COMBINATIONAL CIRCUITS

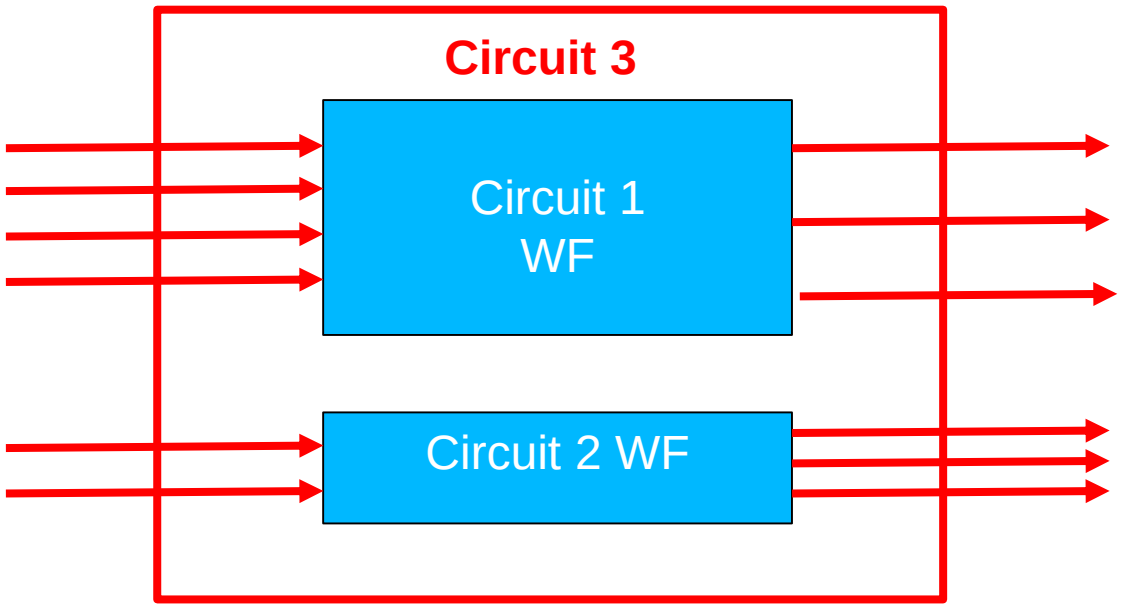
- Principles
- Characteristics

WELL FORMED CIRCUITS: RULES 1 AND 2

DEFINITION: a well formed (WF) circuit is a circuit built only using the following 4 rules:

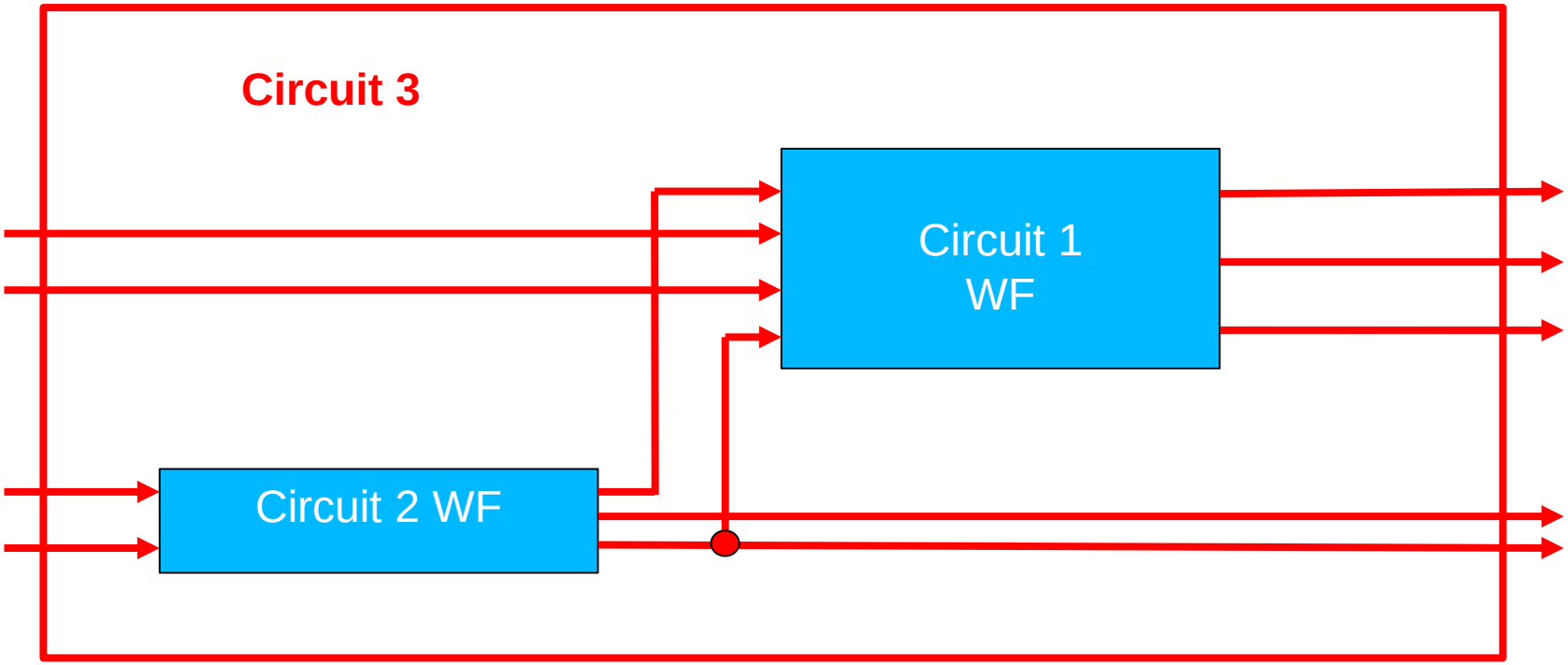
RULE 1: simple gates (OR, AND, NOT, NOR, NAND, XOR, XNOR) and wires are WF circuits

RULE 2: concatenation of two WF circuits is a WF circuit.



