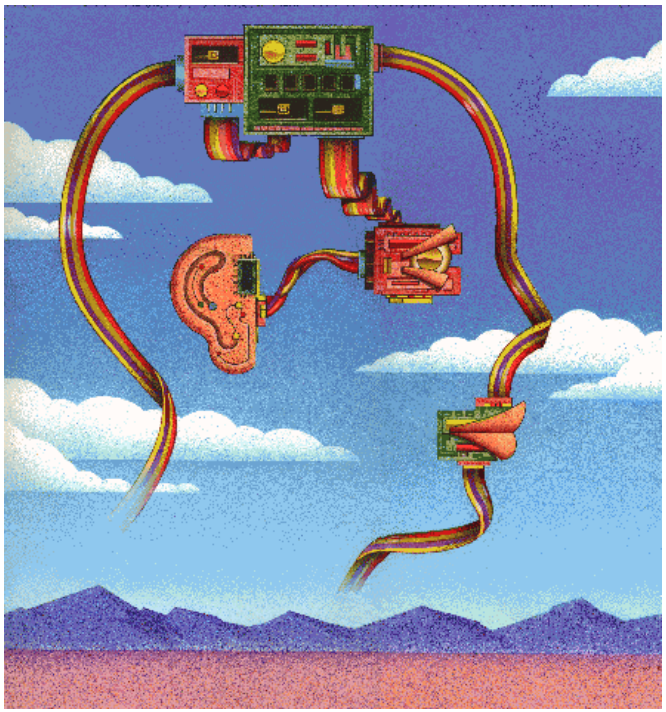
Arithmetic CMOS circuits and Introduction to High Level Synthesis



Andreas G. Andreou
Pedro Julian

Electrical and Computer Engineering
Johns Hopkins University

http://andreoulab.net

Levels of Abstraction –CMOS 1 bit adder-

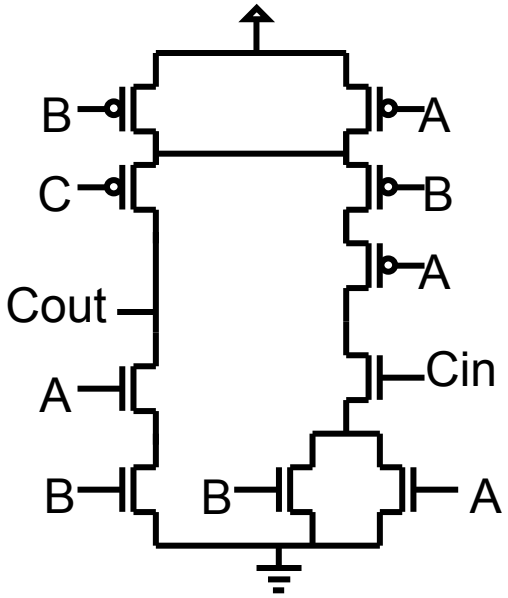
Sum = A XOR B XOR Cin

Cout = (A AND B) OR (B AND Cin)
OR (A AND Cin)

