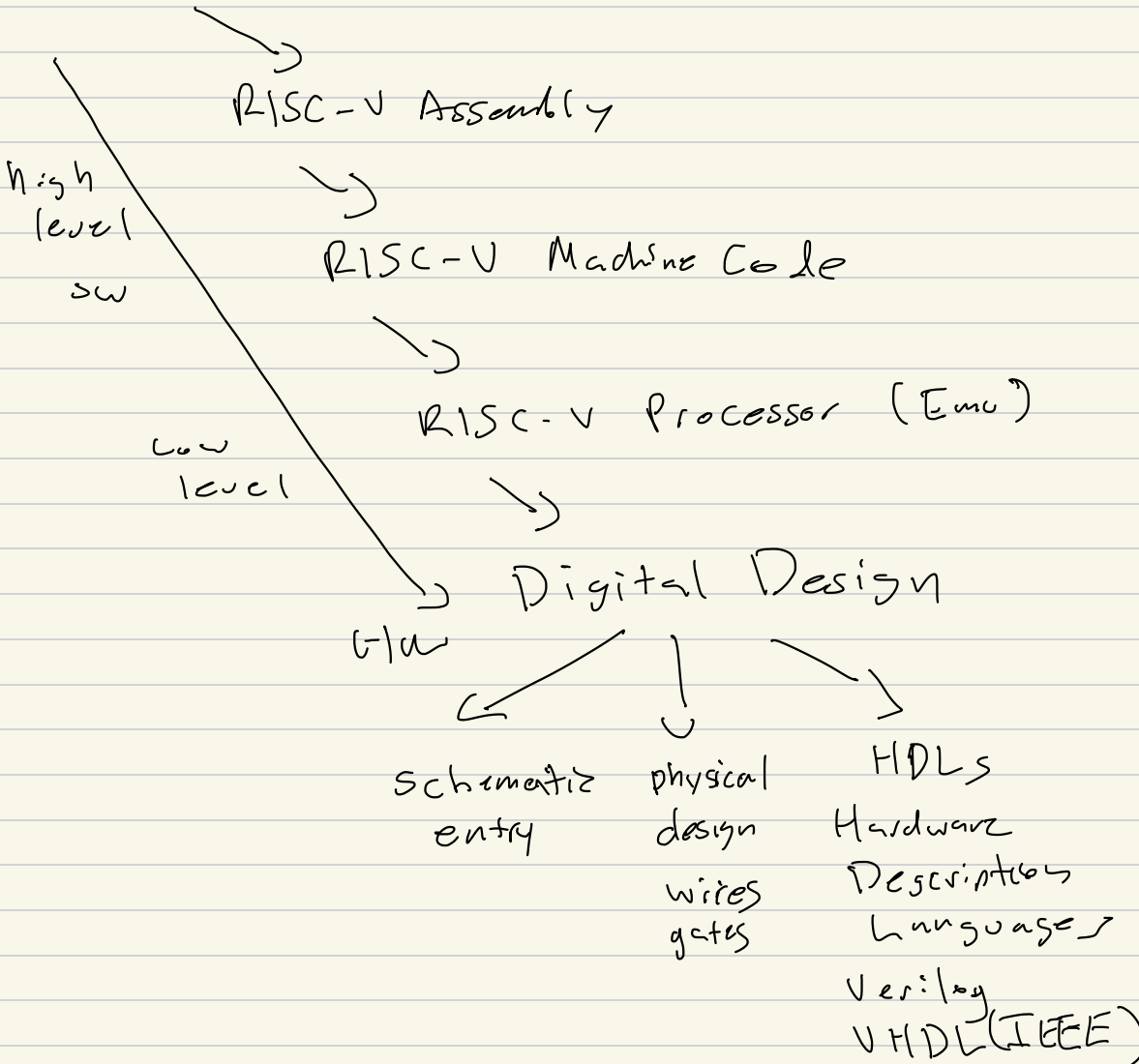
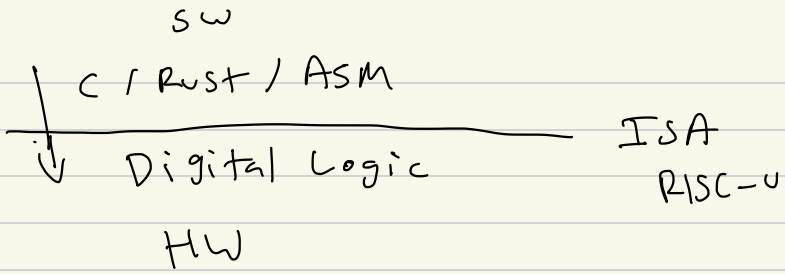