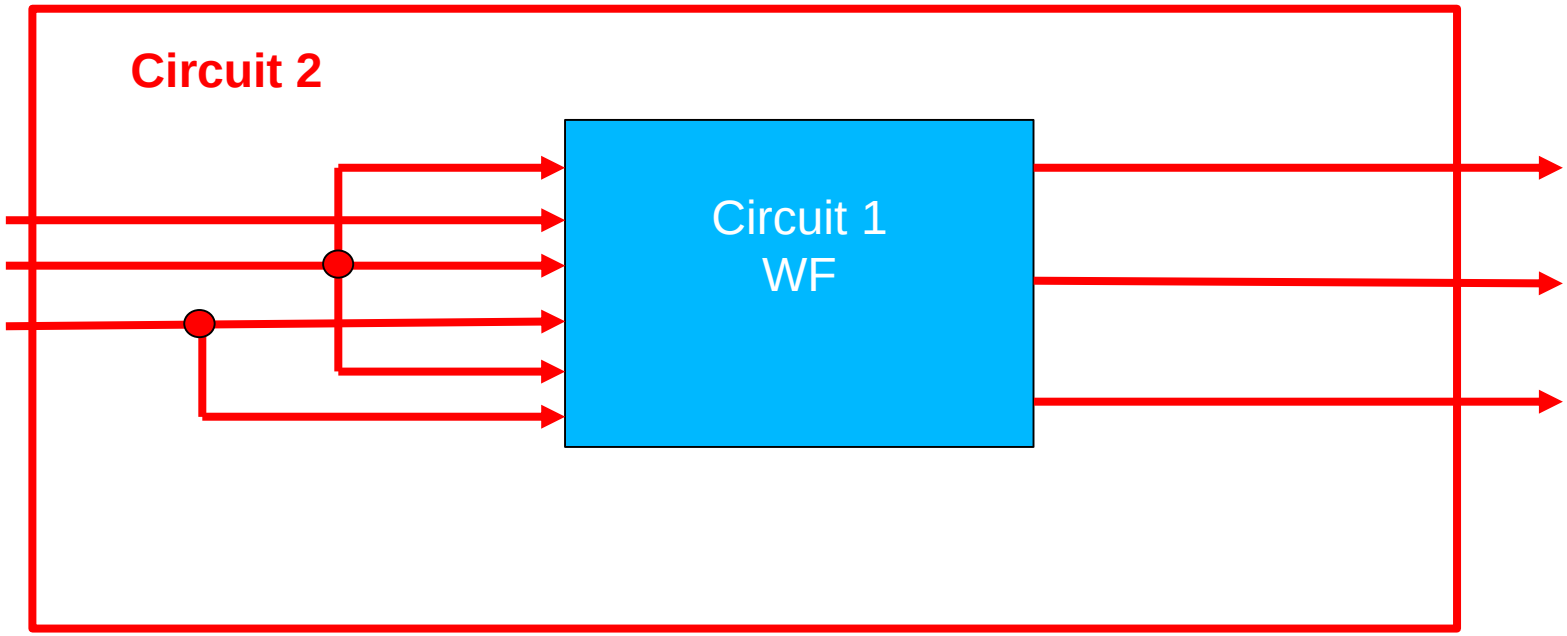
WELL FORMED CIRCUITS: RULE 3 (2)

RULE 3: a circuit obtained by connecting some inputs of a WF circuit to the outputs of a WF circuit is in turn WF



WELL FORMED CIRCUITS: RULE 4 (2)

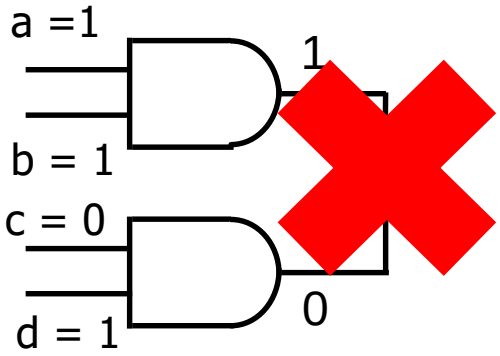
RULE 4: a circuit obtained by connecting together some inputs of a WF circuit is in turn WF



FORBIDDEN CONNECTIONS (2)

The connection below is strictly forbidden.

Potentially the output of the two gates can be different. If for example a = b = 1 , the first gate will deliver a logical 1 and a high voltage while the second gate will deliver a logical 0 and low voltage.



The same wire will carry two different logical values: NO MEANING

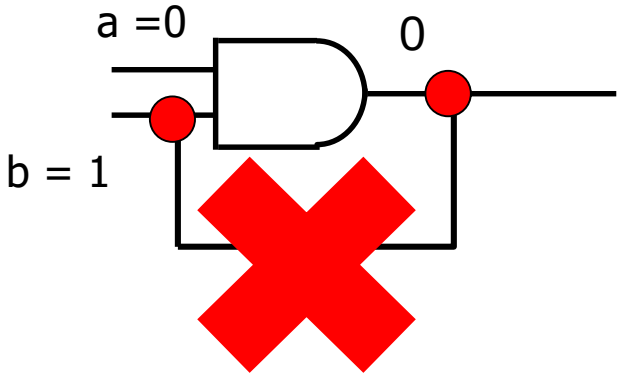
The same wire will create a short circuit.

A WF circuit cannot have have such connections.

FORBIDDEN CONNECTIONS (3)

The connection below is strictly forbidden.

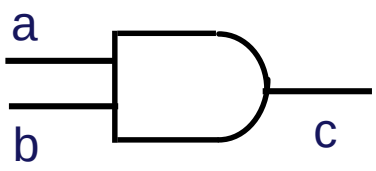
Potentially the output of the two gates can be different. If for example a = 0 and b = 1 , the gate will deliver a logical 0 and a low voltage while it is connected to the b input high voltage and logical 1!!



The same wire will carry two different logical values: NO MEANING
The same wire will create a short circuit.

A WF circuit cannot have have such connections.