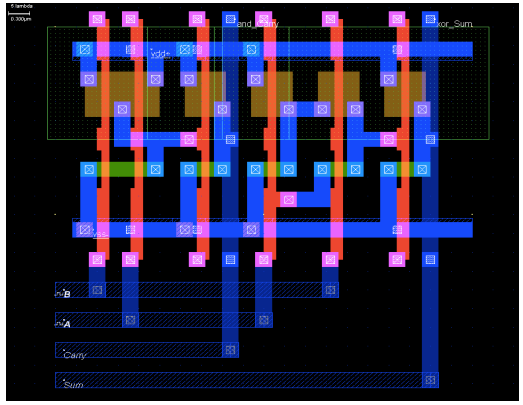
Equation

C_{in}	A	B	Sum	Cout
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

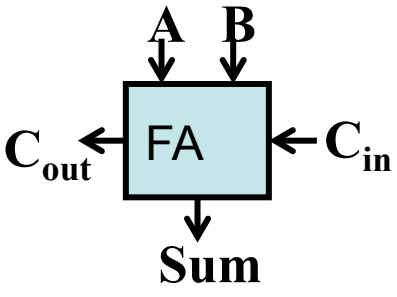
Truth Table



Circuit



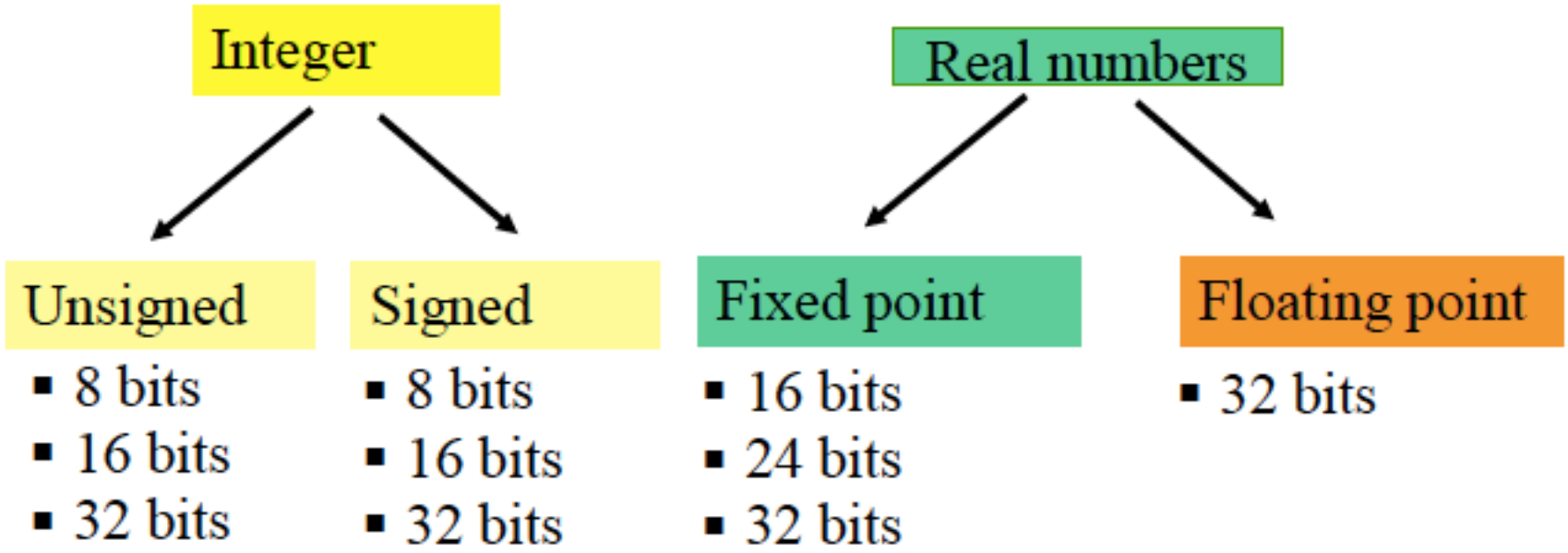
Layout



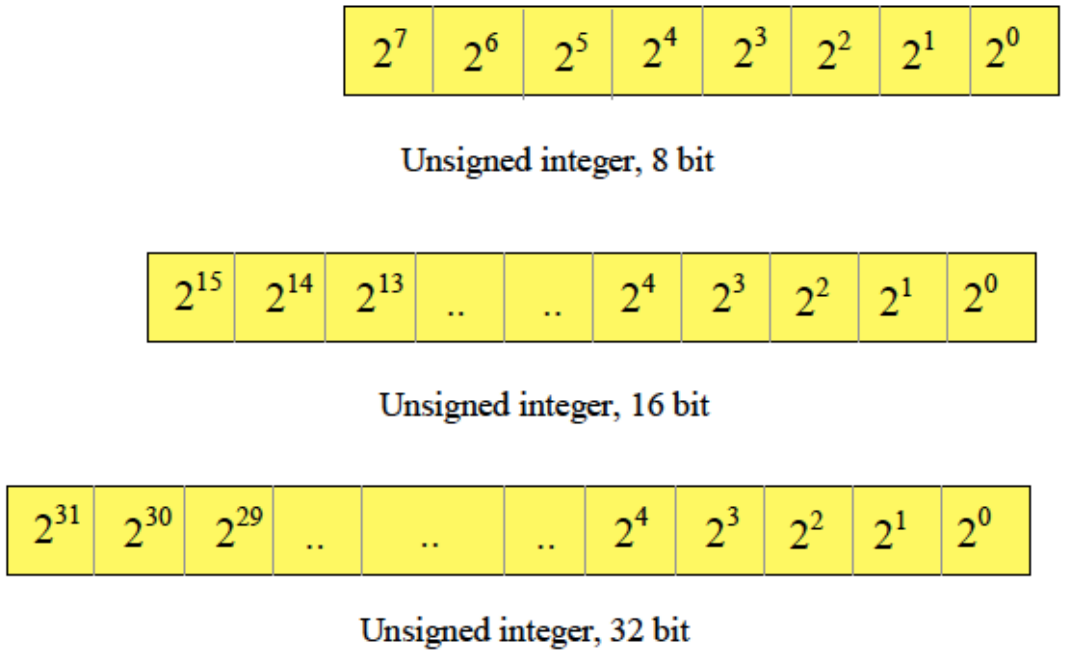
Symbol

LOGICAL PHYSICAL

Numbers in computers



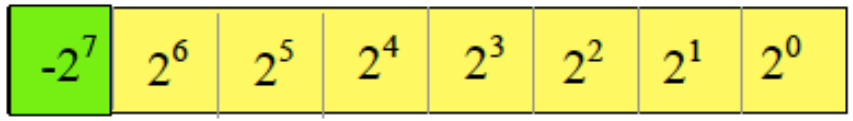
Unsigned integers



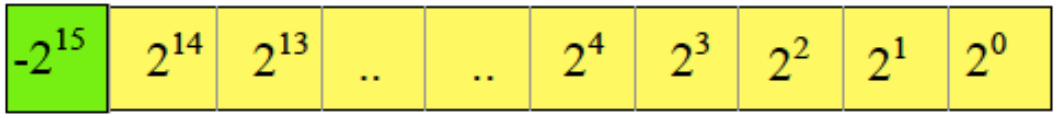
$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
....
$2^{10} = 1024$
$2^{15} = 32768$
$2^{20} = 1048576$
$2^{30} = 1073741824$
$2^{31} = 2147483648$

$$01101011 = 2^6 + 2^5 + 2^3 + 2^1 + 2^0 = 64 + 32 + 8 + 2 + 1 = 107$$

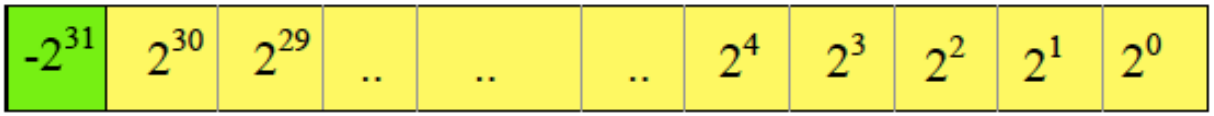
Signed integers



Signed integer, 8 bit format



Signed integer, 16 bit format

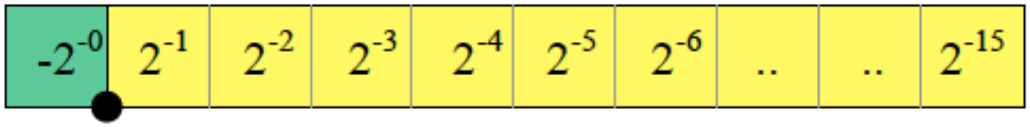


Signed integer, 32 bit format

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
....
$2^{10} = 1024$
$2^{15} = 32768$
$2^{20} = 1048576$
$2^{30} = 1073741824$
$2^{31} = 2147483648$

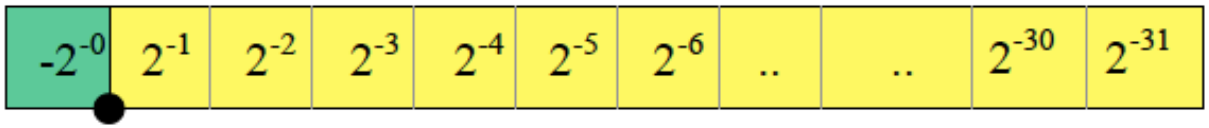
$$11101011 = -2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0 = -128 + 64 + 32 + 8 + 2 + 1 = -21$$

Real numbers: Fixed point



Fixed point, 16 bit

Fixed point



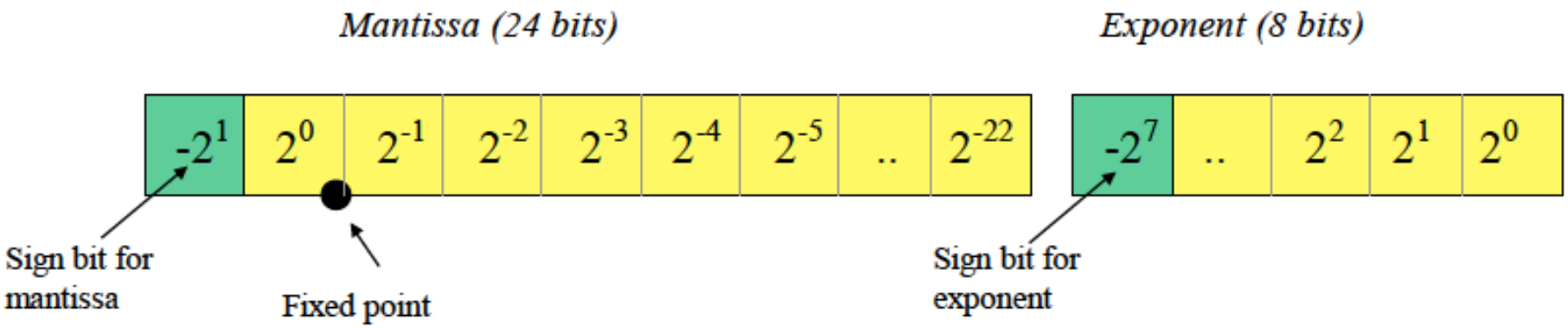
Fixed point, 32 bit

$2^{-0} = 1.0$
$2^{-1} = 0.5$
$2^{-2} = 0.25$
$2^{-3} = 0.125$
$2^{-4} = 0.0625$
$2^{-5} = 0.03125$
$2^{-6} = 0.015625$
....
$2^{-15} = 0.000030517578125$
$2^{-31} = 0.0000000004656$

$$01100100 = 2^{-1} + 2^{-2} + 2^{-5} = 0.5 + 0.25 + 0.03125 = 0.78125$$

$$11100100 = -2^{0} + 2^{-1} + 2^{-2} + 2^{-5} = -1.0 + 0.5 + 0.25 + 0.03125 = -0.21875$$

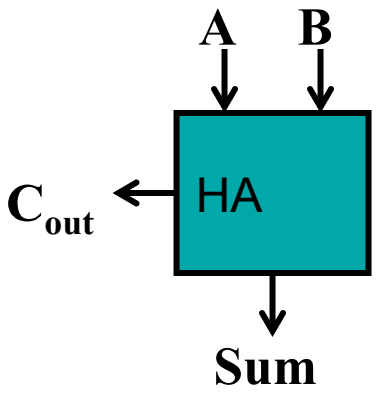
Real numbers: Floating point



$$data = mantissa \times 2^{exponent}$$

$$\begin{aligned}(0110100\cdots)(0\cdots0101) &= (2^0 + 2^{-1} + 2^{-3}) \times (2^2 + 2^0) \\ &= (1.0 + 0.5 + 0.125) \times (4 + 1) \\ &= 1.625 \times 2^5 = 52.0\end{aligned}$$

