Fundamental Computer Architecture

Luslab Computing Seminars

Last Time - From this!

Royal Belgian Institute of Natural Sciences

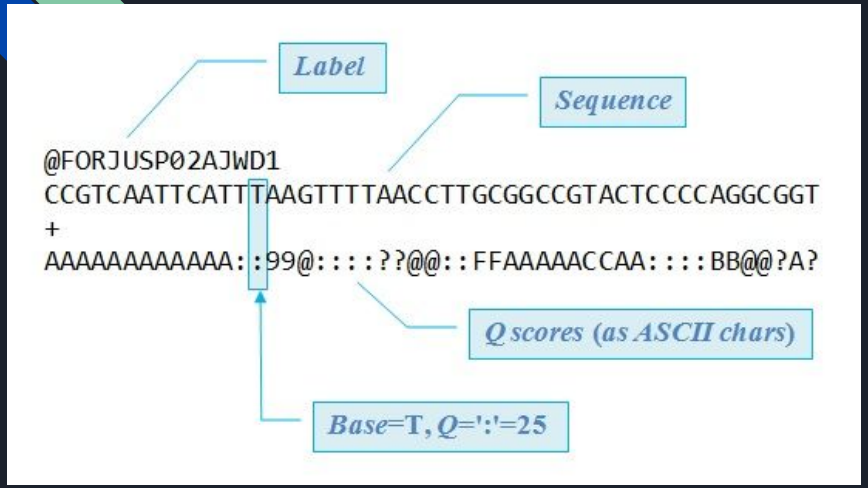
To This!

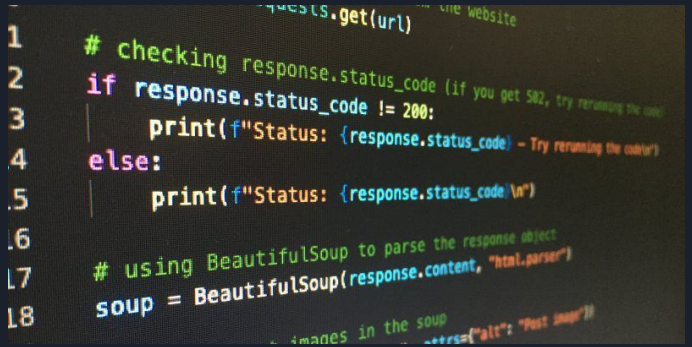
1945 onwards

Computational Data Drivers

- Storage / Representation
- Simple Arithmetic
- Higher Order Calculation
- Accuracy
- Volume

Basic Computing Requirements

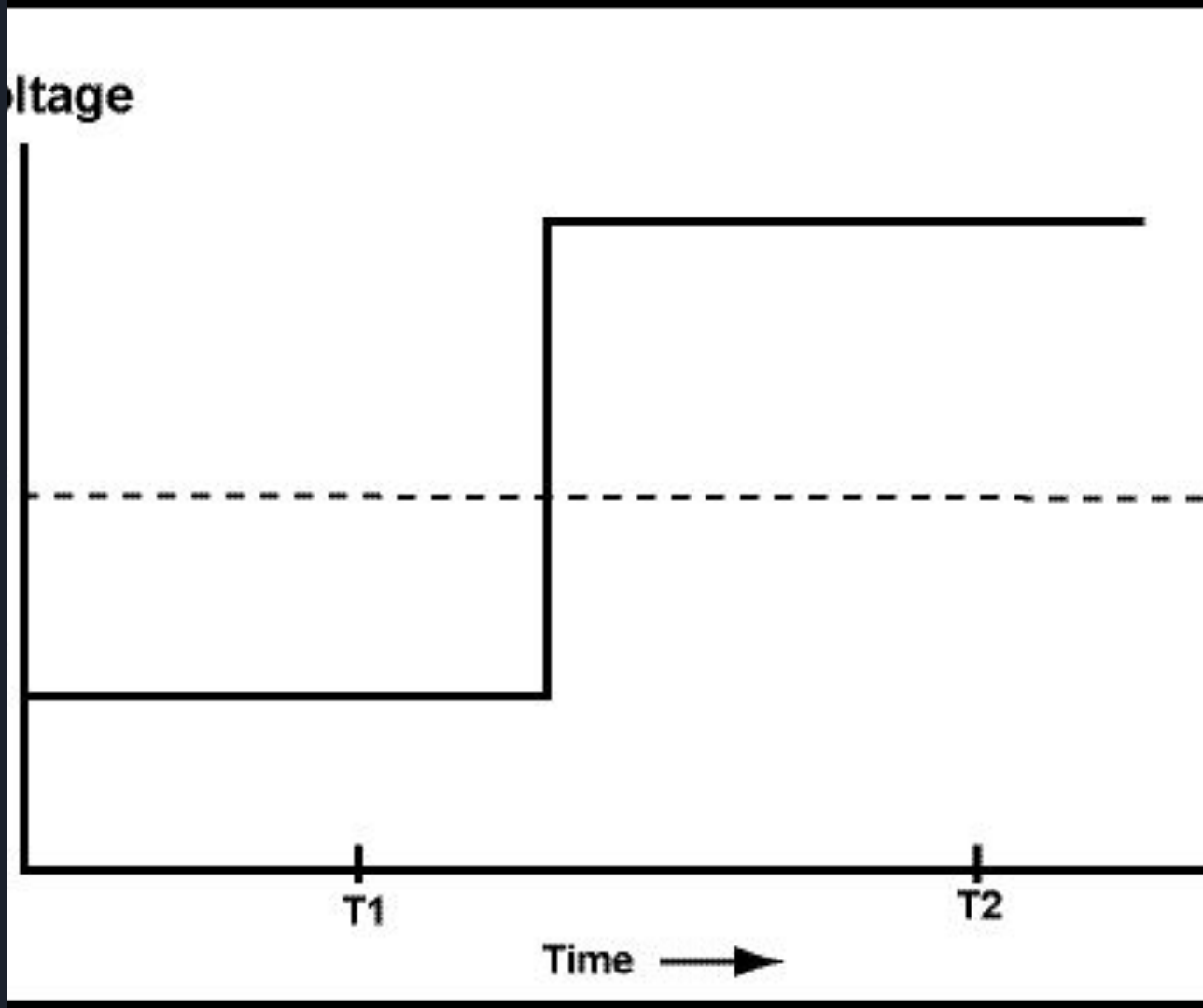




Normal instruc	
00000	SUB
00001	AND
00010	ADD
00011	OR
00100	XOR
00101	LSR
00110	LSL
00111	ASR
01000	BREV
01001	LDILO

Binary Data Representation

- Base 10 needs enough voltage in between numbers to make a correct reading
- Base 2 is either on/off - convenient
- Binary logic is easy to understand



Binary Numbers

Binary to Decimal

This binary Number... → **1 1 1 1 1 1 1 1** Equals this Decimal number

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

128 + 64 + 32 + 16 + + + 2 + 1 = 255

This binary Number... → **1 0 0 1 0 1 0 1** Equals this decimal number

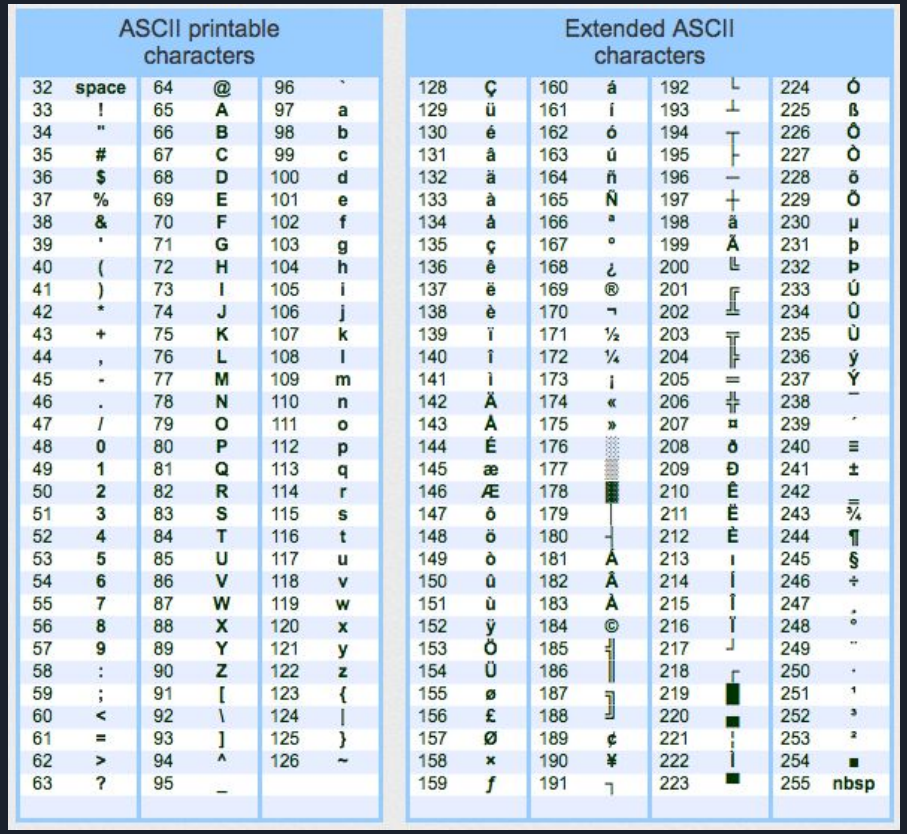
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