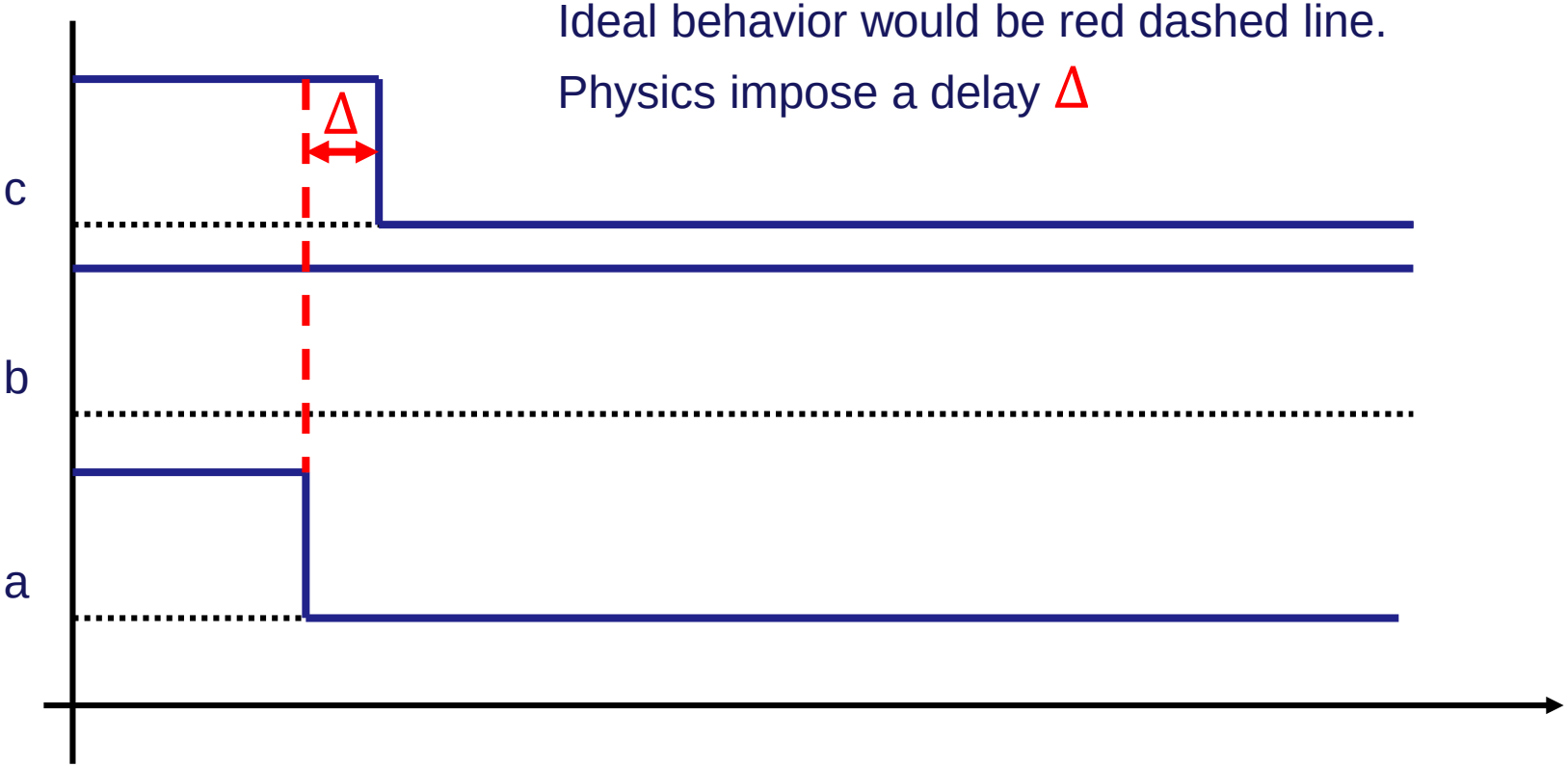
GATE DELAY (1)



AND gate

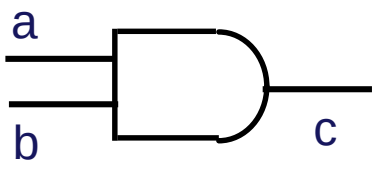
The output value c takes some time before taking into account input value change!!
Ideal behavior would be red dashed line.

Physics impose a delay Δ

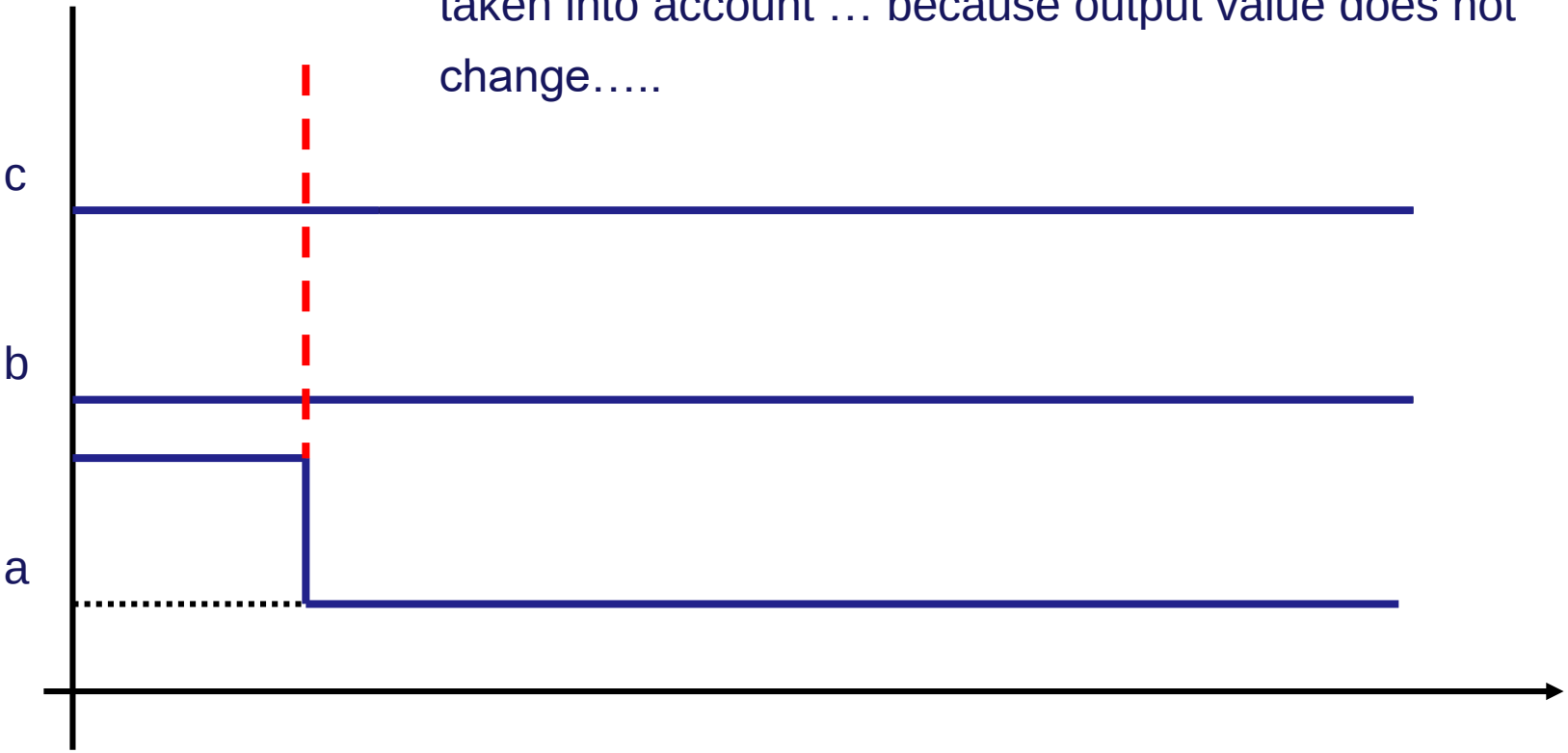


GATE DELAY (2)

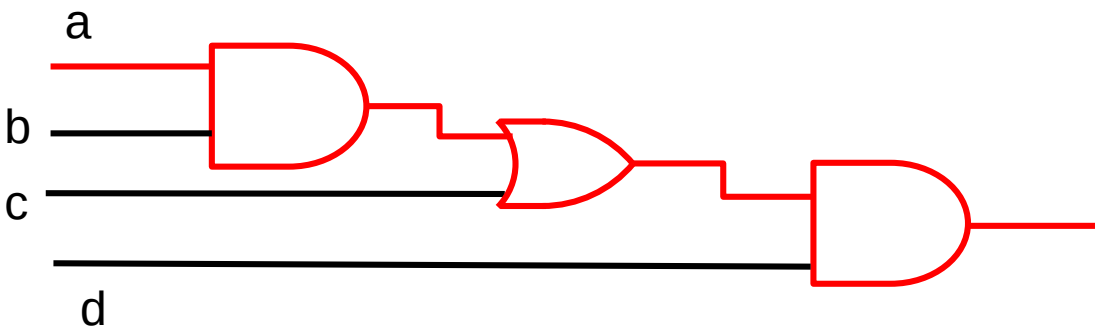
AND gate



Gate delay depends upon input.
For an AND gate, if one of the input value is 0, changing the other input value wil be instantaneously taken into account … because output value does not change…..