The Half Adder

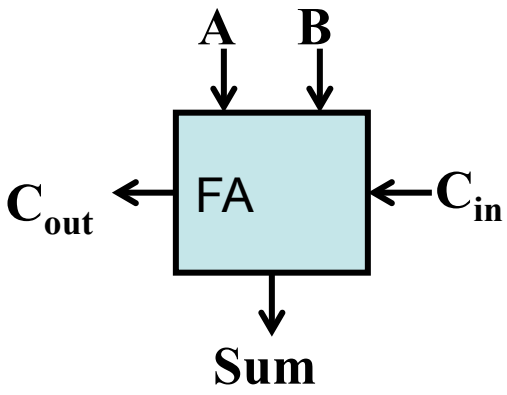


Sum = A XOR B

Cout = A AND B

A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The Full Adder

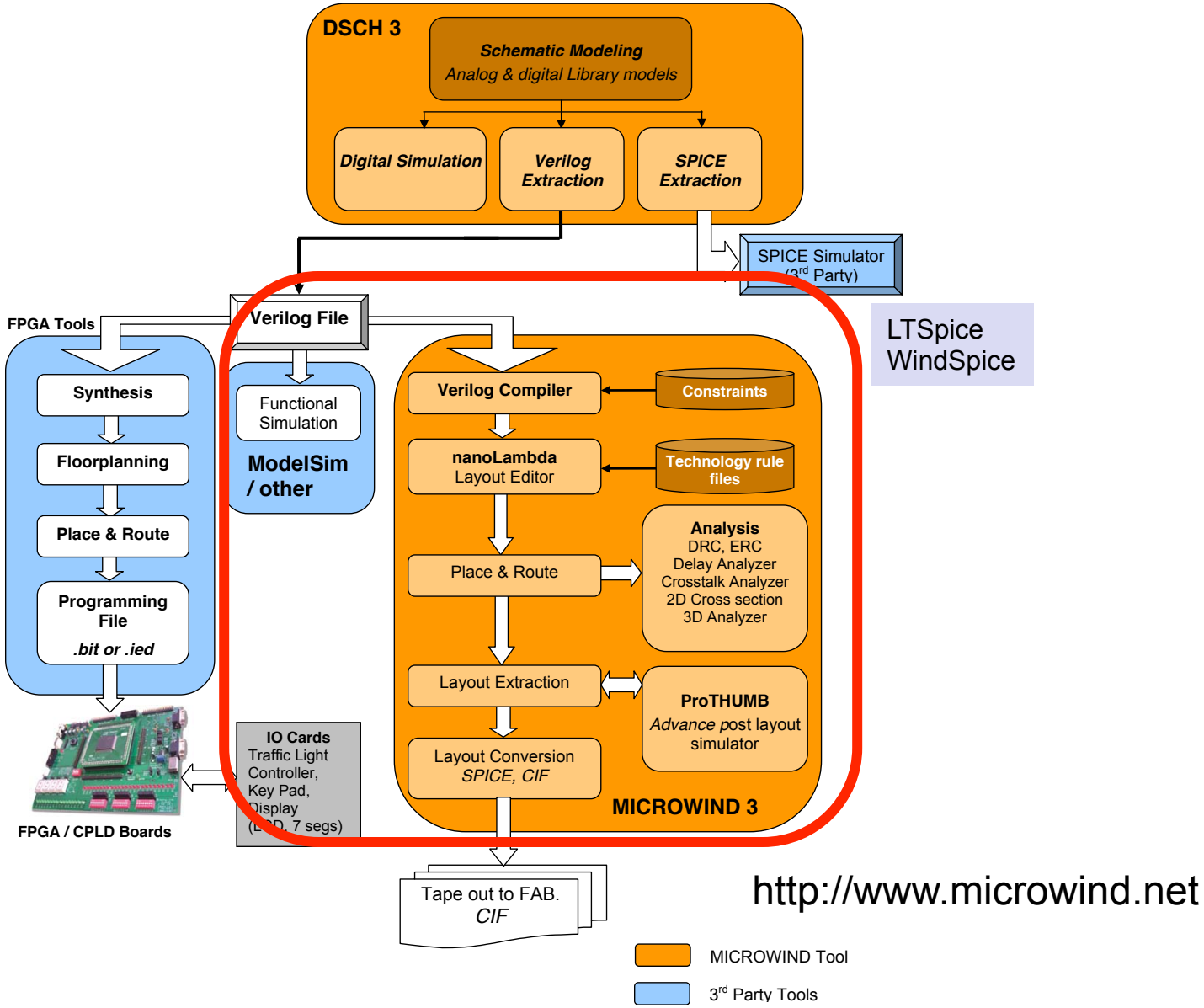


Sum = A XOR B XOR Cin

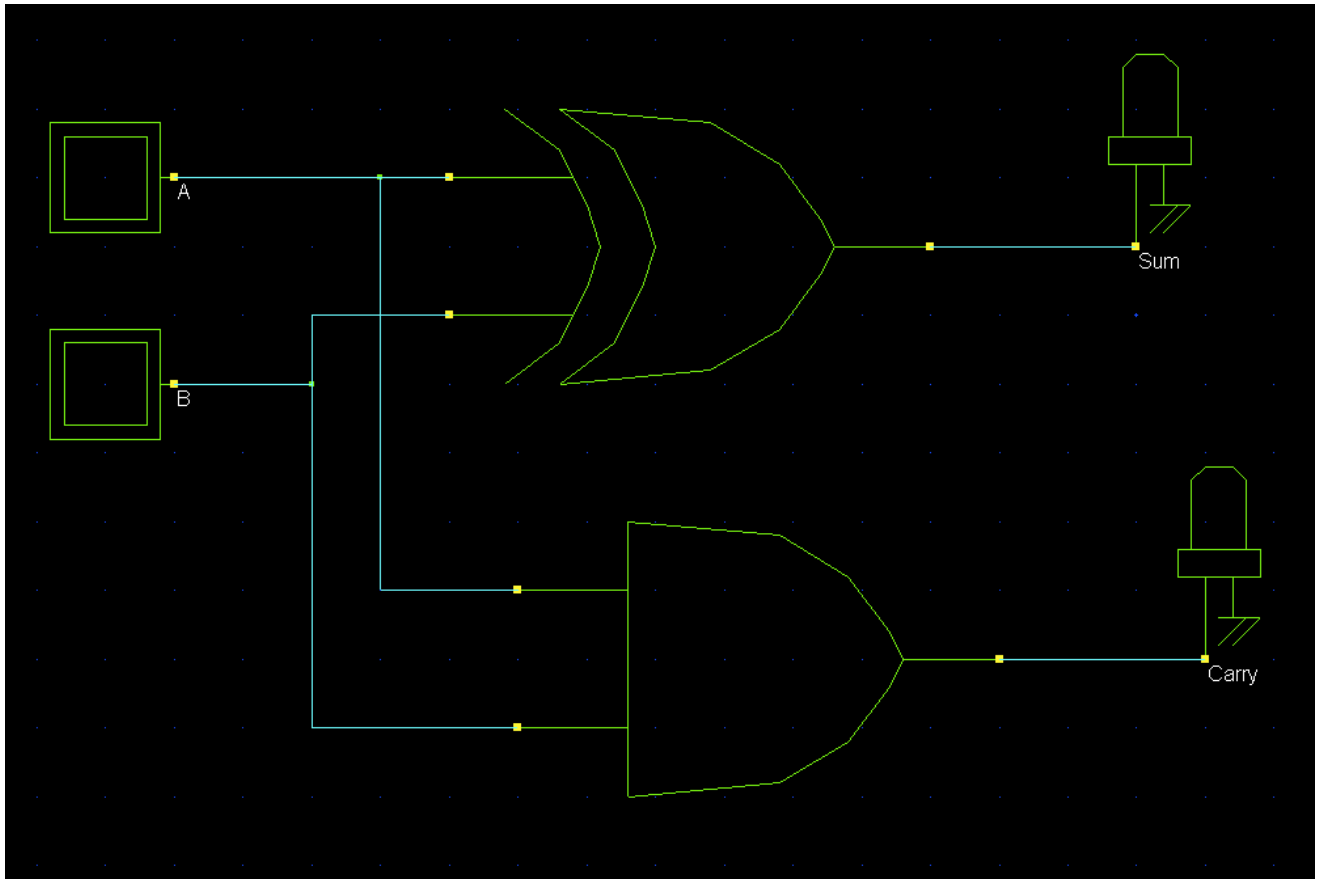
Cout = (A AND B) OR (B AND Cin)
OR (A AND Cin)