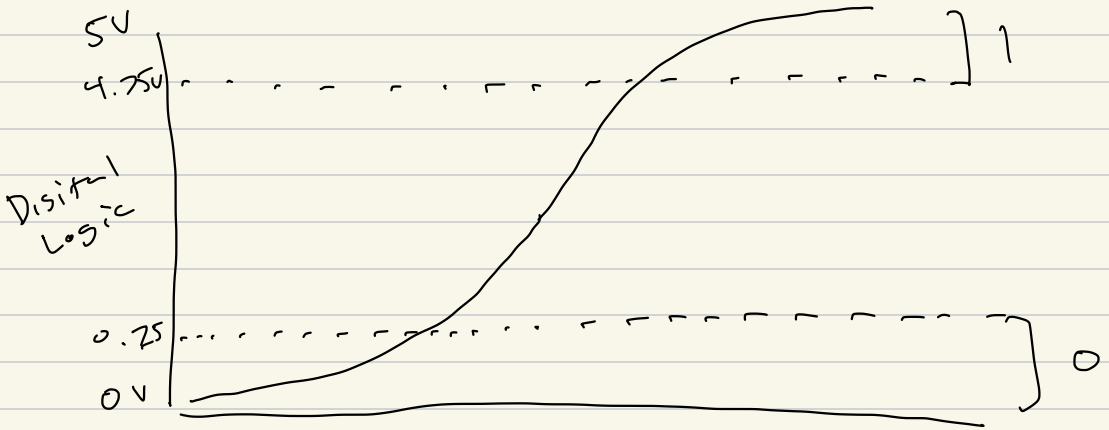
Combinational Logic / Sequential Logic

C code / Rust code





Analog \rightarrow Digital

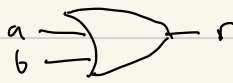
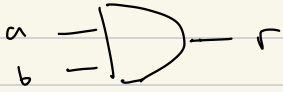


power source
wires
devices \rightarrow gates

AND

OR

NOT



C/Rust

$r = a \& b$

$r = a | b$

$r = !a$

Boolean
Algebra

$r = a \cdot b$

$r = a + b$

$r = \bar{a}$

Logic

$r = a \wedge b$

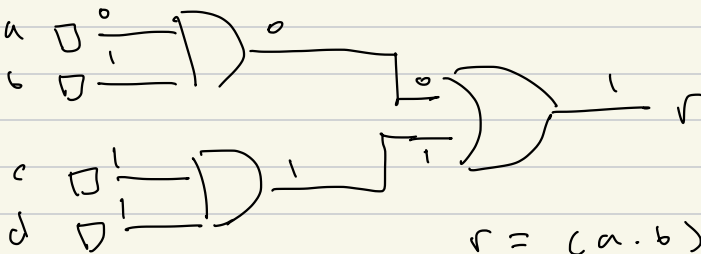
$r = a \vee b$

$r = \neg a$

a	b	r
0	0	0
0	1	0
1	0	0
1	1	1

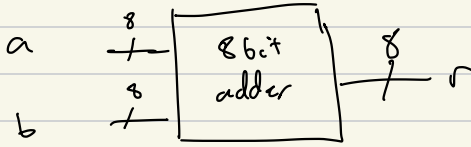
a	b	r
0	0	0
0	1	1
1	0	1
1	1	1

a	r
0	1
1	0



$r = (a \cdot b) + (c \cdot d)$

Goal: 8-bit adder



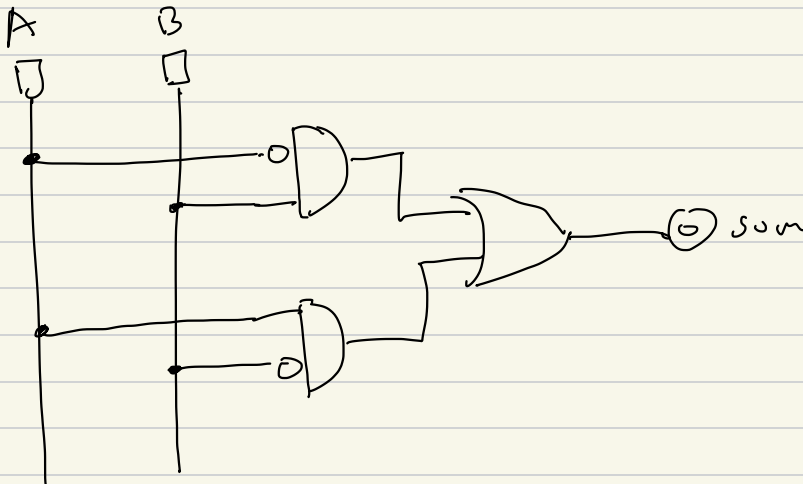
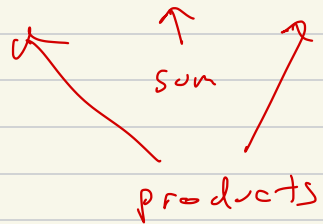
sum-of-products