128 + 0 + 0 + 16 + + + 0 + 1 = 149

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

Decimal	Binary
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010
19	10011

ASCII



ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Computer Instructions

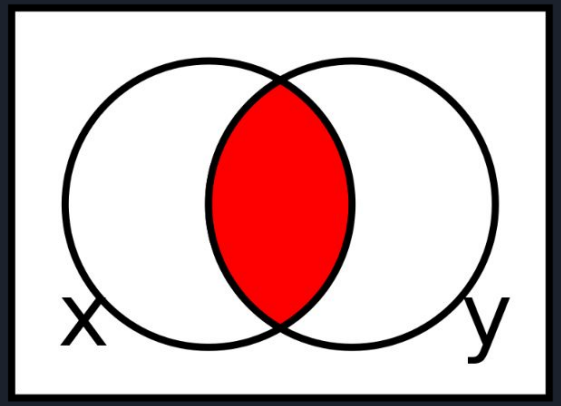
- Every computer has a supported instruction set
- Instruction is also encoded in binary known as machine code
- Instructions used as building blocks for complex programs
- Programming languages abstract machine code away from the user

Computer Instruction Types

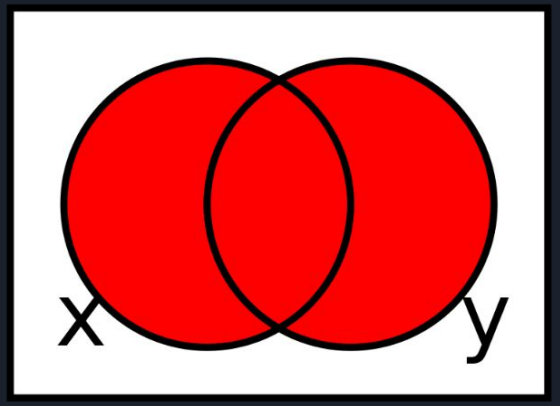
- Data handling and memory operations
- Arithmetic and logic operations
- Control flow operations

Normal instructions				Compressed	
00000	SUB	10000	CMP	000	SUB
00001	AND	10001	TEST	001	AND
00010	ADD	10010	LW	010	ADD
00011	OR	10011	SW	011	CMP
00100	XOR	10100	LH	100	LW
00101	LSR	10101	SH	101	SW
00110	LSL	10110	LB	110	LDI
00111	ASR	10111	SB	111	MOV
01000	BREV	11000	LDI	Reserved for FPU	
01001	LDILO	11001			
01010	MPYUHI	Special Insn		11010	FPADD
01011	MPYSHI			11011	FPSUB
01100	MPY	11100	BREAK	11100	FPMPY
01101	MOV	11101	LOCK	11101	FPDIV
01110	DIVU	11110	SIM	11110	FPI2F
01111	DIVS	11111	NOOP	11111	FPF2I

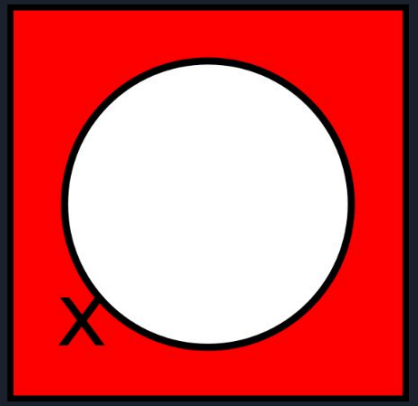
Boolean Logic



$x \wedge y$



$x \vee y$



$\neg x$

Truth Tables

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

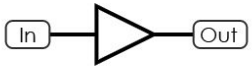
Binary Addition of 2 single bits

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Logic Gates

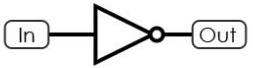
Logic Gates - Symbols and Truth Tables

BUF (Buffer)



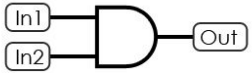
In	Out
0	0
1	1

NOT (Inverter)



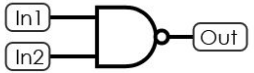
In	Out
0	1
1	0

AND



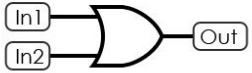
In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

NAND (NOT AND)



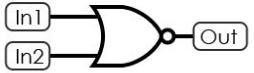
In1	In2	Out
0	0	1
0	1	1
1	0	1
1	1	0

OR



In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	1

NOR (NOT OR)

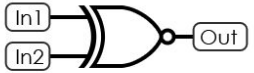


In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	0

XOR (Exclusive Or)

In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	0

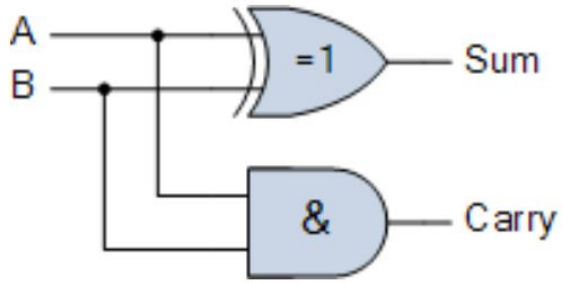
XNOR (NOT XOR)



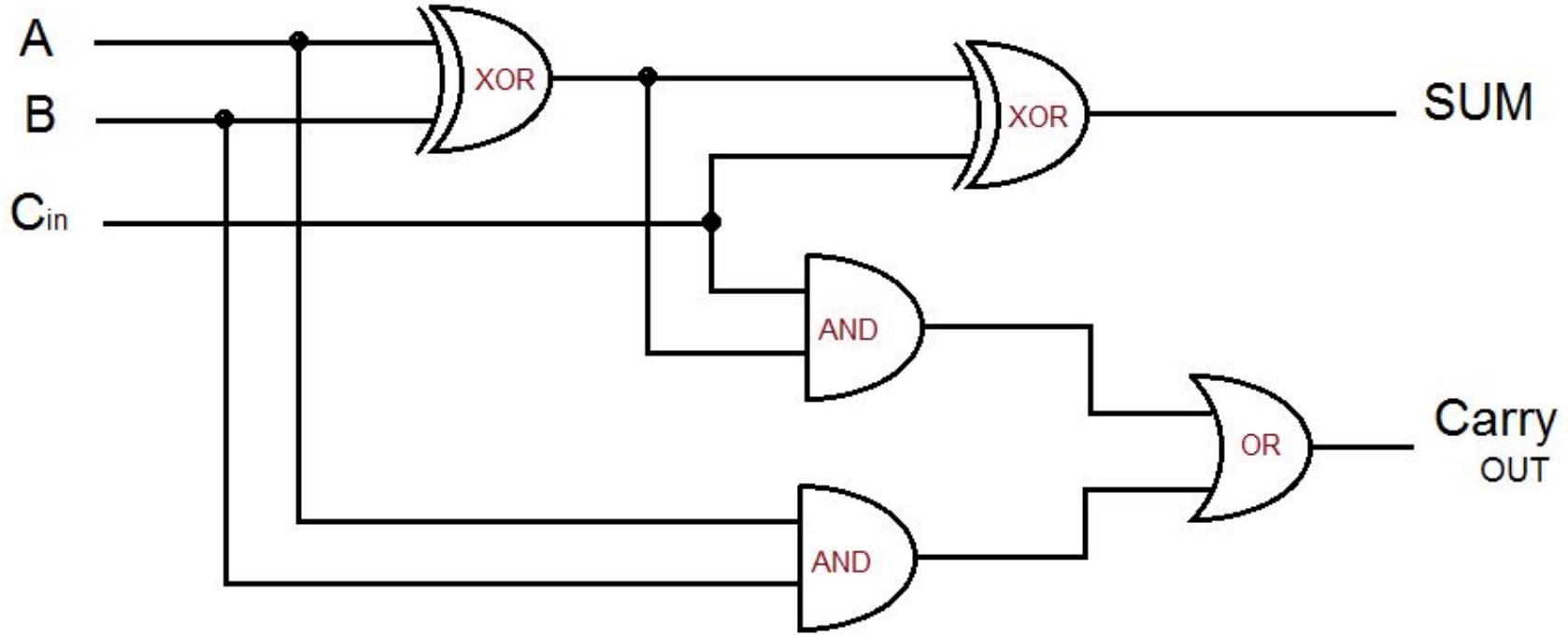
In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	1

A circle behind a symbol indicates that the output signal is inverted.