C_{in}	A	B	Sum	Cout
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

Computer Aided Design Tools

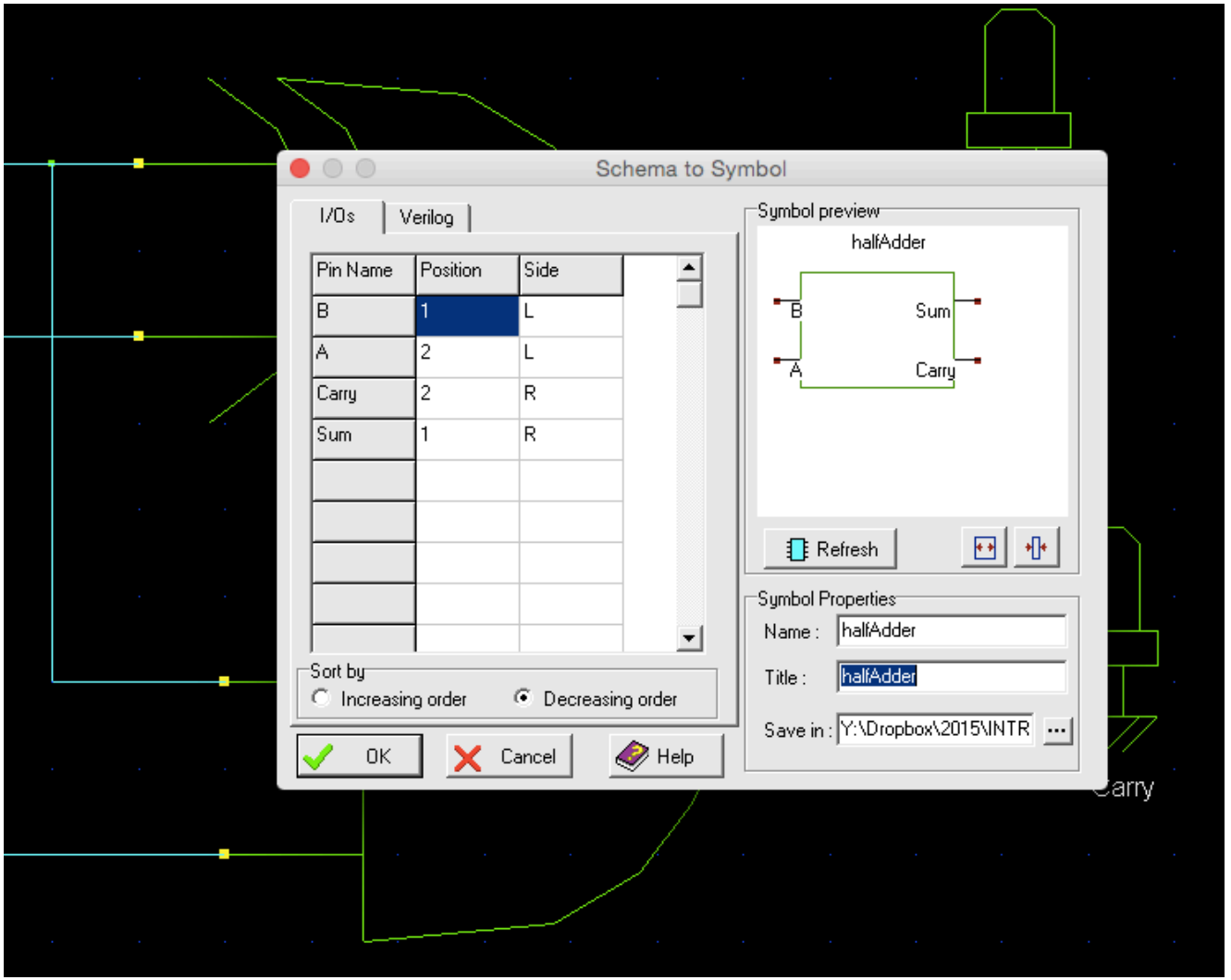


Half Adder

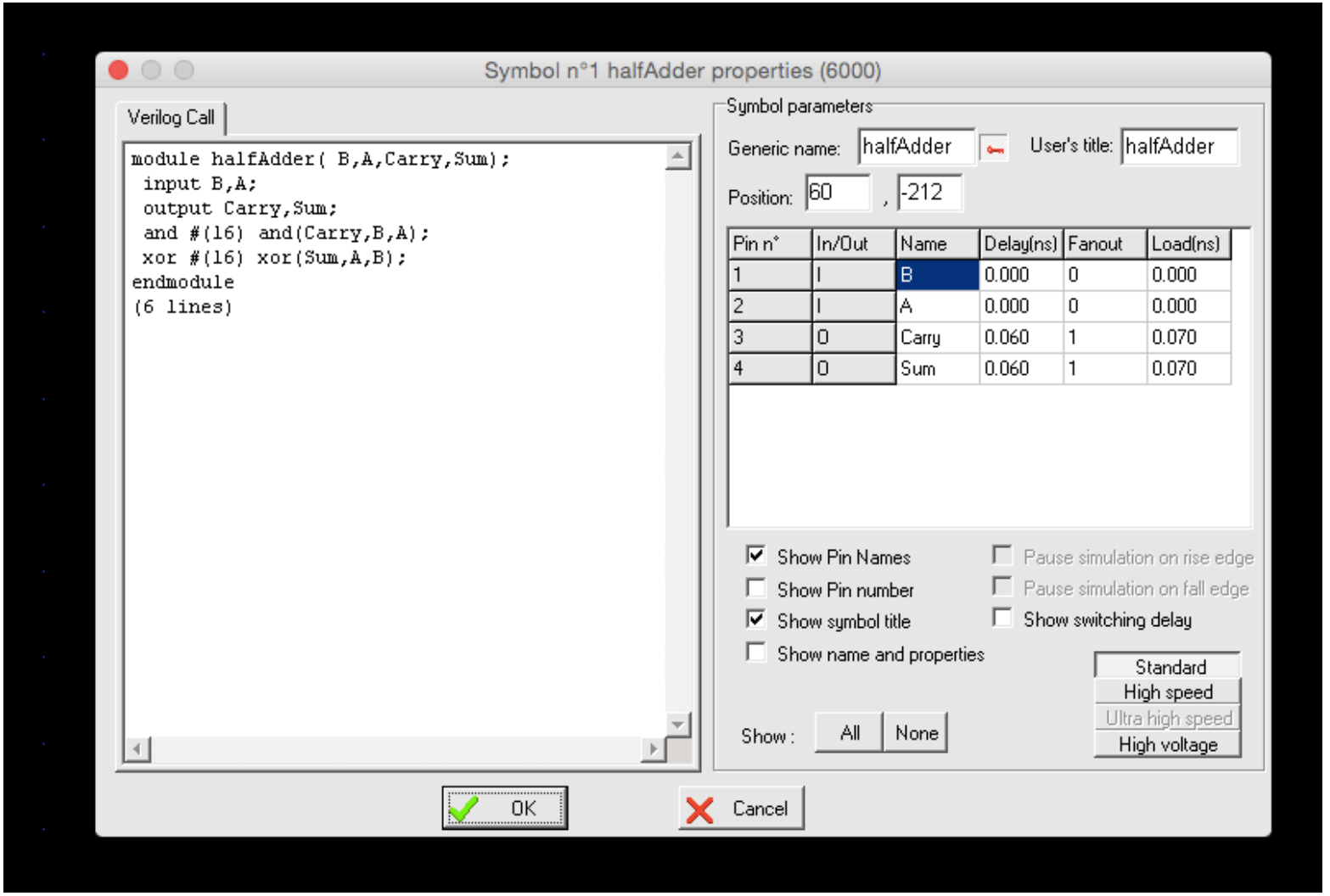


HalfAdder.sch