sum of two 1-bit values

$$\text{XOR sum} = a \oplus b$$

a	b	sum
0	0	0
0	1	1
1	0	1
1	1	0

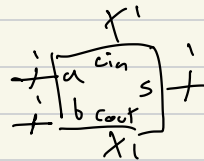
$$\text{sum} = (\bar{a} \cdot b) + (a \cdot \bar{b})$$



Sum-of-products

- 1) Define your function
- 2) Build truth table
- 3) Identify rows with output = 1
- 4) construct product terms for each row
 - a) don't invert if input is 1
 - b) invert if input is 0
- 5) Sum (+) all product terms.

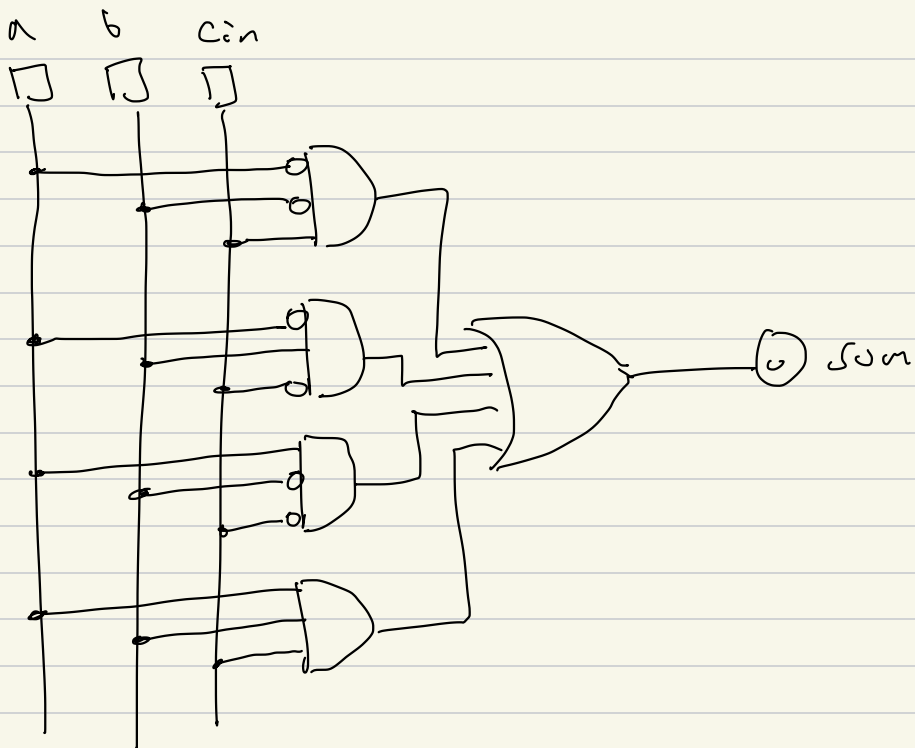
1 bit full adder



a	b	cin	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

$$\text{sum} = (\bar{a} \cdot \bar{b} \cdot c_{in}) + (\bar{a} \cdot b \cdot \bar{c}_{in}) + (a \cdot \bar{b} \cdot \bar{c}_{in}) + (a \cdot b \cdot c_{in})$$