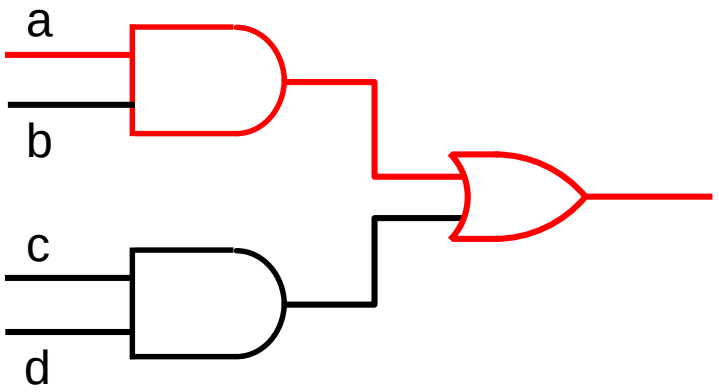


VARYING DEPTH



DEPTH 3

In red, one critical path is displayed.
THERE IS NOT A UNIQUE CRITICAL PATH

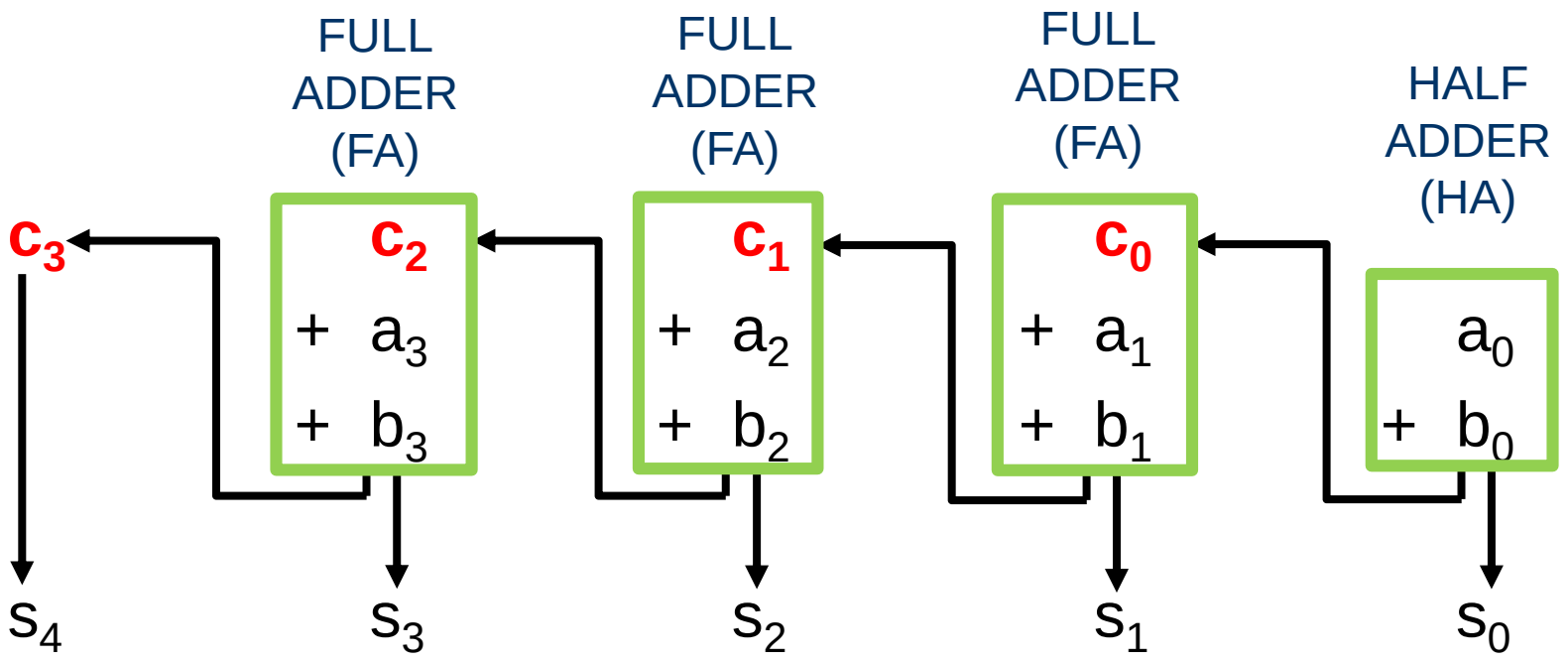


DEPTH 2

Adding two 4 bits numbers (a) and (b), the result (s) on 5 bits

$(a_3a_2a_1a_0) + (b_3b_2b_1b_0) = (s_4s_3s_2s_1s_0)$

Bitwise algorithm view



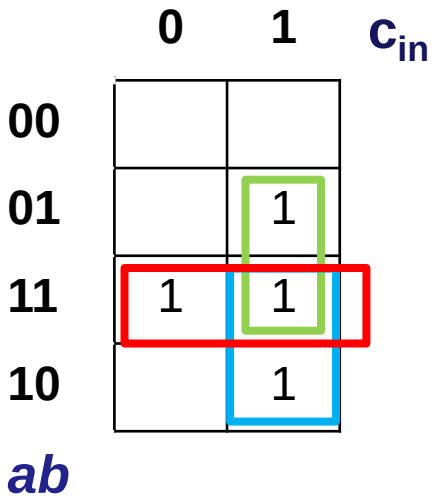
FULL ADDER

A Full Adder sums 3 bits and delivers a result on 2 bits: $c_{in} + a + b = (c_{out}s)$ where c_{in} denotes the incoming carry while c_{out} denotes the outgoing carry. Truth table below:

