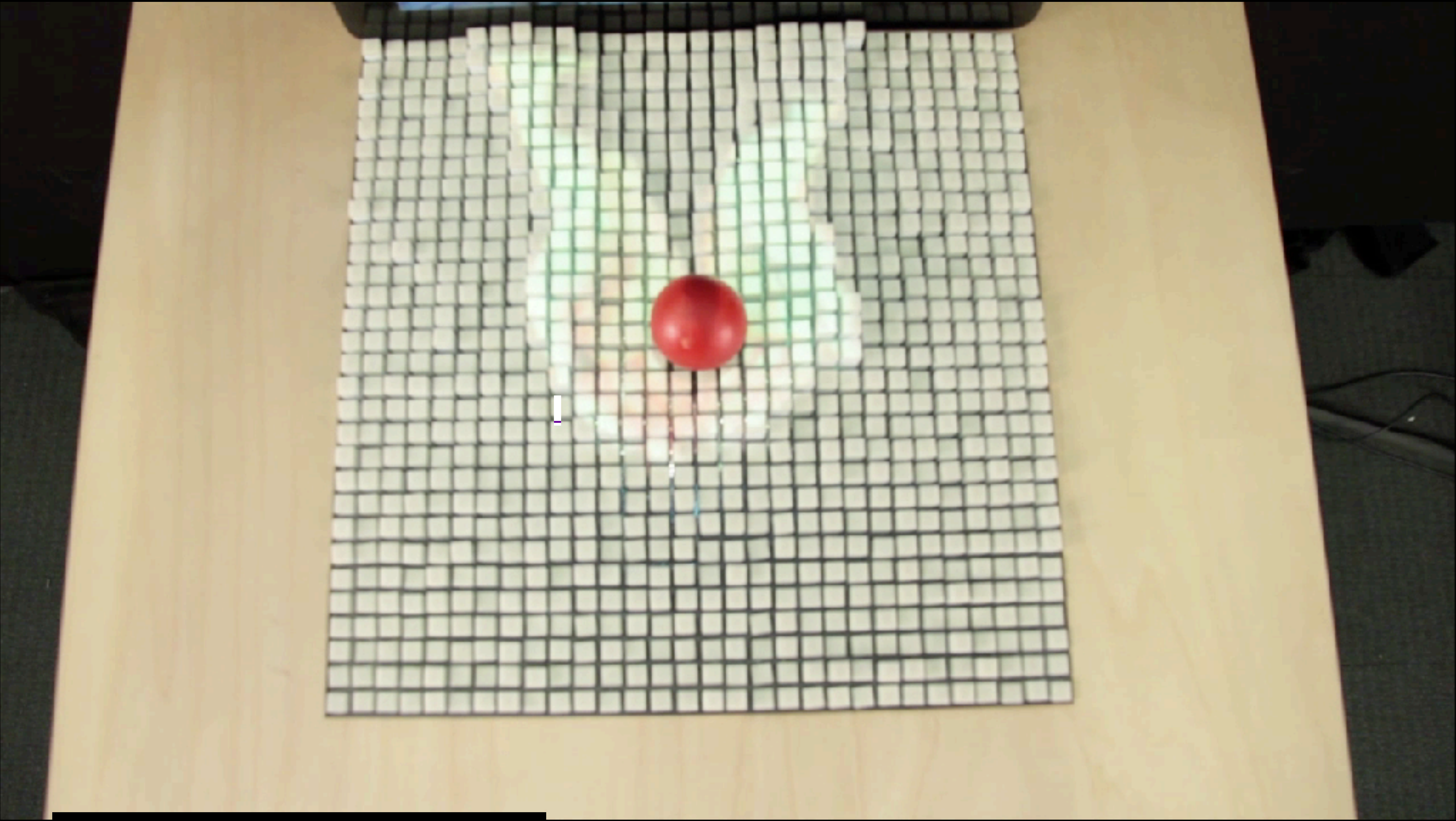
Bits and Atoms

Luke Franzke - 2017

Creative Code

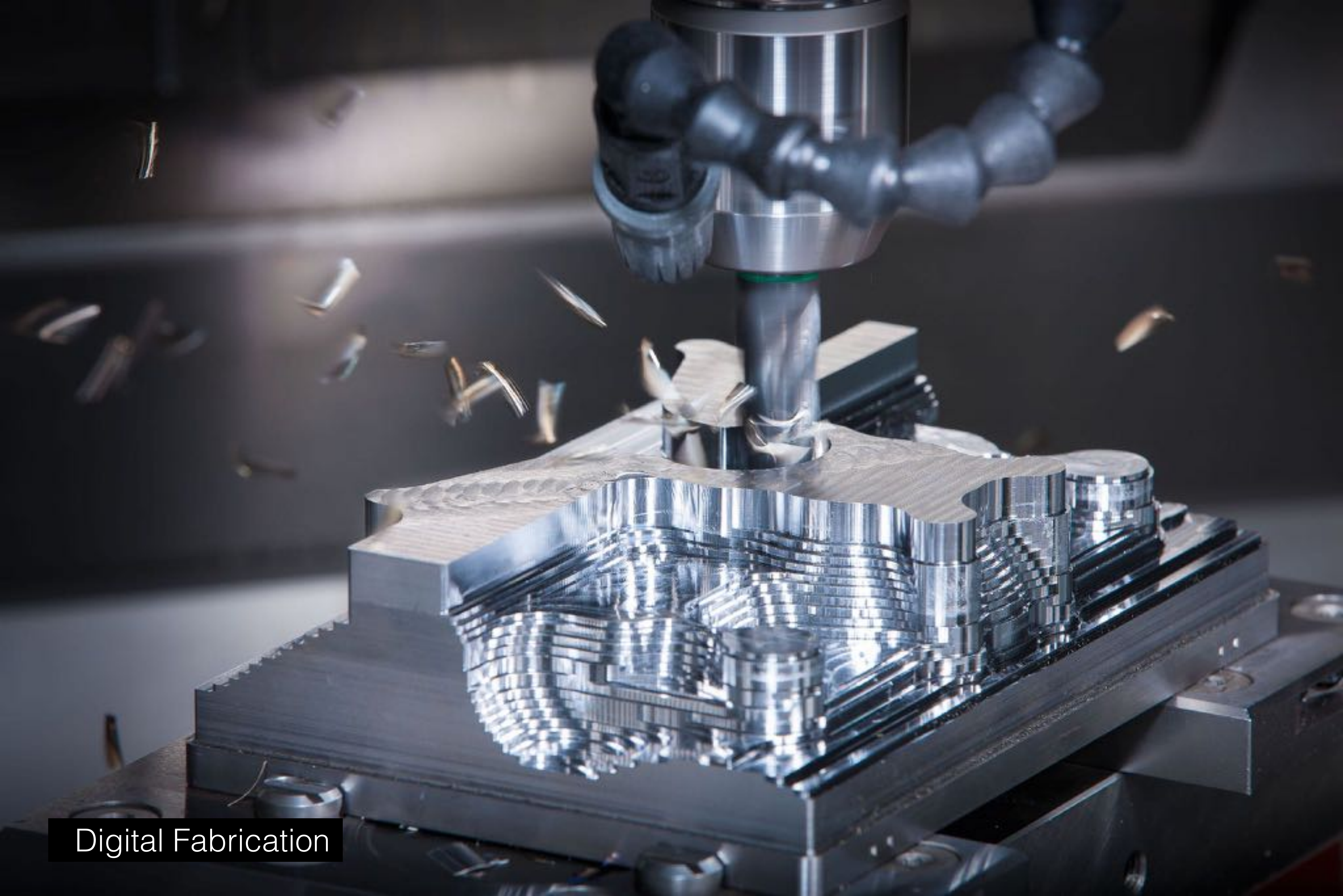


Human Computer Interaction

Interactive Experience

Adrien Mondot and Claire Bardainne

The Arts