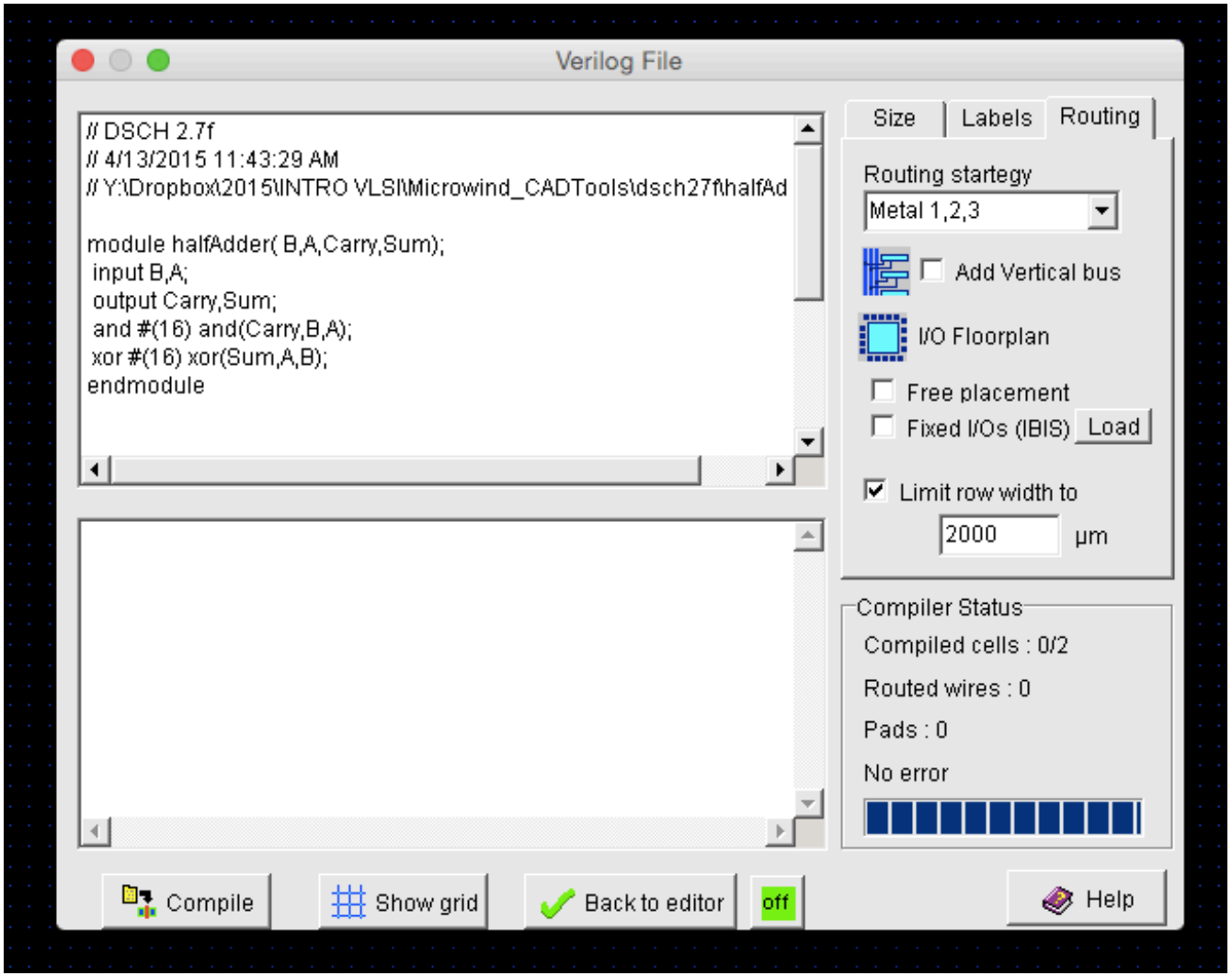
Generate symbol and Verilog



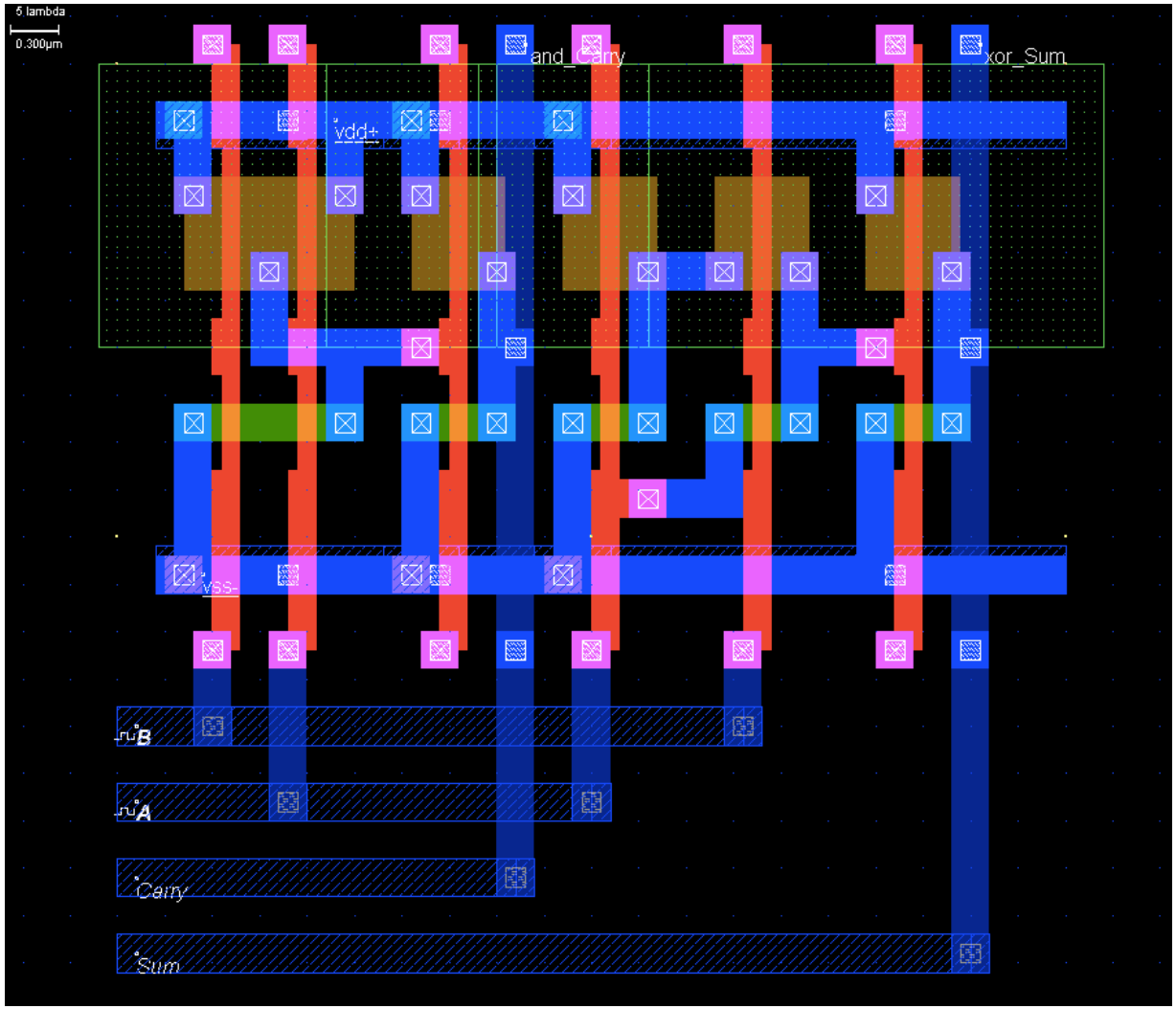
Verilog file



Import from Microwind

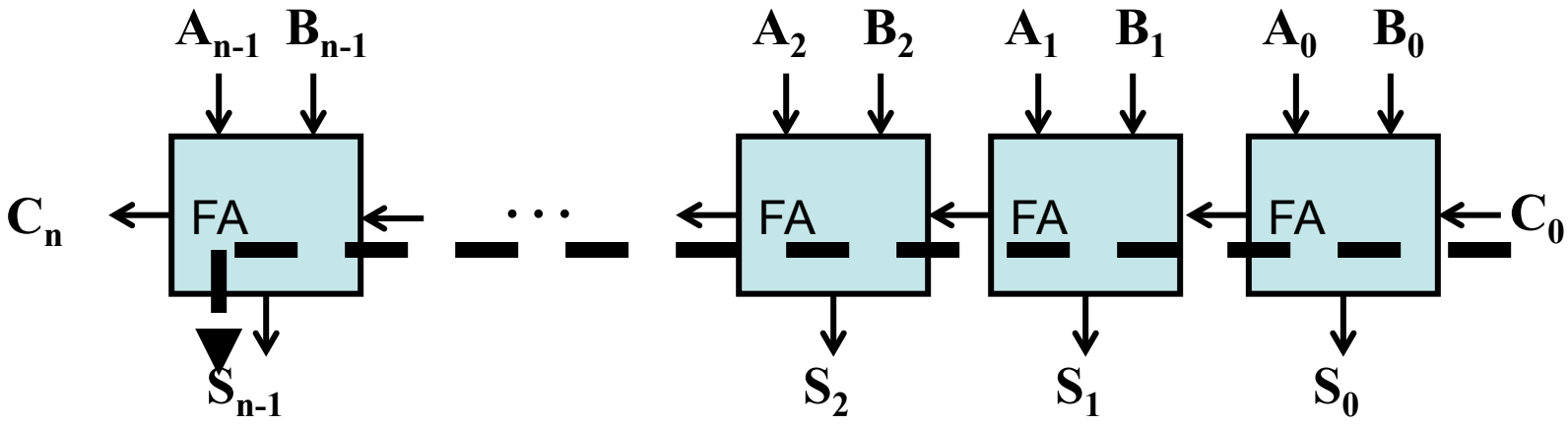


Layout