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

COMPUTING c_{out}

c_{in}	a	b	c_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

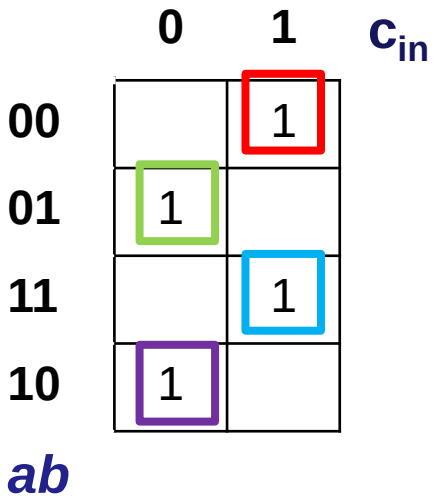


KMAP for c_{out}
Minimal SOP
$c_{out} = ab + bc_{in} + ac_{in}$

Canonical Form
$c_{out} = abc_{in}' + a'b\,c_{in} + ab'c_{in} + ab\,c_{in}$

COMPUTING s (1)

c_{in}	a	b	s
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



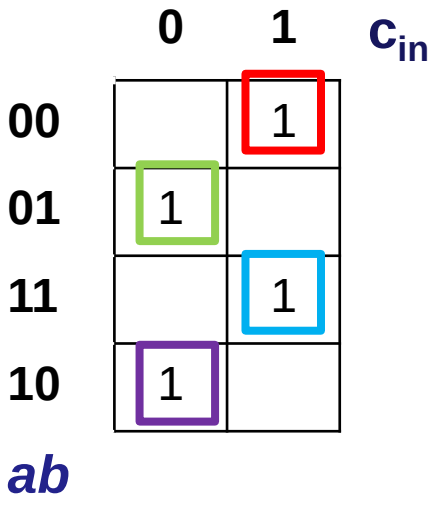
KMAP for s
Minimal SOP = Canonical Form

Canonical Form

$s = a'bc_{in}' + ab'c_{in}' + a'b' c_{in} + ab\ c_{in}$

COMPUTING s (2)

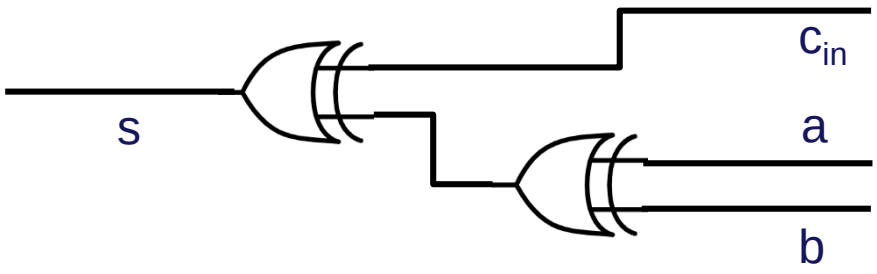
# 1	c_{in}	a	b	s
0	0	0	0	0
1	0	0	1	1
1	0	1	0	1
2	0	1	1	0
1	1	0	0	1
2	1	0	1	0
2	1	1	0	0
3	1	1	1	1