Digital Fabrication


Gramazio & Kohler
Digital Fabrication

Gramazio & Kohler

Digital Fabrication


Michael Hansmeyer
Generative Design

Generative Design


Heydar Aliyev Centre, Zaha Hadid
Parametric Design

The impact of code in design

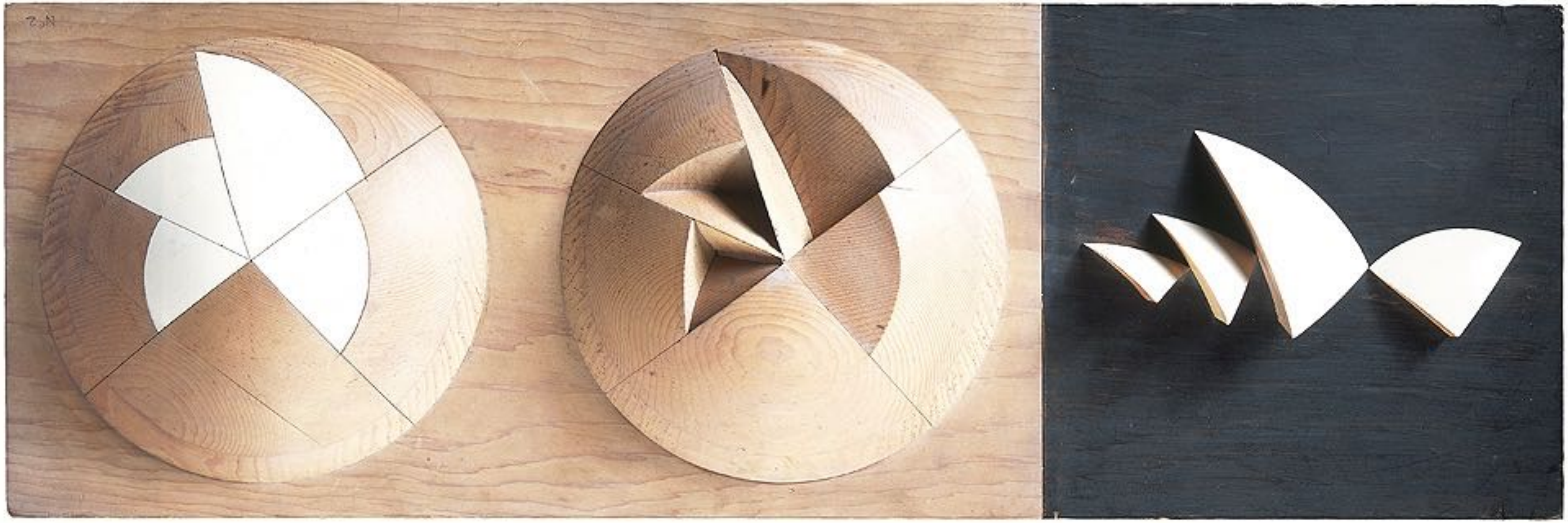
Before Parametric Design

Reproducible curves

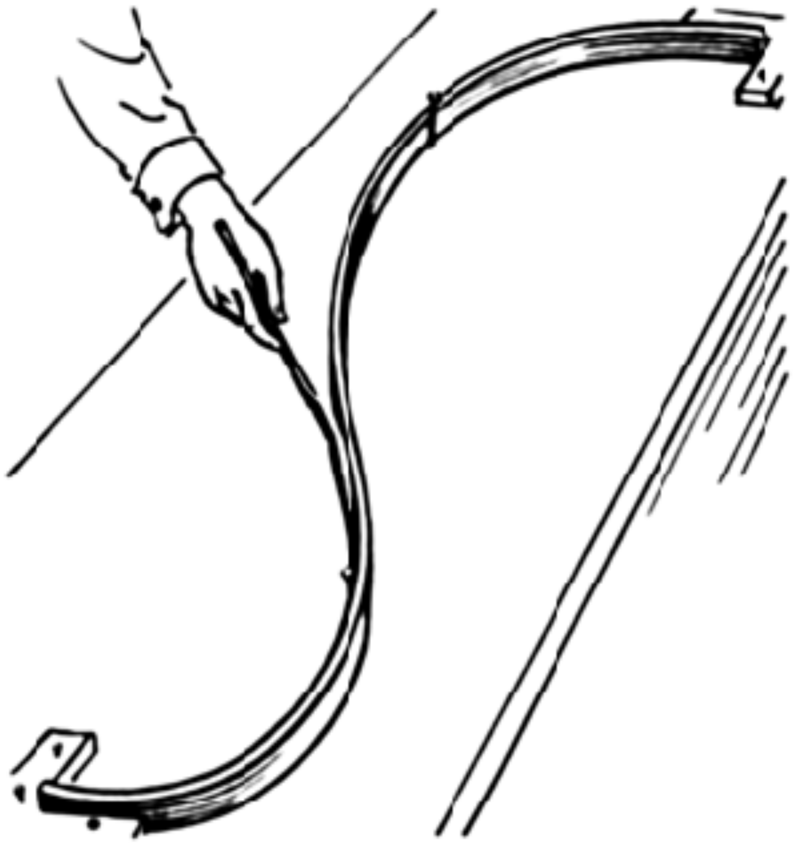

Before Parametric Design
Splines: Reproducing curves at different scales

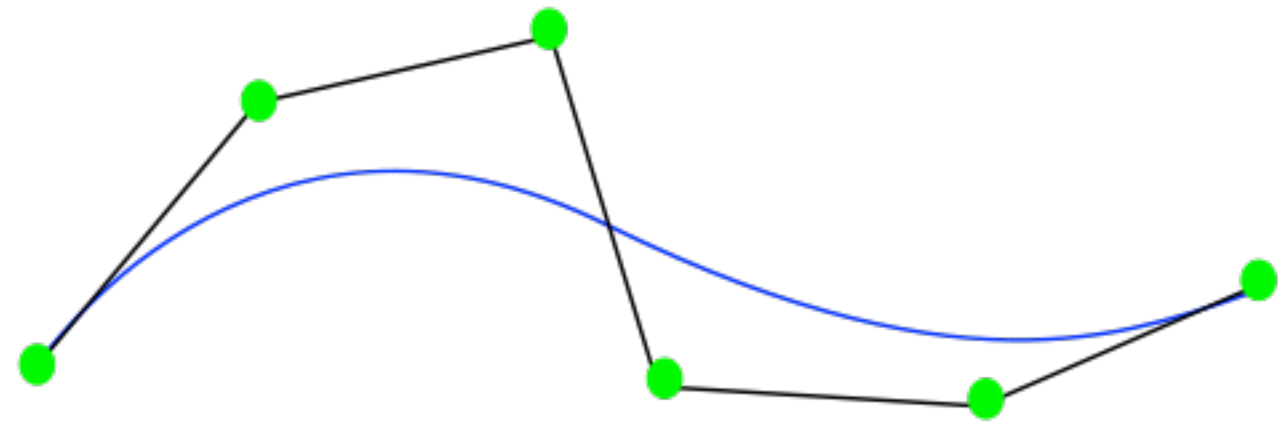