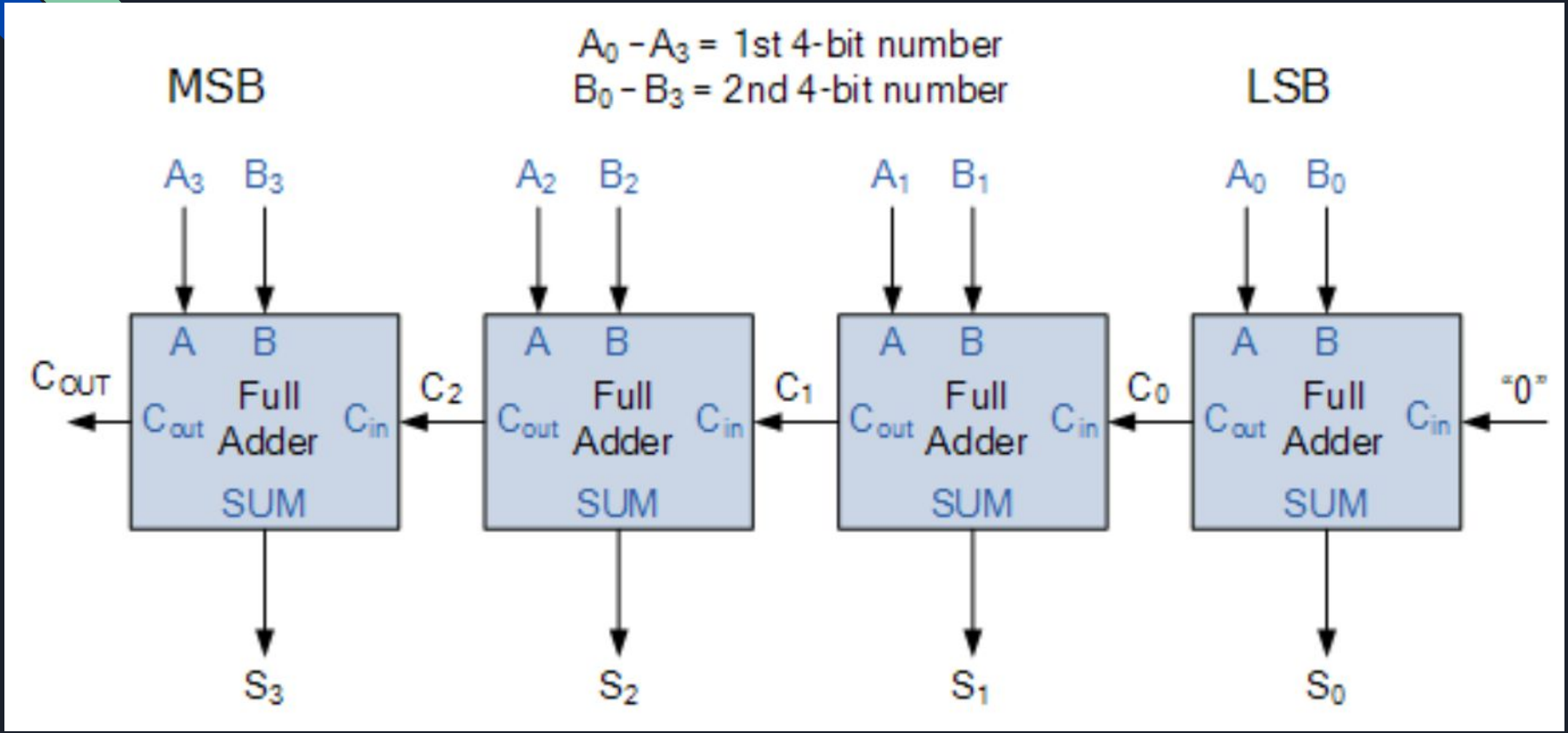
The Half Adder

Symbol	Truth Table			
	B	A	SUM	CARRY
	0	0	0	0
	0	1	1	0
	1	0	1	0
	1	1	0	1

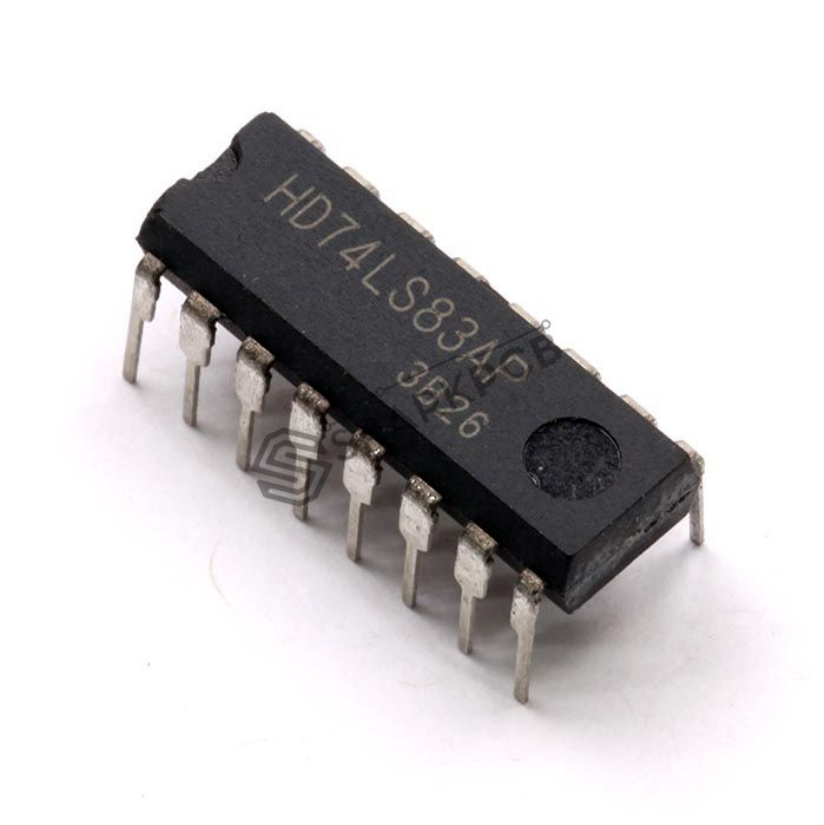
The Full Adder

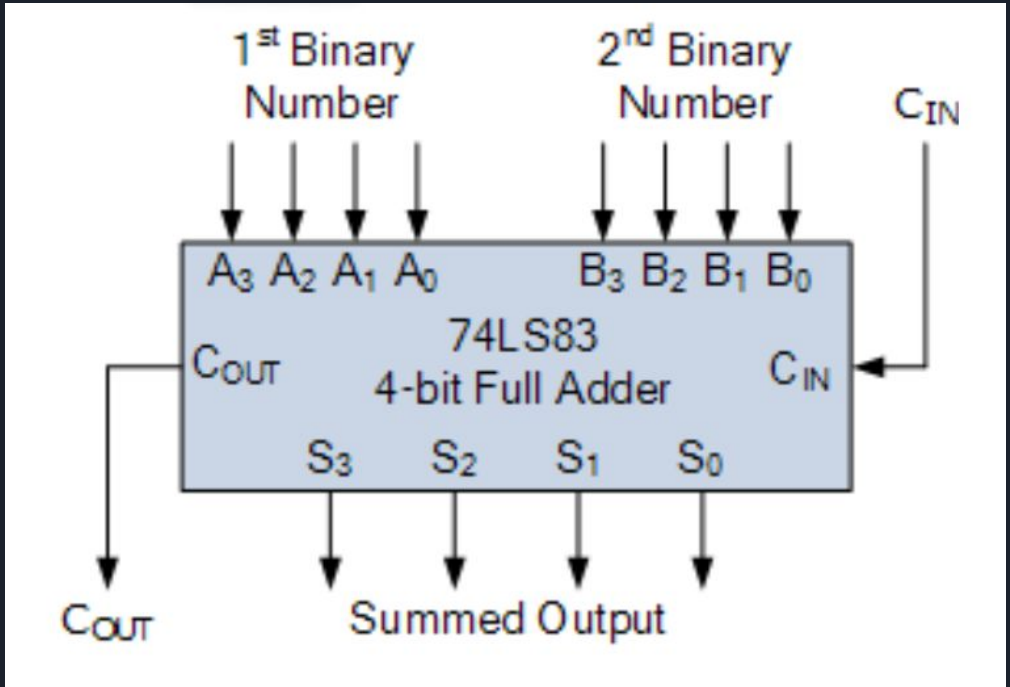


A 4-bit Ripple Carry Adder

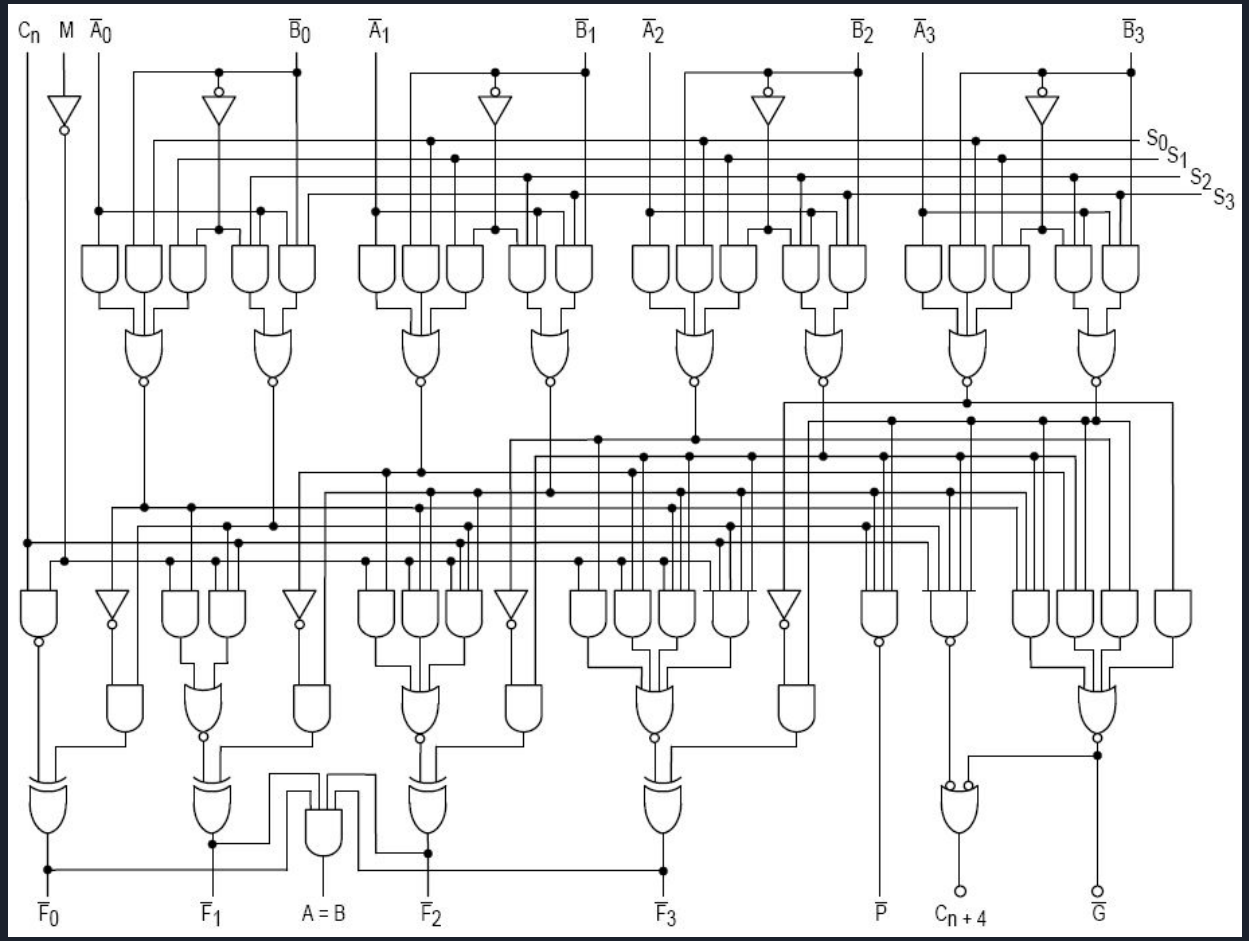


You can actually buy this adder!