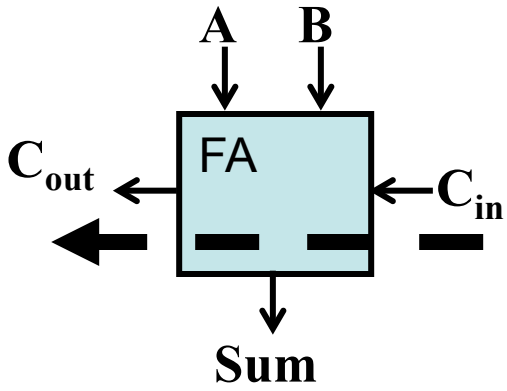


N-Bit Ripple-Carry Adder: Series of FA Cells

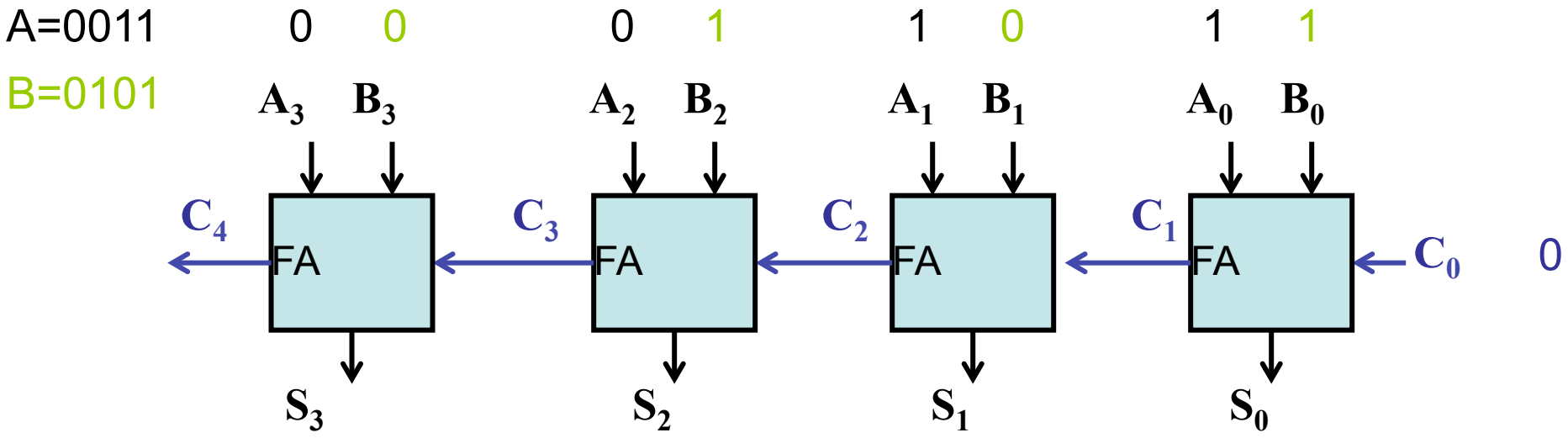
- To add two n-bit numbers



- Note: adder delay = Tc * n
- Tc = (C_{in}:C_{out} delay)