Before Parametric Design

Jørn Utzon sketches



Before Parametric Design





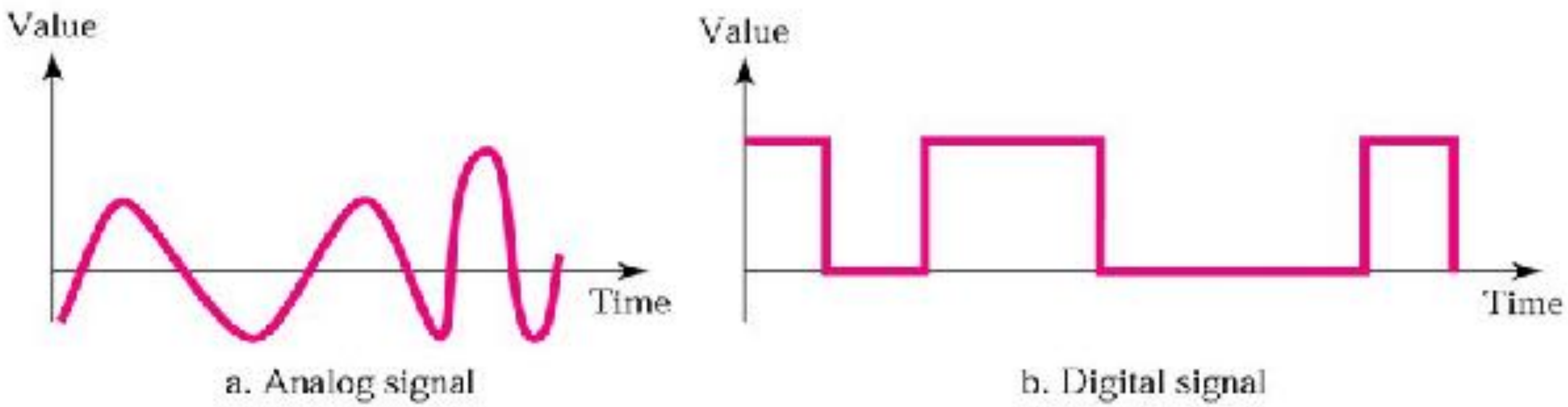
Splines: Analog to Mathematical Model (Bezier Curve)

Getting to the Basics

- What is Digital?
- What is Analog?