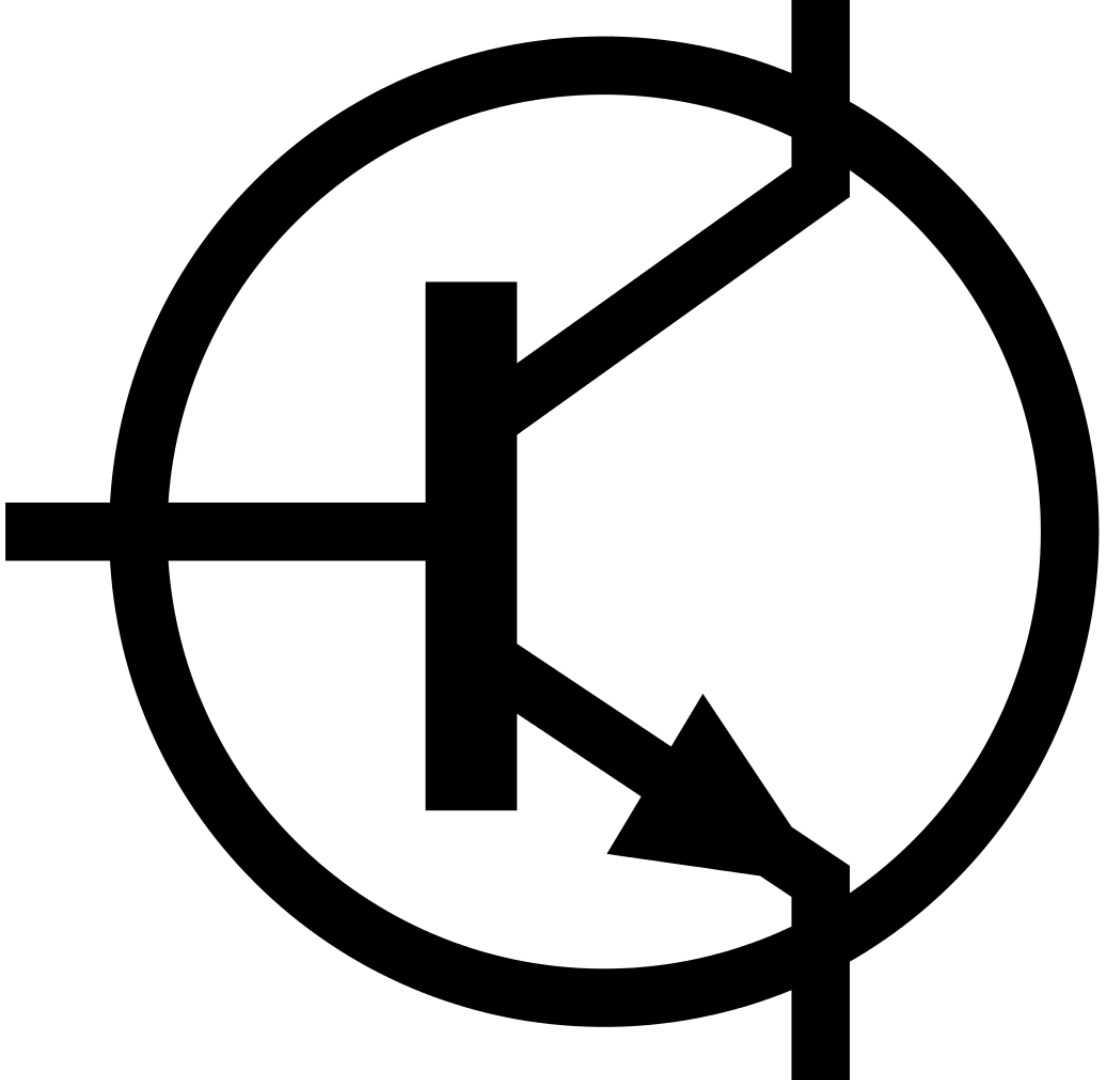


Other Logic circuits

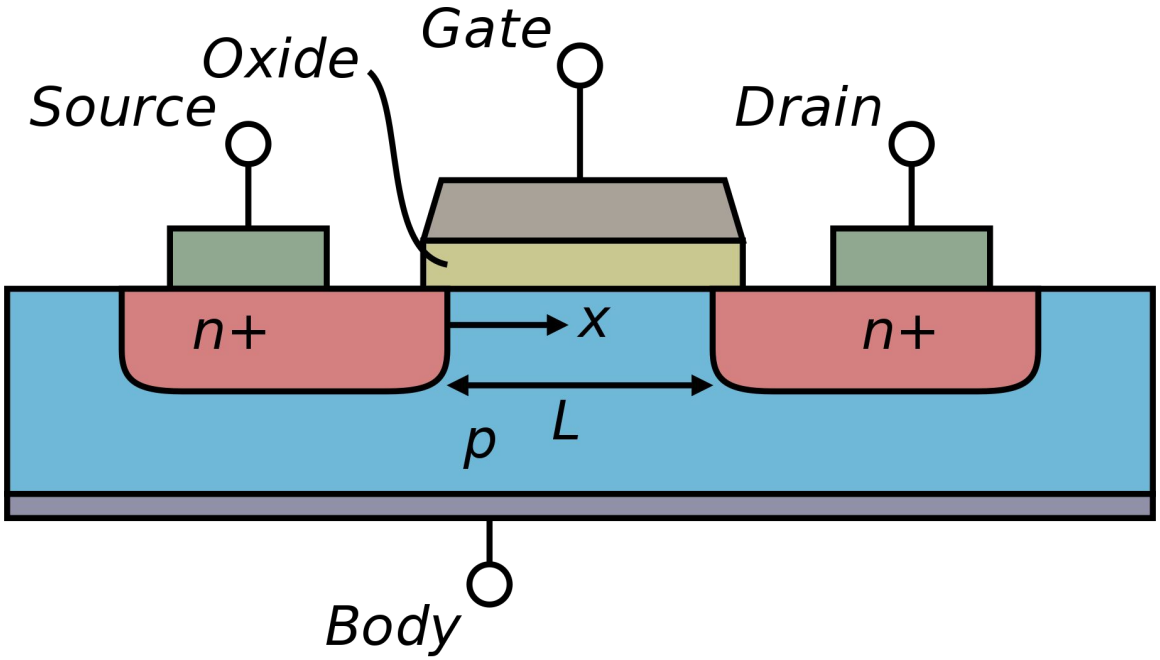


The Humble Transistor

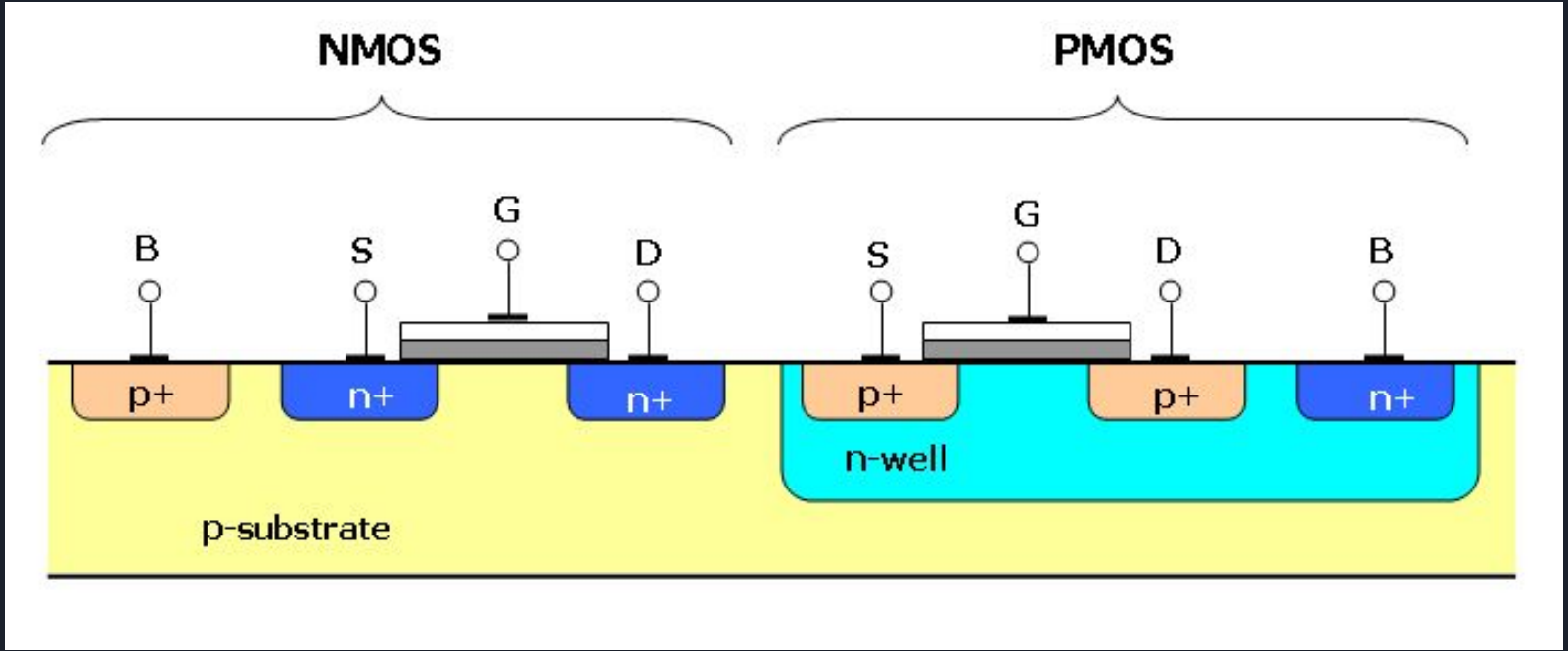
- Passes current through from top to