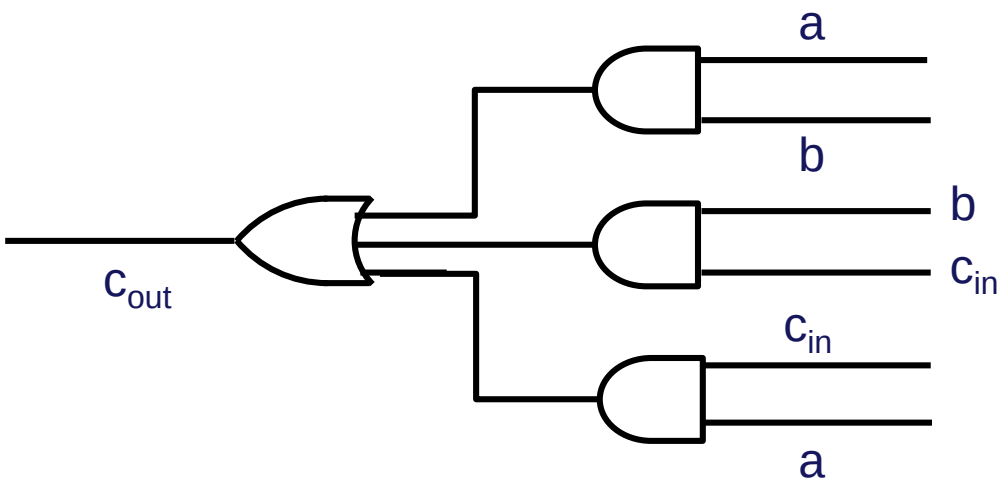
$s = a \oplus b \oplus c_{in}$

Much simpler

CIRCUITS FOR COMPUTING s AND c_{out}



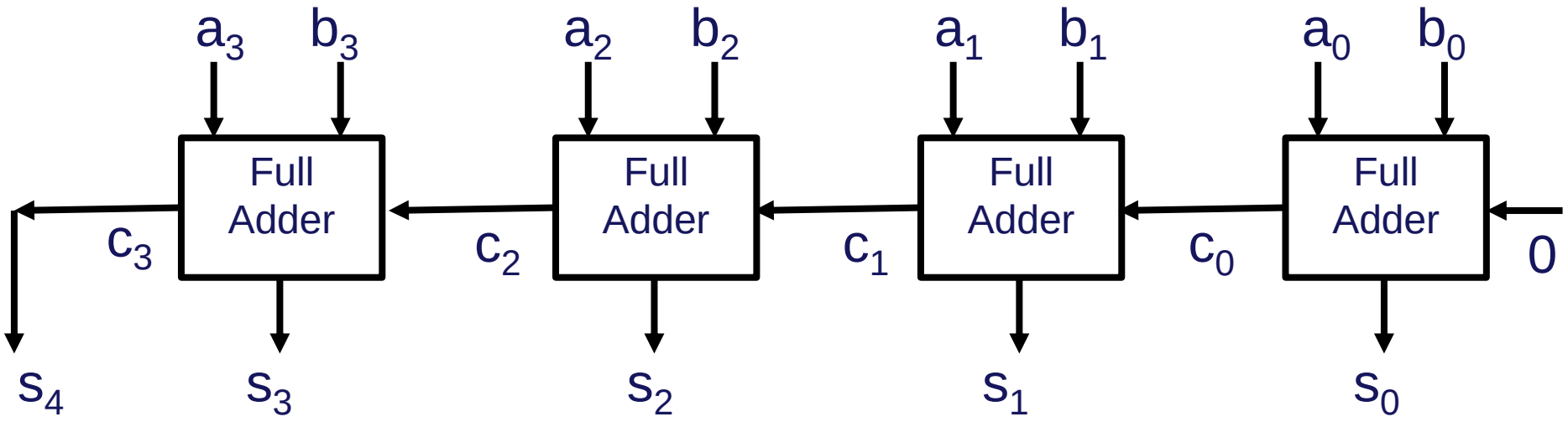
2 XOR gates
Total delay: 2 Δ



3 AND gates + 1 OR Gate
Total delay: 2 Δ

ASSUMPTION: Let us assume that all gates (AND, OR, XOR) have the same delay Δ

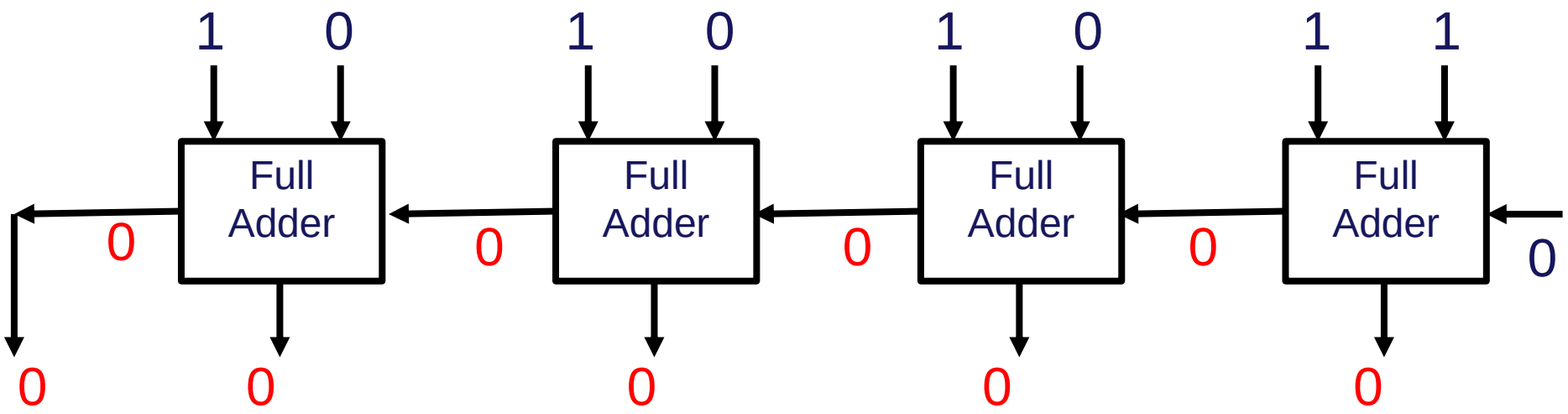
BUILDING THE RIPPLE CARRY ADDER



Ripple Carry Adder (RCA): the carries are propagated from the rightmost bit to the leftmost one.

REMARK: the first Full Adder could be replaced by a simpler structure: Half Adder

RIPPLE CARRY ADDER TIMING (1)

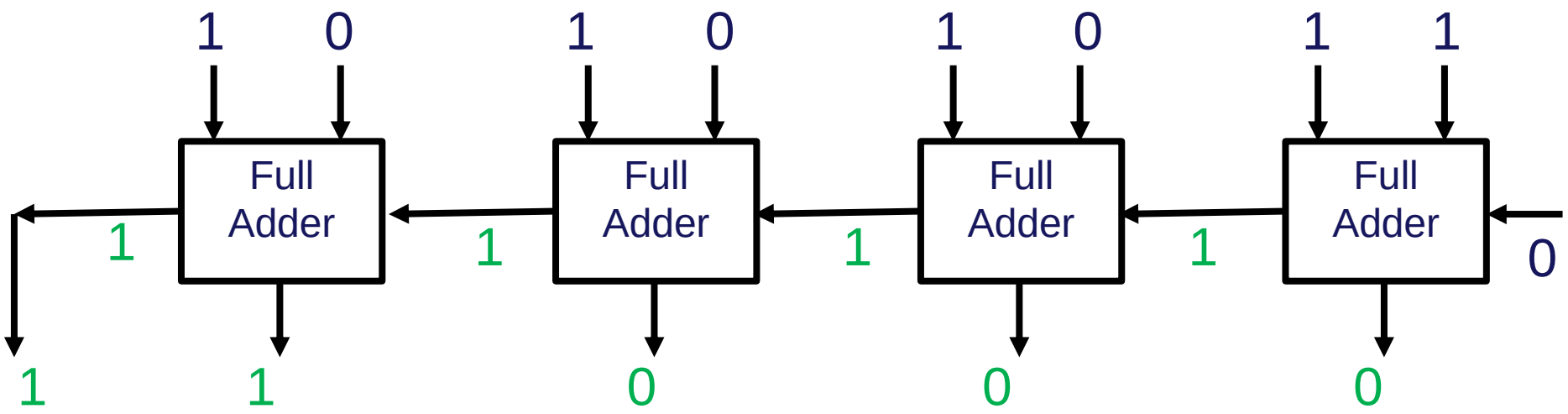


TIME T = 0

1	1	1	1	
	1	1	1	1
	0	0	0	1
1	0	0	0	0

Nothing is computed only the inputs are valid!!
In blue the inputs
In red the invalid outputs
In green the valid outputs

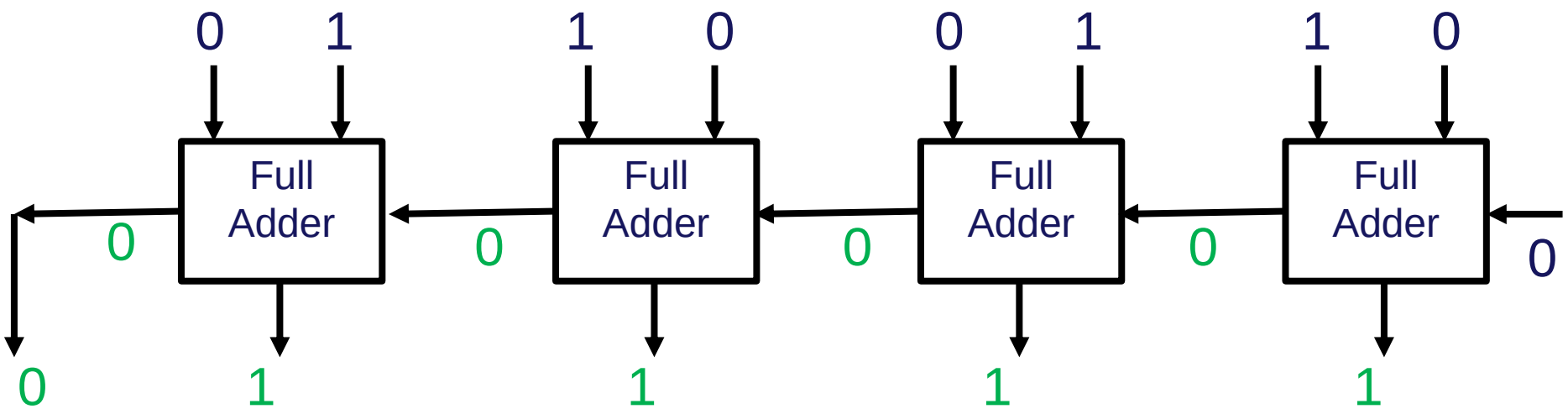
RIPPLE CARRY ADDER TIMING (5)



TIME T = 8 Δ

In blue the inputs
In red the invalid outputs
In green the valid outputs

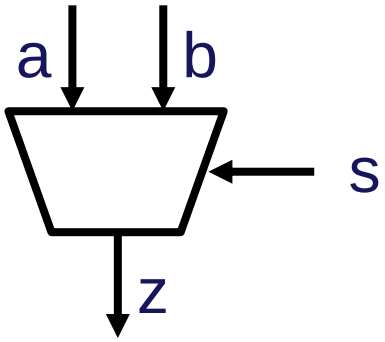
RIPPLE CARRY ADDER TIMING (1)