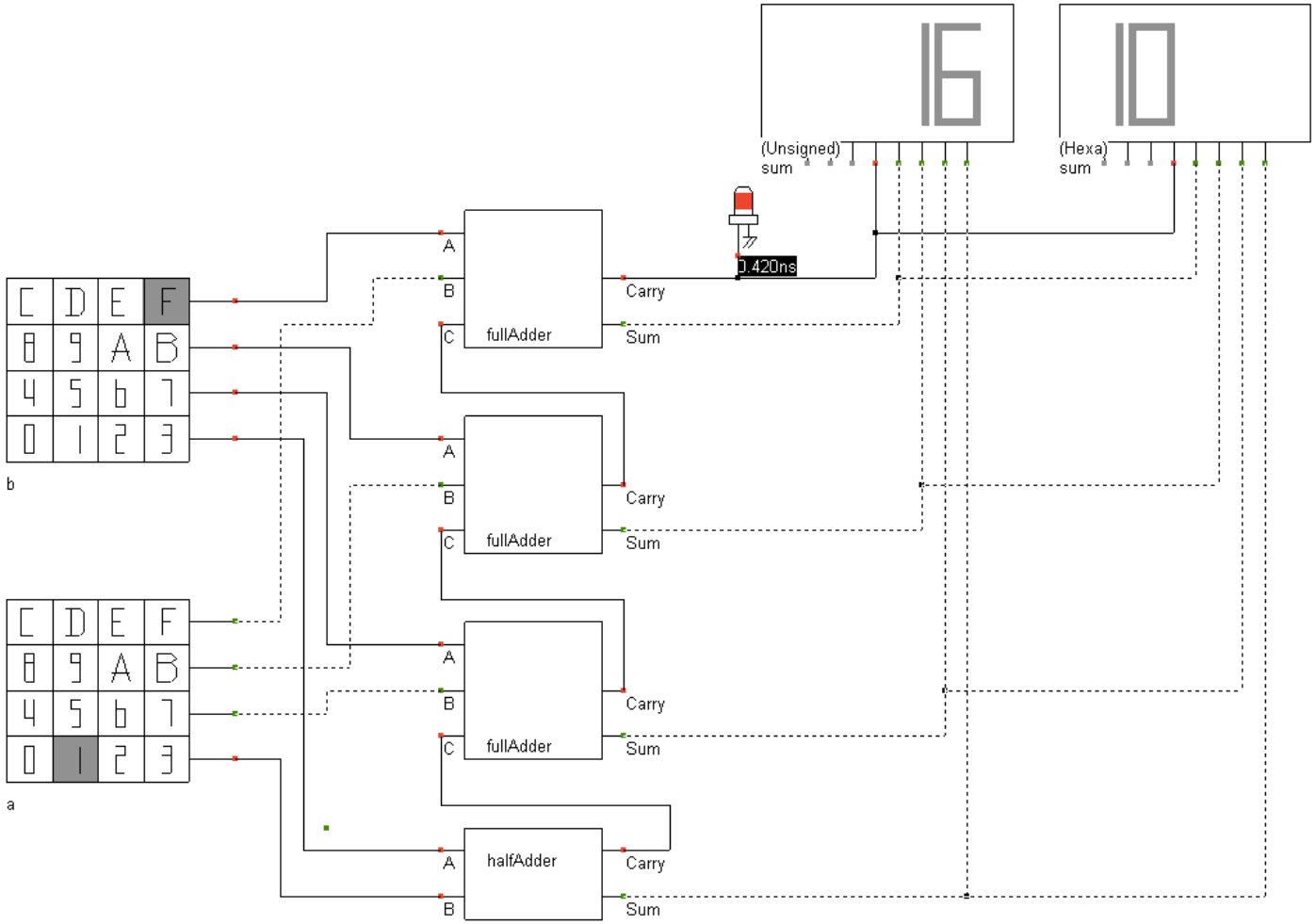


4-bit Ripple Carry Addition: Example

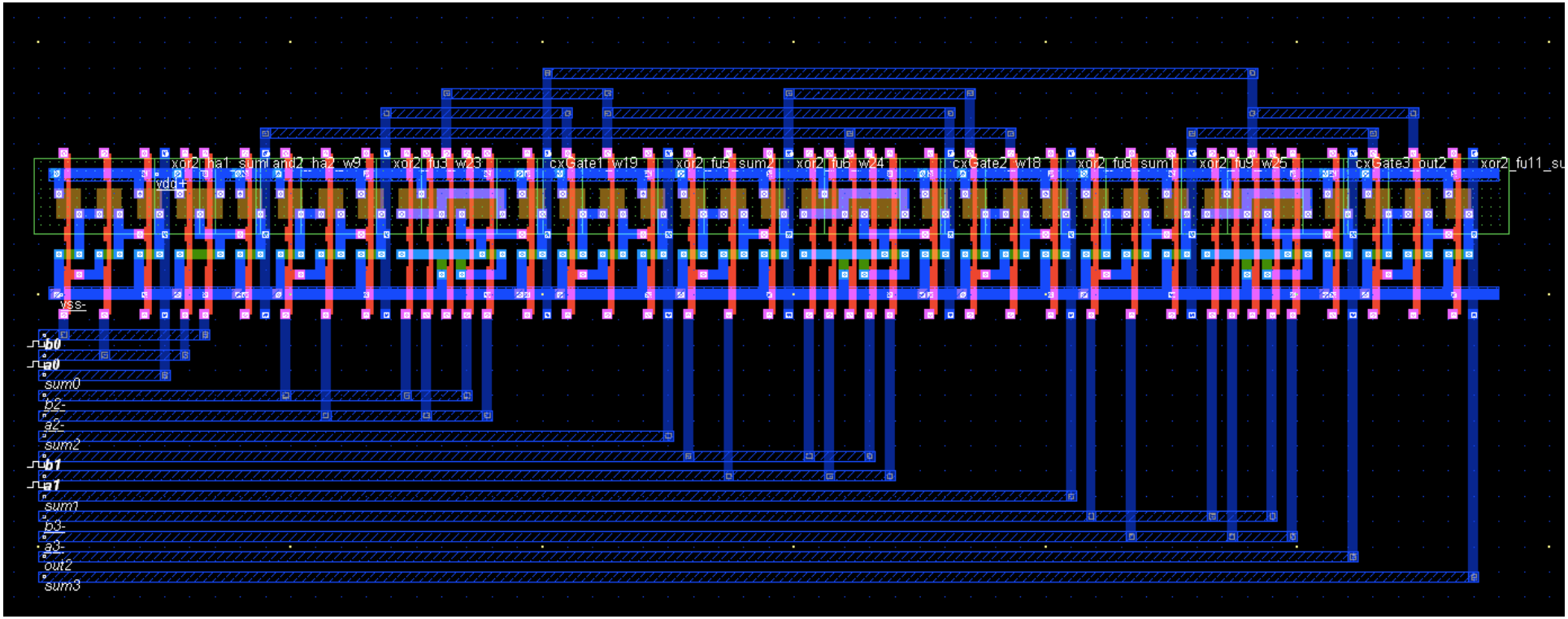


T=0	0	0	0	0	0	0	0	0	S=0000
T=1	0	0	0	1	0	1	1	0	S=0110
T=2	0	0	0	1	1	0	1	0	S=0100
T=3	0	0	1	0	1	0	1	0	S=0000
T=4	0	1	1	0	1	0	1	0	S=1000