bottom depending on whether there is a current/signal at the base

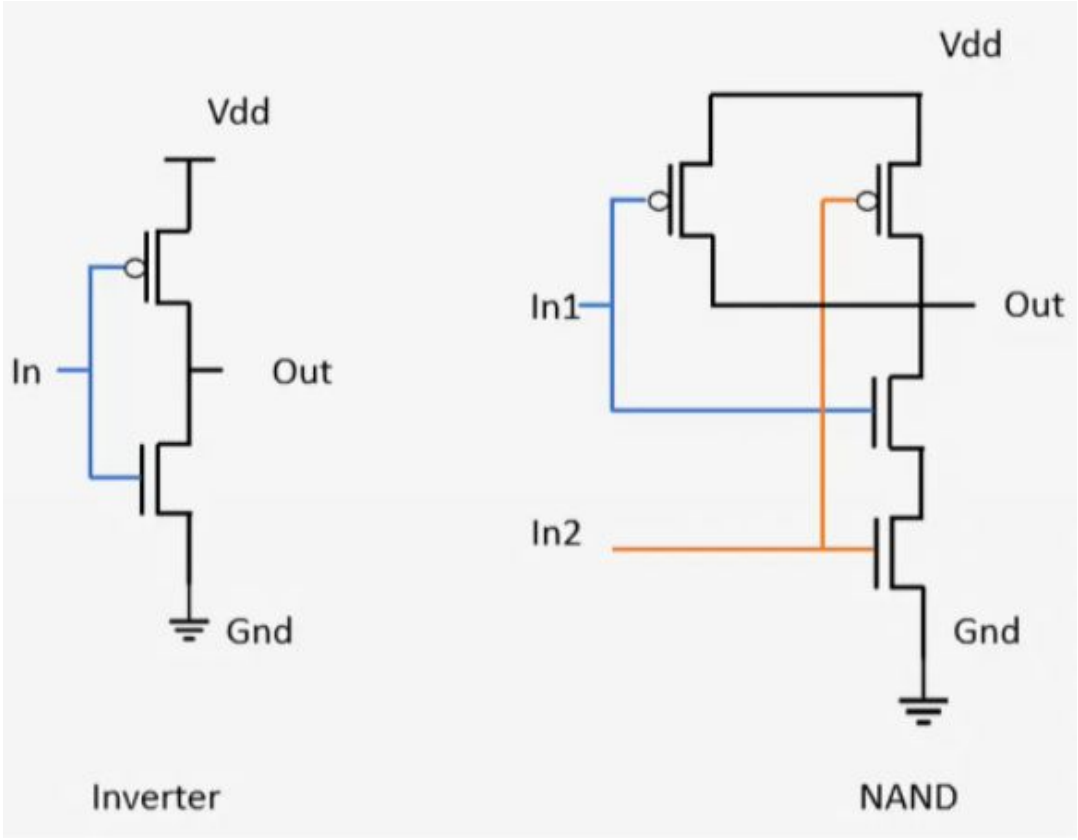


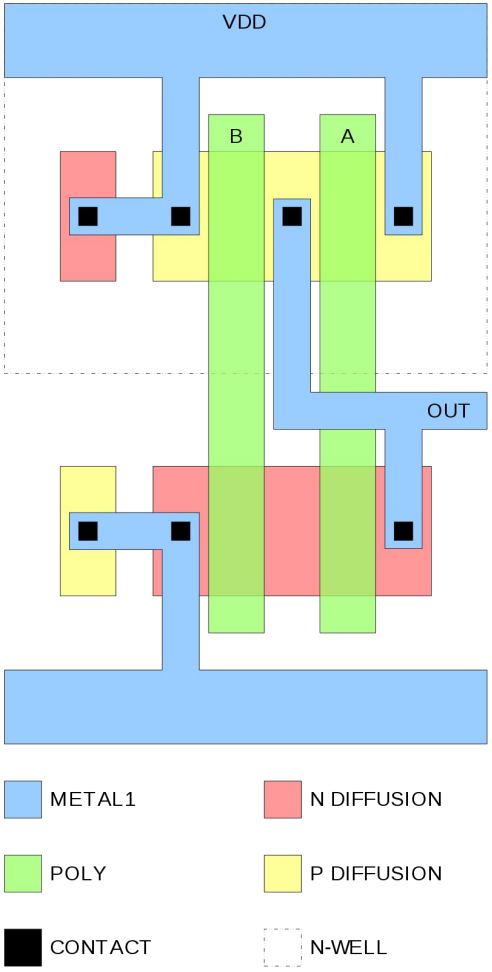
MOSFET - Metal–Oxide Semiconductor Field-Effect Transistor