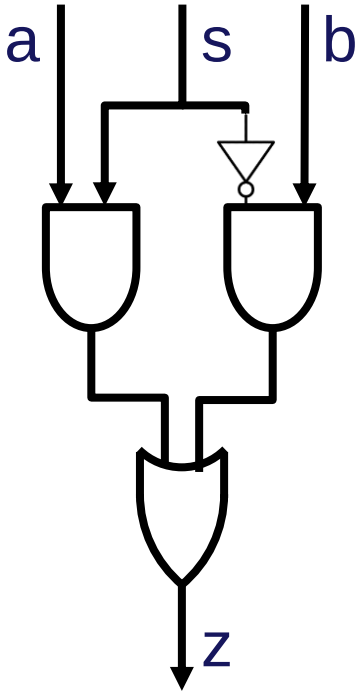
TIME T = 2 Δ

All of the outputs are valid
In blue the inputs
In red the invalid outputs
In green the valid outputs

A SIMPLE 2x1 MULTIPLEXER (MUX)

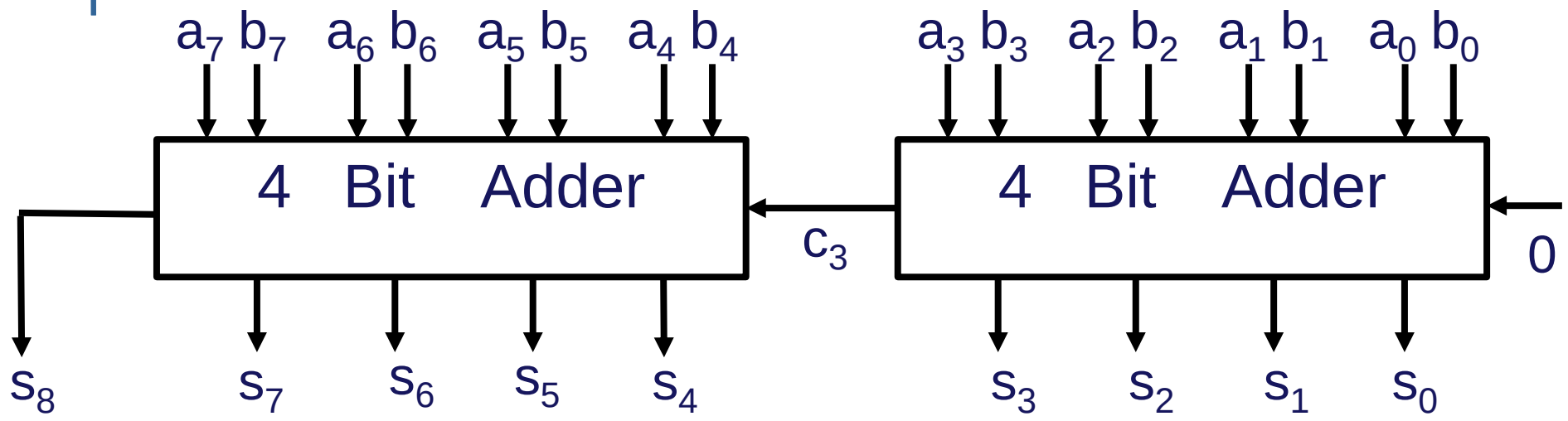


If s = 0 then z = a
If s = 1 then z = b



z = s' a + s b
2 AND gates + 1 OR Gate + 1 NOT Gate
Delay: 2 Δ (NOT gate effect being neglected)