4-bit example



4-bit adder layout



One-Bit Full Adder: Share Logic

- An observation
  - Almost always,
    sum = NOT carry

includes 111

Sum = A.B.Cin +
(A+B+Cin).**Cout**

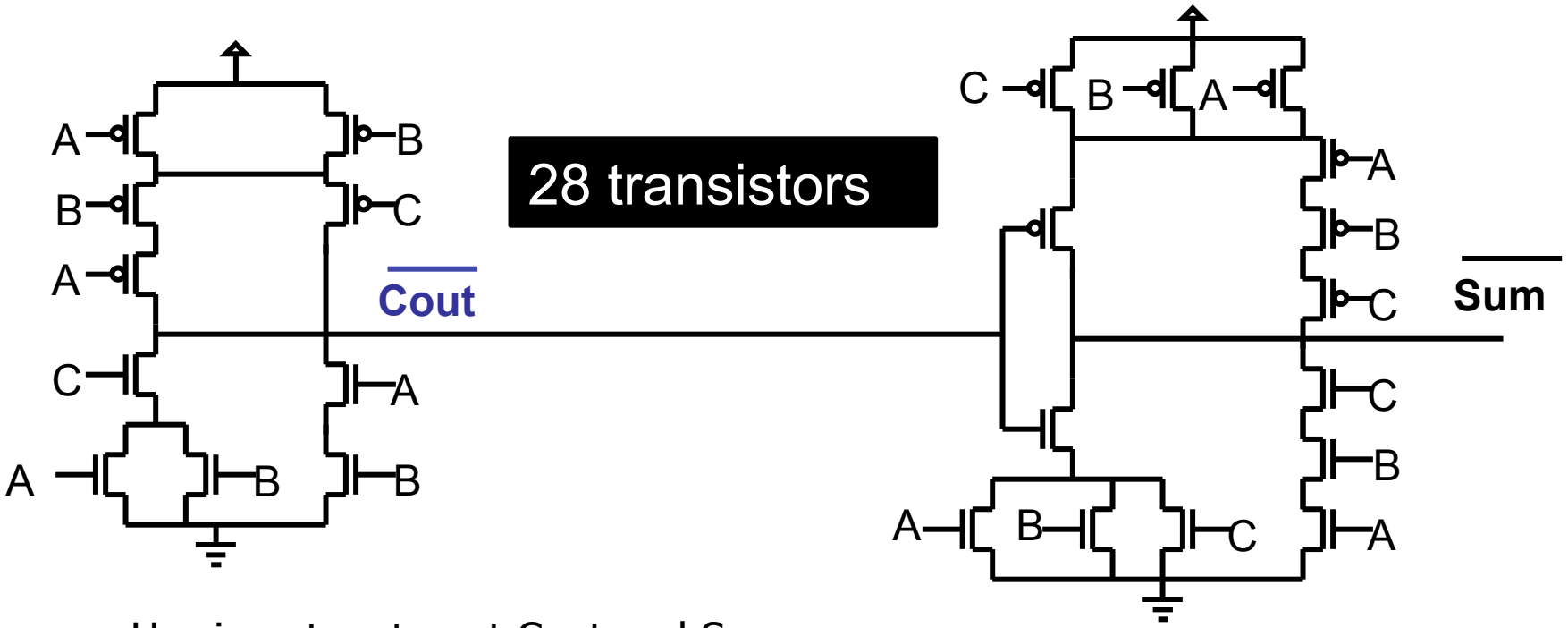
excludes 000

C_{in}	A	B	Sum	Cout
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

One-Bit Full Adder: Transistor Implementation

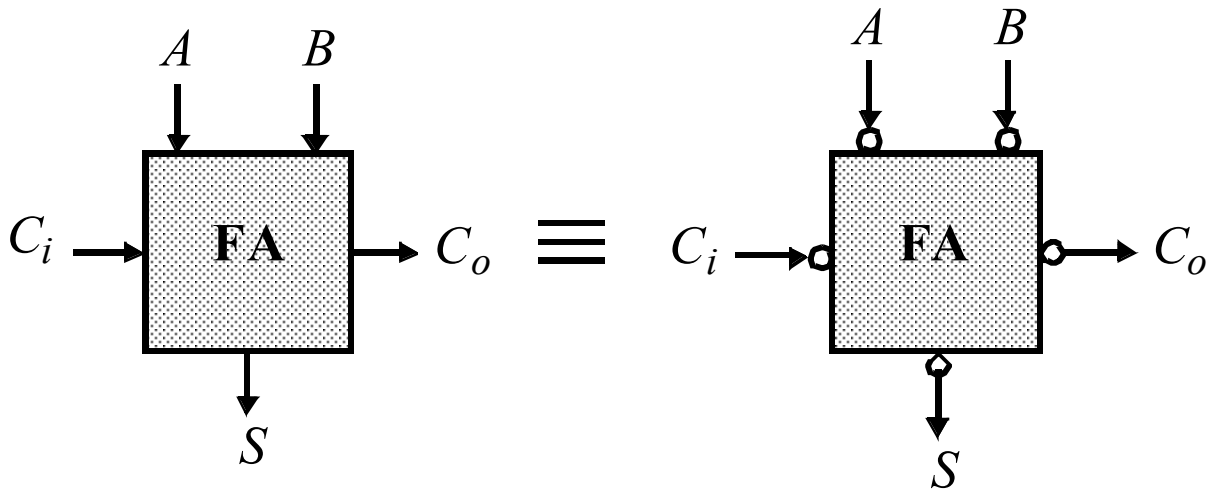
Cout = A.B + C.(A+B)

Sum = A.B.C + (A+B+C).$\overline{\text{Cout}}$



- Use inverters to get Cout and Sum
- C transistors close to output
- Cout delay: 2 inverting stages (1-stage possible?)
- Sum delay: 3 inverting stages (not an issue, though)

Inversion Property



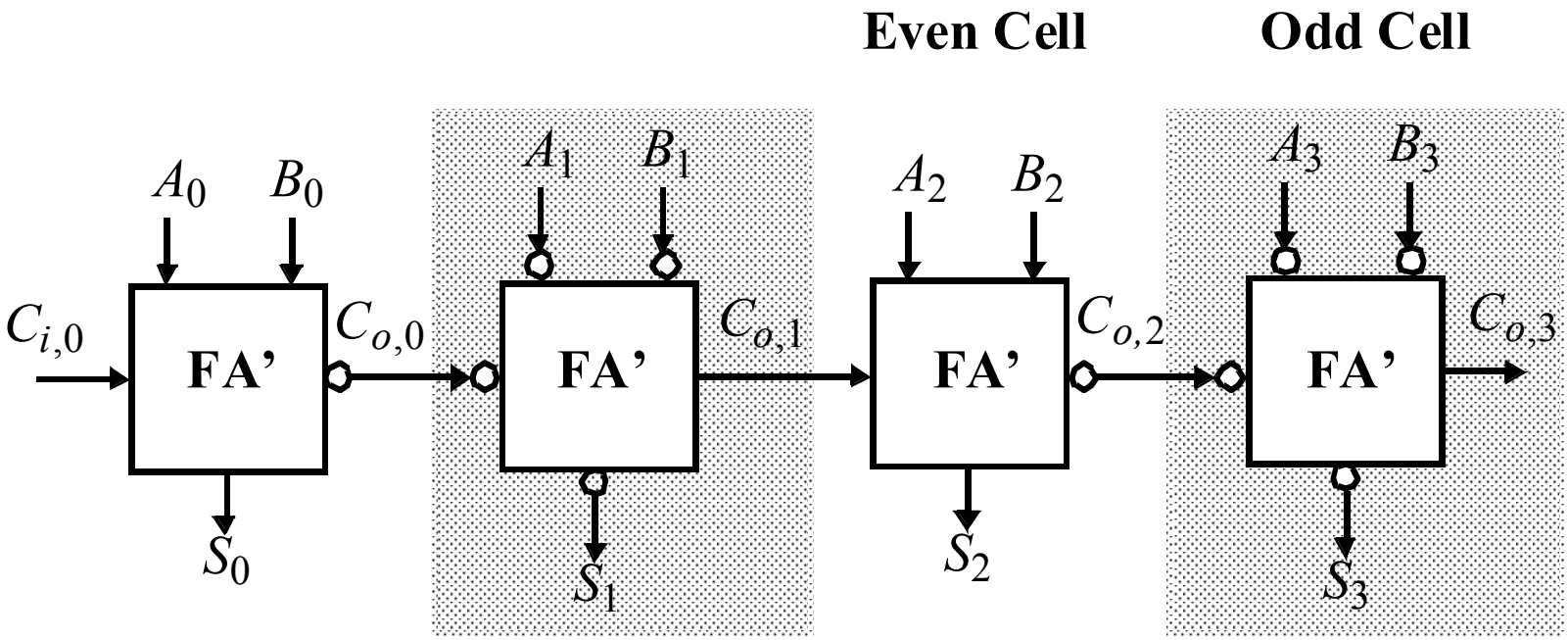
$$\bar{S}(A,B,C_i) = S(\bar{A},\bar{B},\overline{C_i})$$

$$\overline{C_o}(A,B,C_i) = C_o(\bar{A},\bar{B},\overline{C_i})$$

From Rabaey

Critical Path

- Reducing Inverting Stages



  - Exploit inversion property
  - Now we need 2 different type of cells

From Rabaey