CMOS - Complementary Metal Oxide Semiconductors

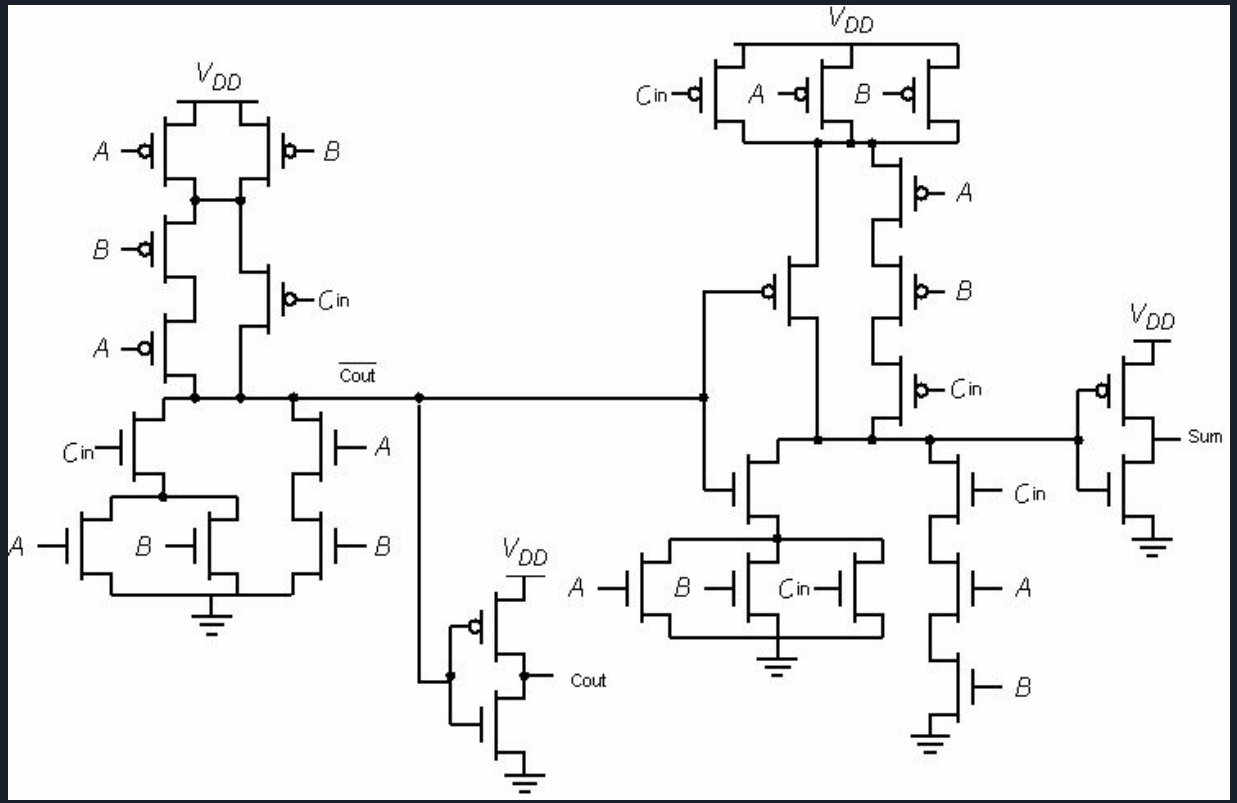




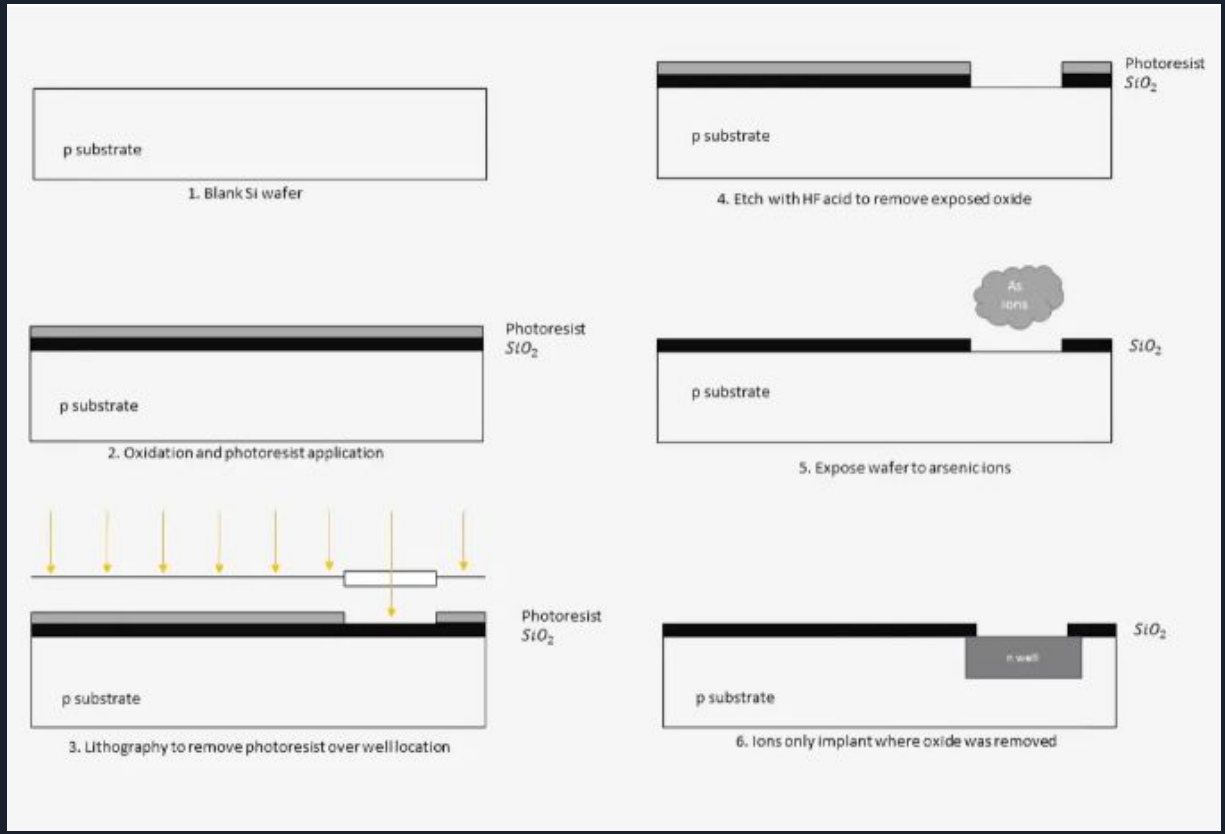


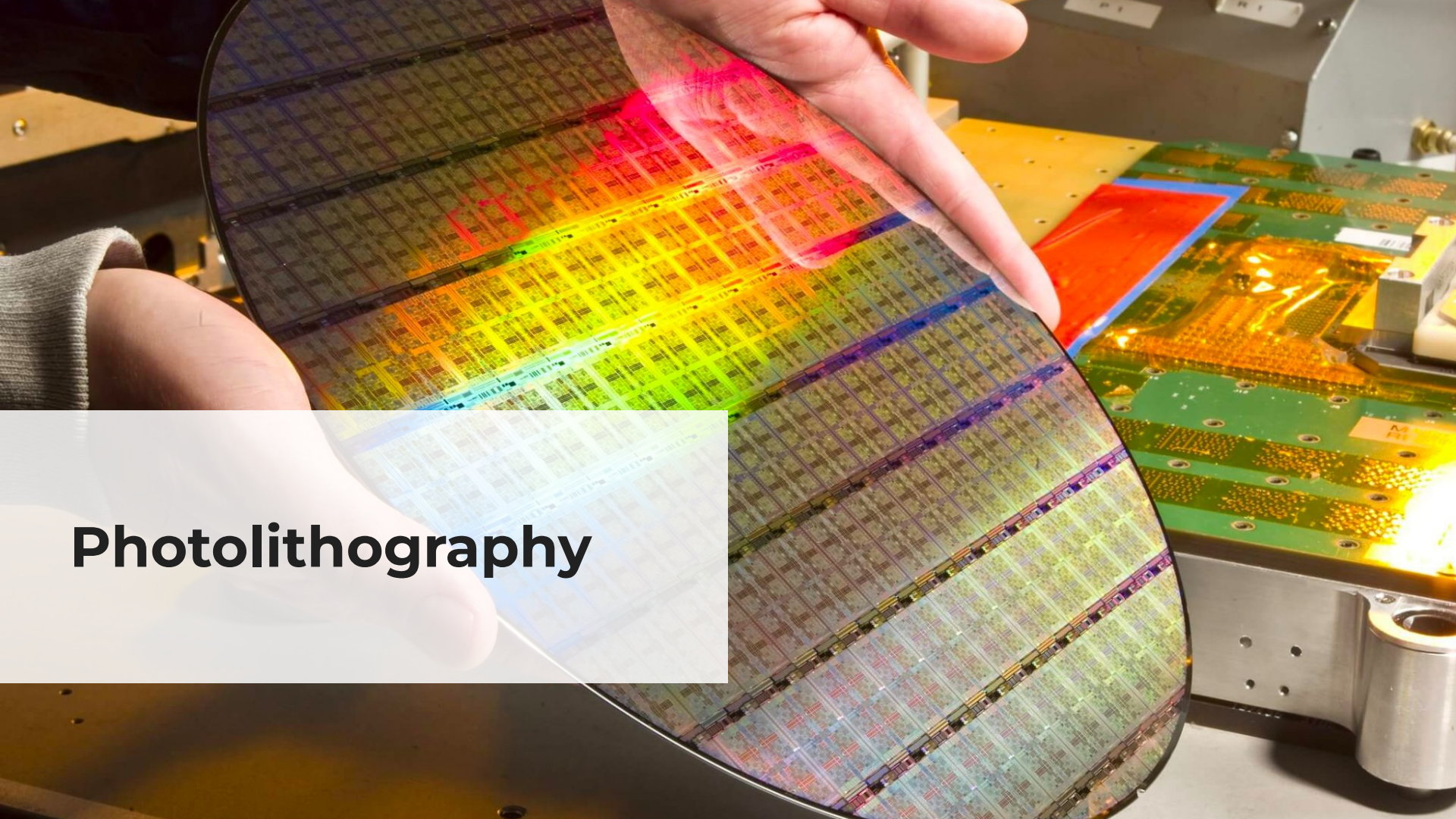
CMOS Logic Gates