$$\text{cout} = \dots$$



4 bit adder - 4 bit Ripple Carry Adder

$a[4]$ $a_3 a_2 a_1 a_0$

$b_3 b_2 b_1 b_0$

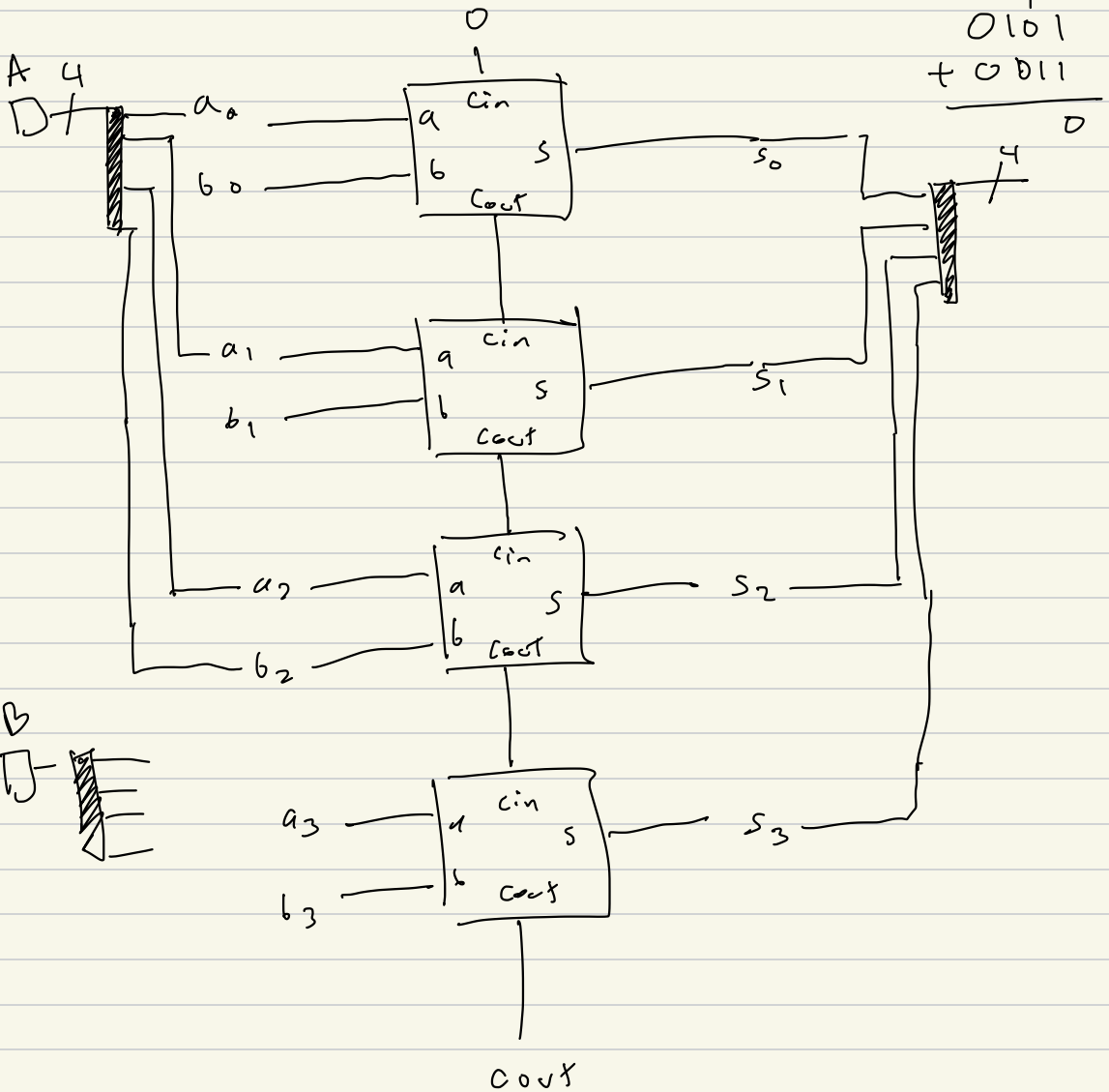
cin
0101

+ 0011

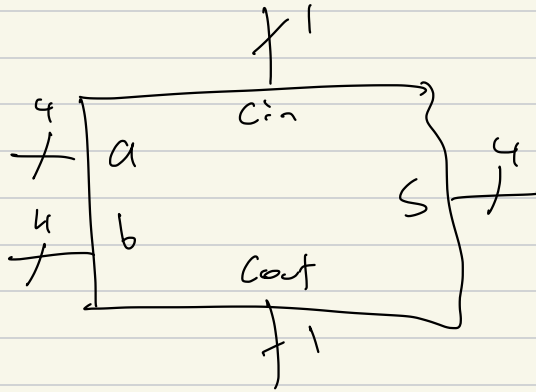
4

0

A 4
D+



4 bit ripple-carry adder



Goal N-bit register

State

SR Latch

