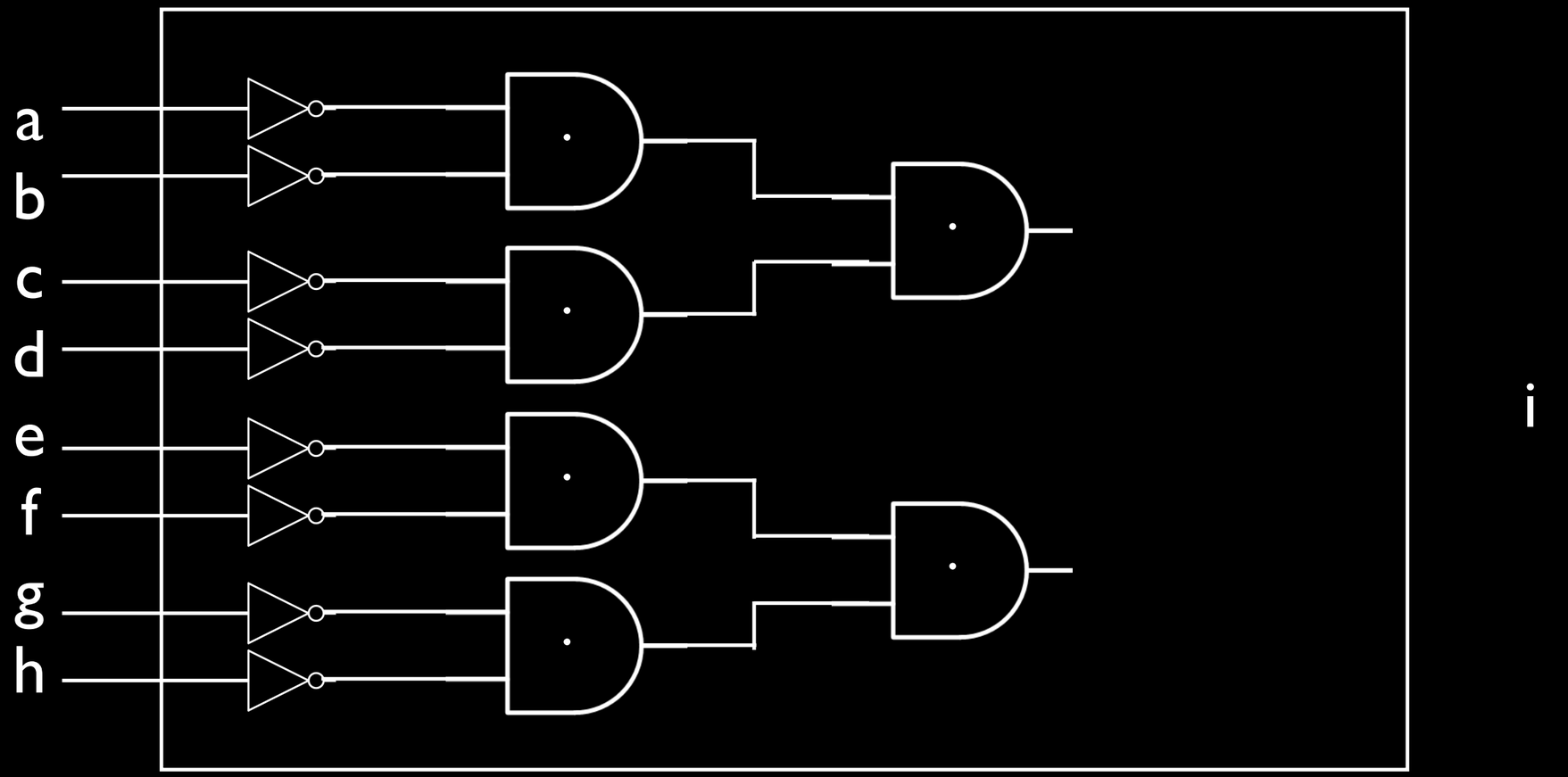
Equals 0



Equals 0



Equals 0



Equals 0