CMOS Full adder



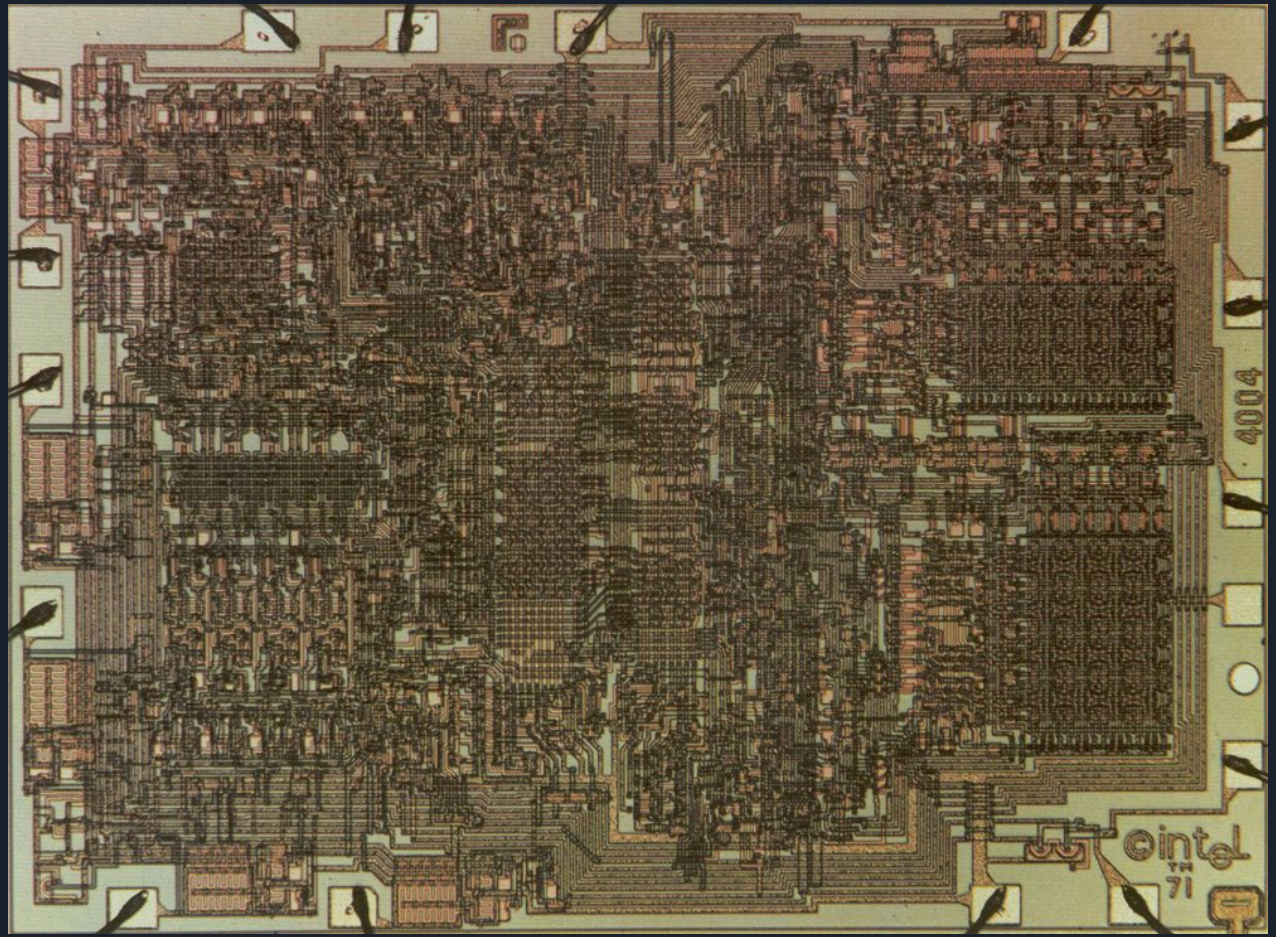
Photolithography



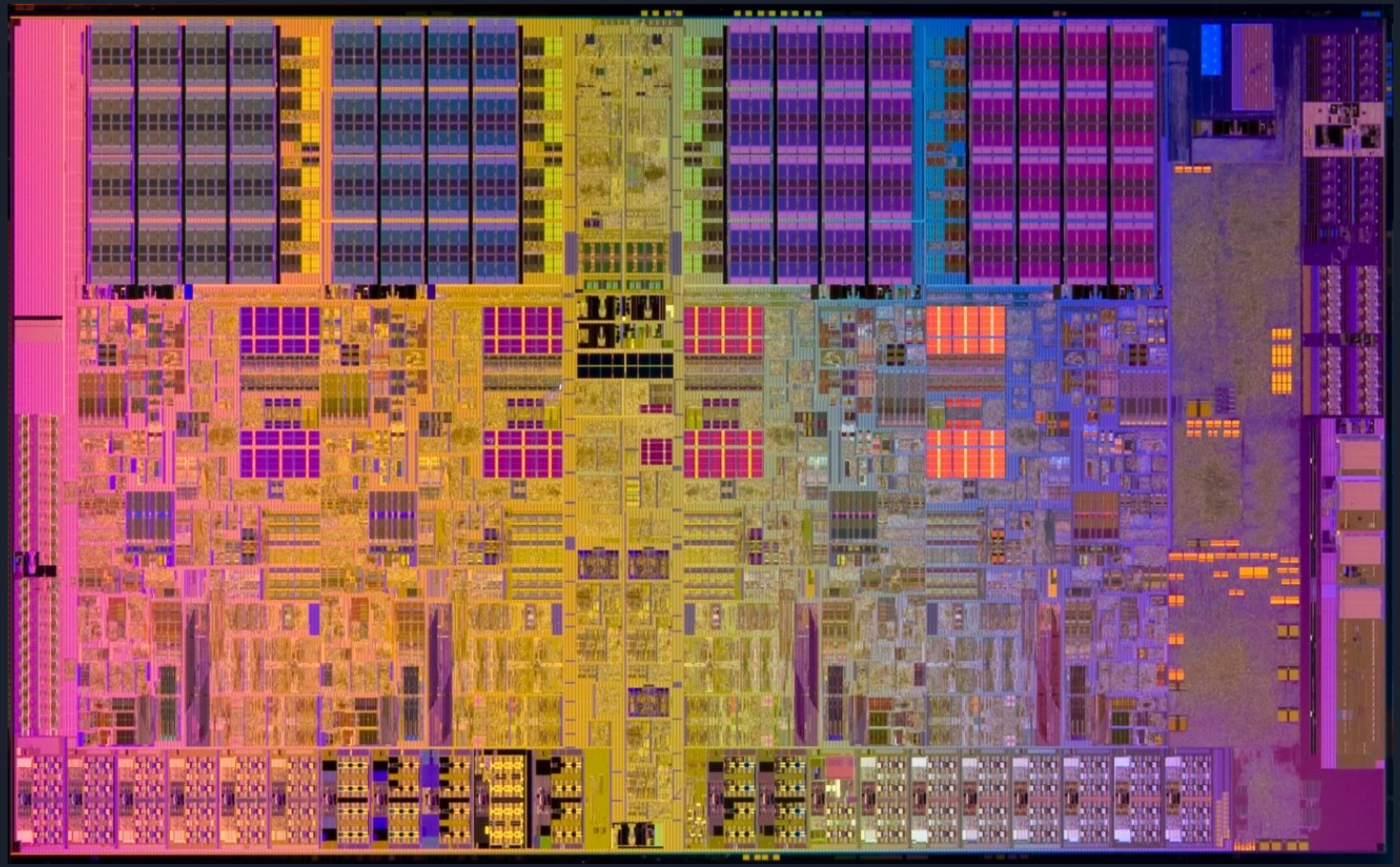


Photolithography

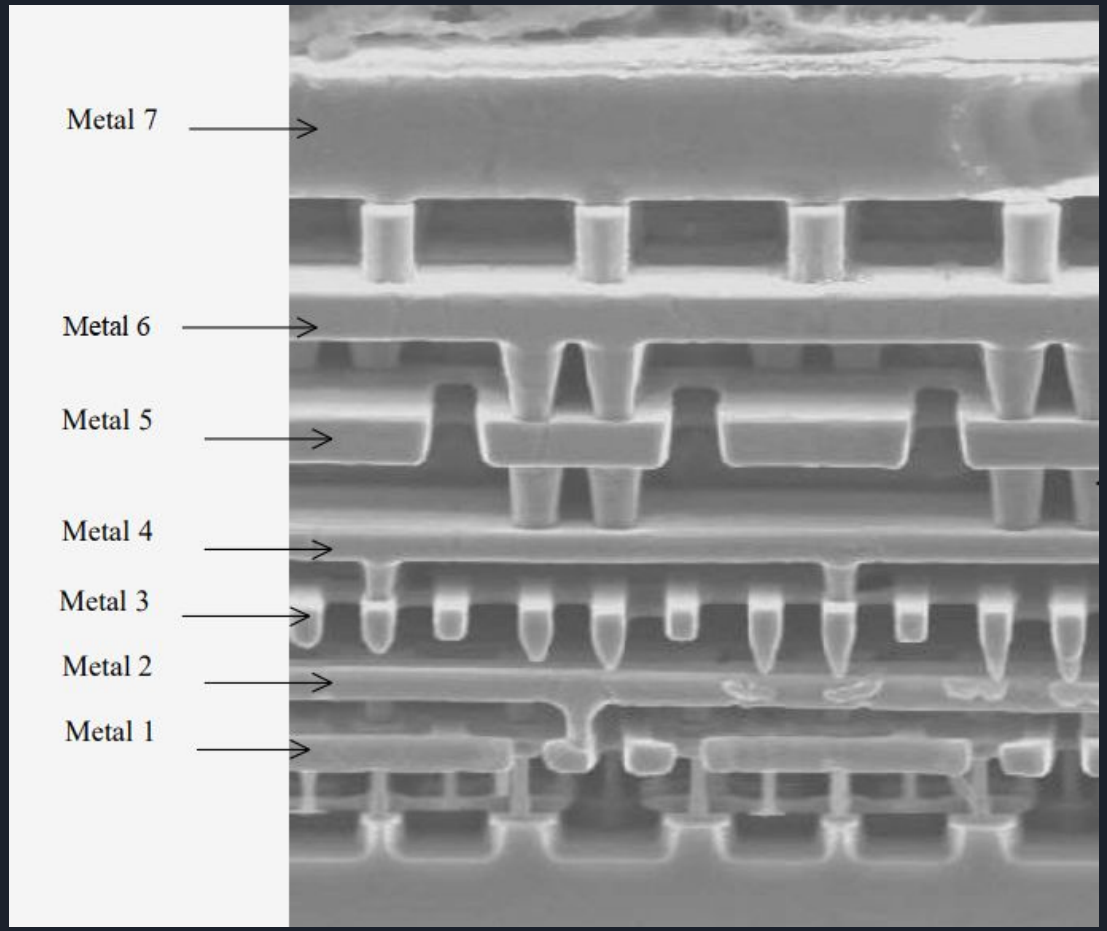
Scaling up!