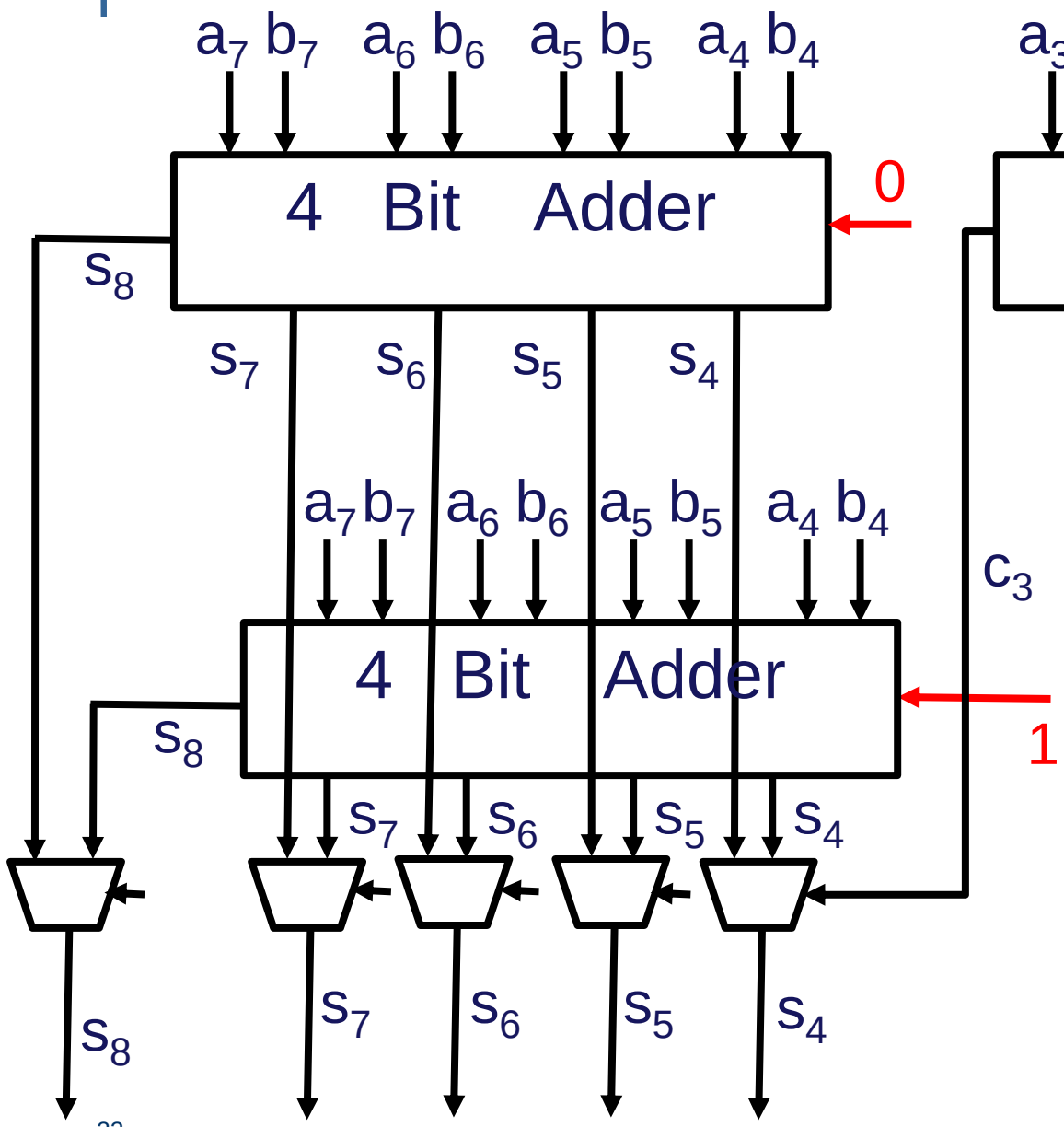
AN 8 BIT RCA ADDER

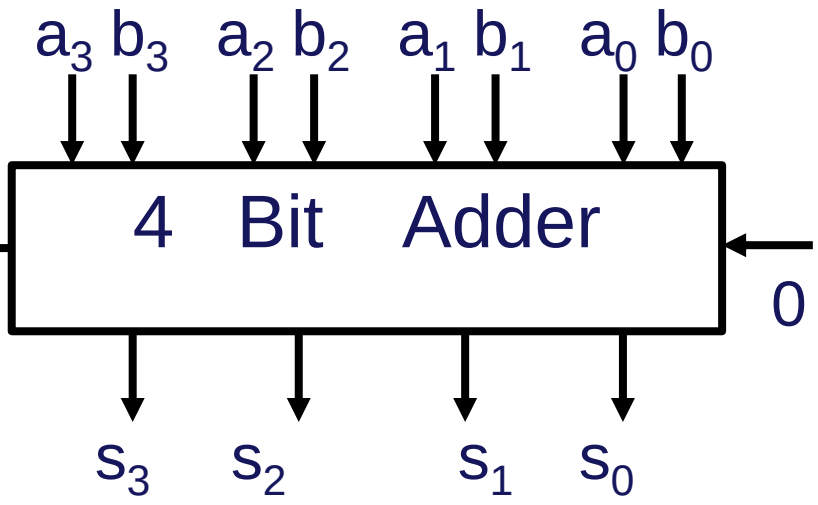


Let us build a 8 bit RCA adder using a 4 Bit RCA Adder (see above).

Stab (RCA(8)) = 16 Δ

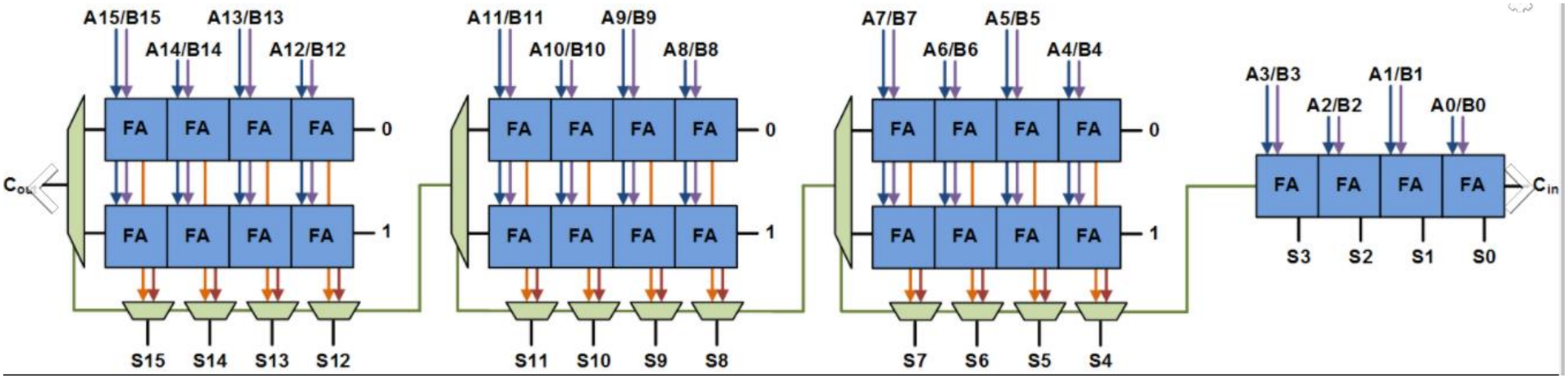
AN 8 BIT ADDER: CARRY SELECT ADDER





The three adders are computing in parallel. After 8 Δ, the 3 adders have coorect outputs. Then we just nedd to go through MUX (in parallel), 2 Δ

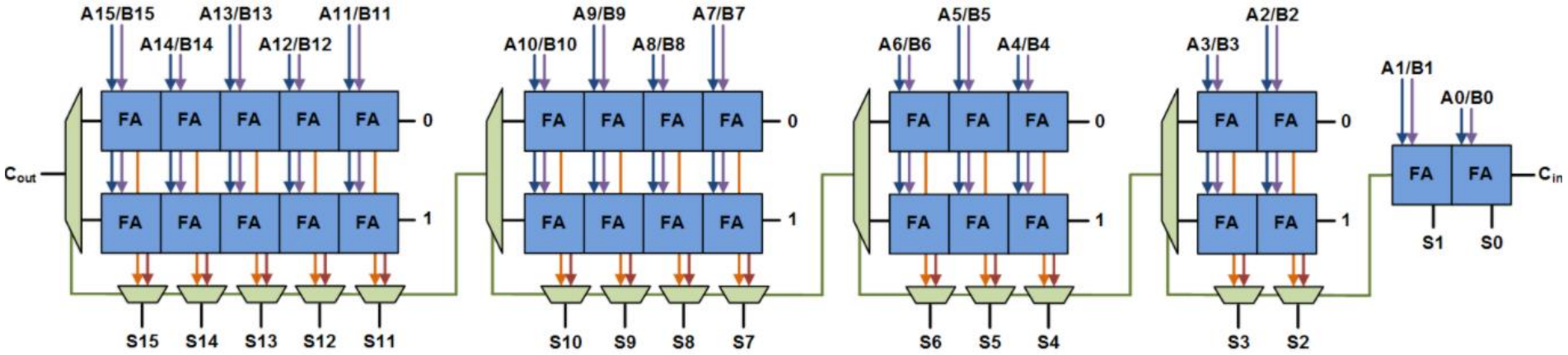
CARRY SELECT ADDER VARIANTS (1)



Using constant size building blocks: a 4 x 4 Adder

Picture from Wikipedia

CARRY SELECT ADDER VARIANTS (2)



Using variable size building blocks: a 5-4-3-2-2 Adder

Picture from Wikipedia

Acknowledgements

A lot of material was found in Wikipedia.

Some of these slides were inspired by slides developed by:

- University of Washington Computer Science & Engineering (CSE 370)
- Y.N. Patt (Univ of Texas Austin)
- S. J. Patel (Univ of Illinois Urbana Champaign)
- Walid A. Najjar (Univ California Riverside)
- Brian Linard (Univ California Riverside)
- G.T. Byrd (Univ North Carolina)