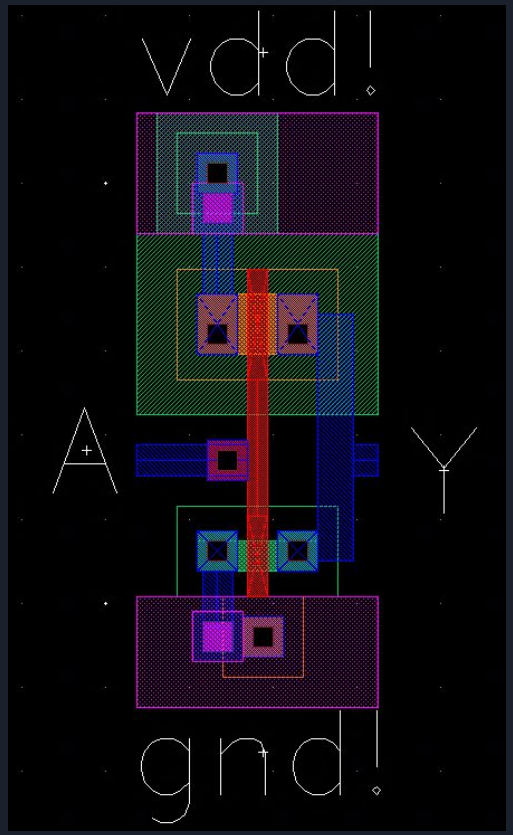
Scaling up!

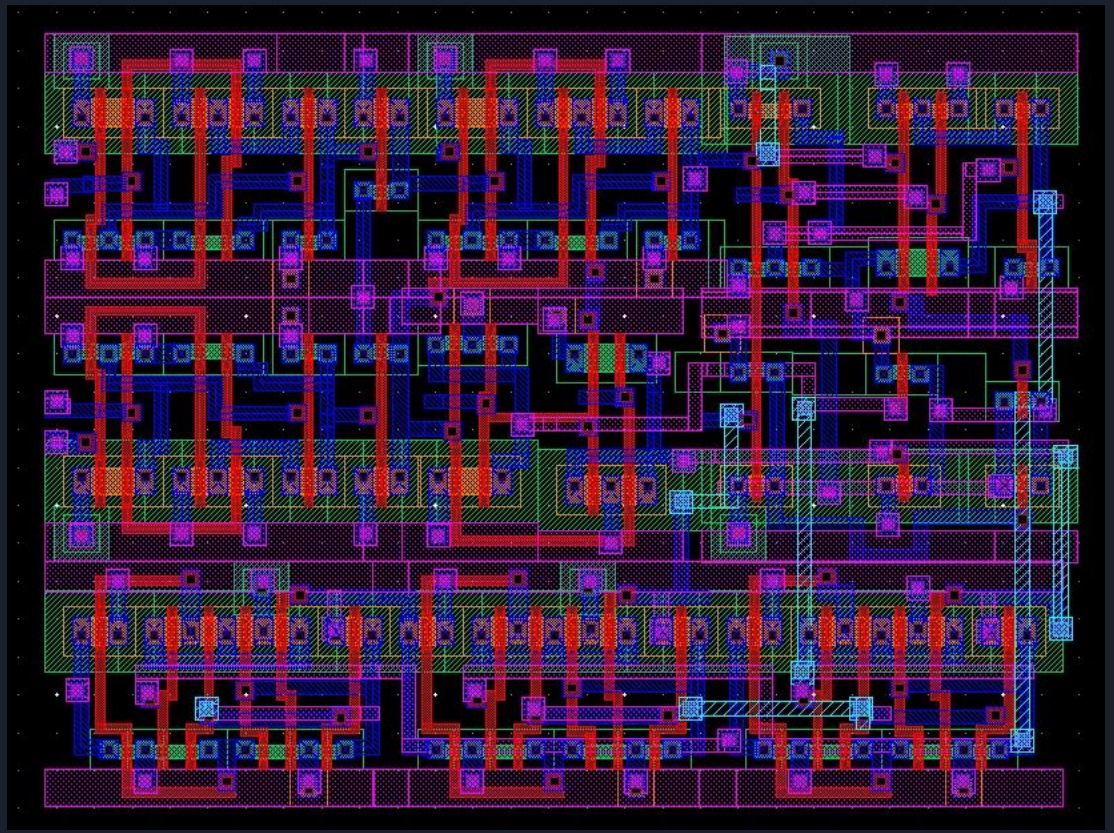


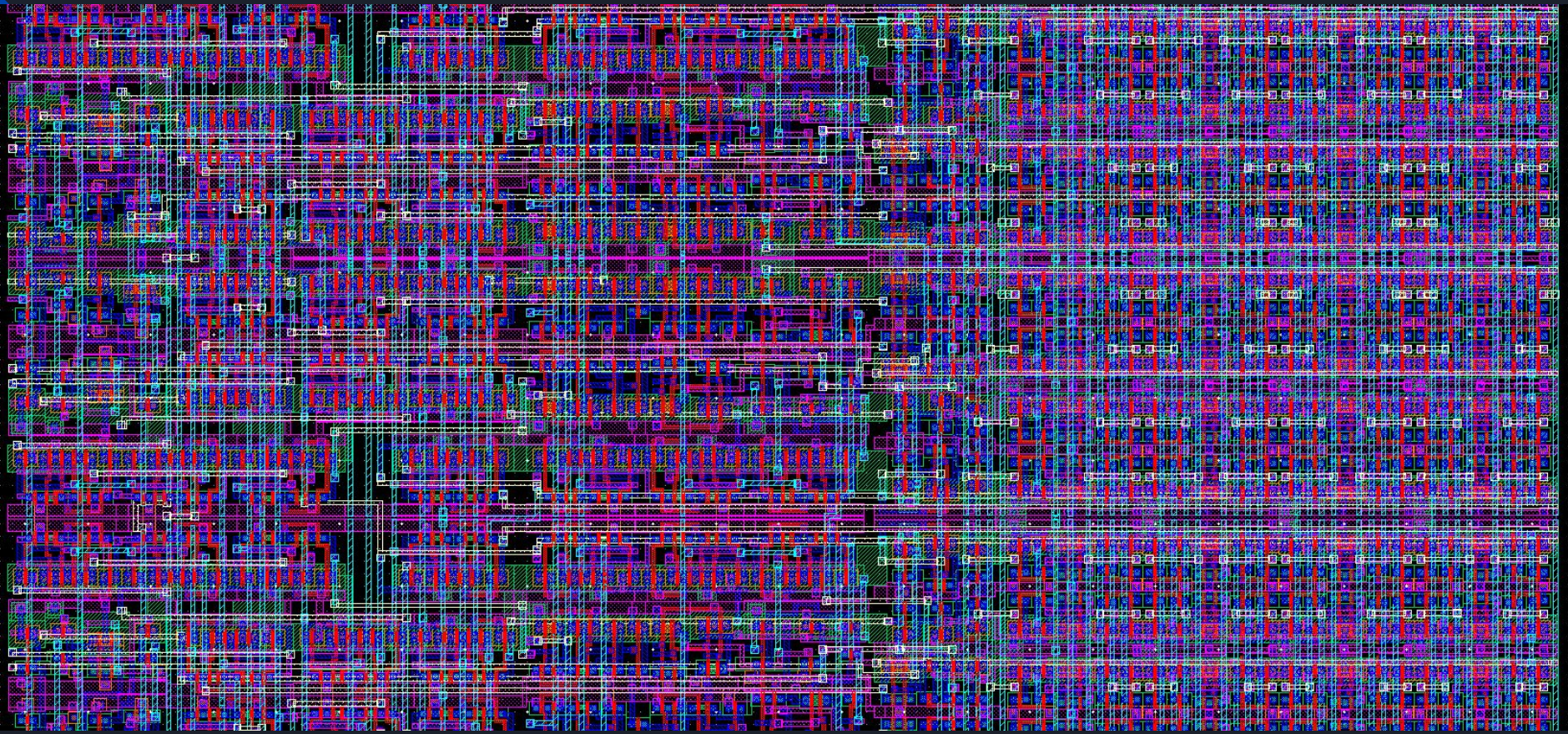
Supporting Architecture



Design Programs







Summary

- Whistle stop tour of fundamental computer architecture
- Designed to provide a base for others to build on in their seminars
- We have shown that data is represented by binary numbers
- Computers have an instruction set that can be used to create programs that perform operations on binary data
- Many instructions can be represented using boolean functions
- Boolean functions can be represented using logic circuit comprised of logic gates
- Logic circuits can be made from transistors
- Transistors and supporting metal connections can be printed in layers onto silicon to make a ‘chip’
- Chip form all of the components that make up a modern computer including the CPU, GPU and many other parts