Analog and Digital Signals

- An analog signal is a continuous wave form that changes smoothly over time
- A digital signal is discrete. It can have only a limited number of defined values, often as simple as 1 and 0

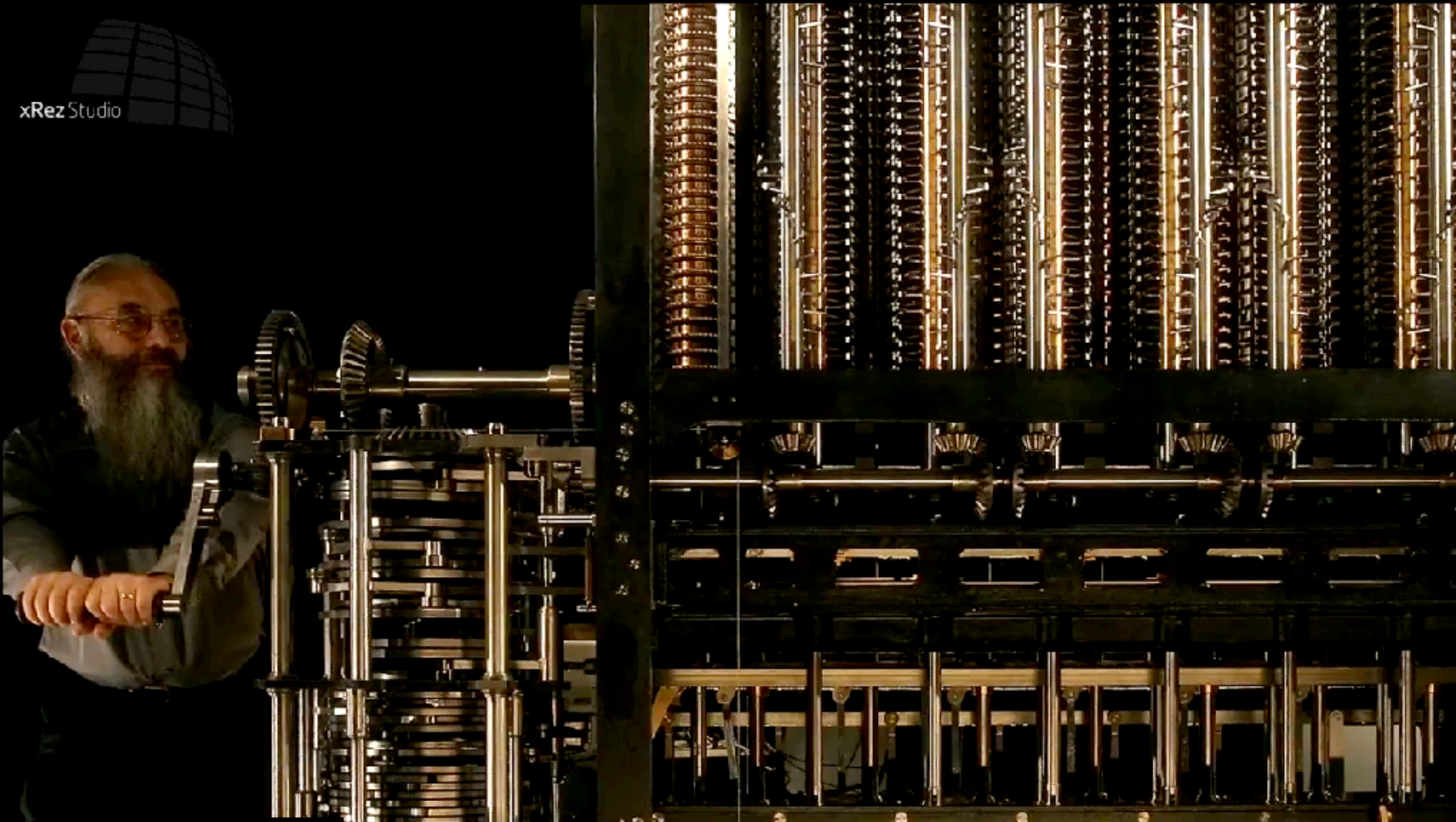


Analog vs Digital

Analog Technology

Jacquard Loom

Jacquard Loom



Babbage engine

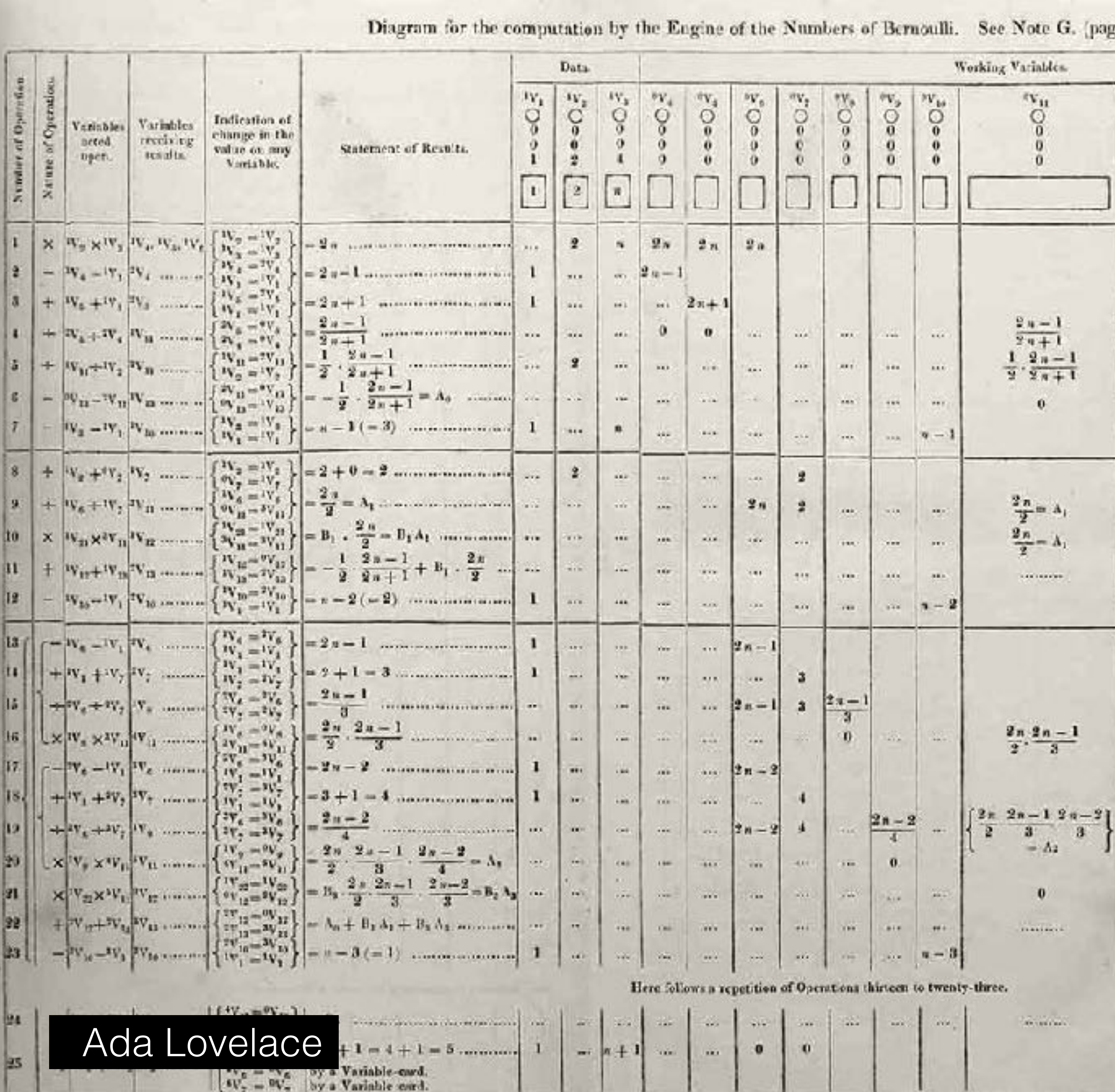


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page

Number of Operation	Nature of Operation	Statement of Results
1	×	$= 2n$
2	−	$= 2n - 1$
3	+	$= 2n + 1$
4	÷	$= \frac{2n-1}{2n+1}$
5	÷	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$
6	−	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$
7	−	$= n - 1 \ (= 3)$
8	+	$= 2 + 0 = 2$
9	÷	$= \frac{2n}{2} = A_1$
10	×	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$
11	+	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$
12	−	$= n - 2 \ (= 2)$
13	−	$= 2n - 1$
14	+	$= 2 + 1 = 3$
15	÷	$= \frac{2n-1}{3}$
16	×	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$
17	−	$= 2n - 2$
18	+	$= 3 + 1 = 4$
19	÷	$= \frac{2n-2}{4}$
20	×	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$
21	×	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$
22	+	$= A_0 + B_1 A_1 + B_3 A_3$
23	−	$= n - 3 \ (= 1)$

Here follows a repetition of Operations thirteen to twenty-three.

Number of Operation	Nature of Operation	Statement of Results
24		
25		$+1 = 4 + 1 = 5$ by a Variable-card. by a Variable-card.

Ada Lovelace

“

[The Analytical Engine] might act upon other things besides *number*, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine...Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.

”

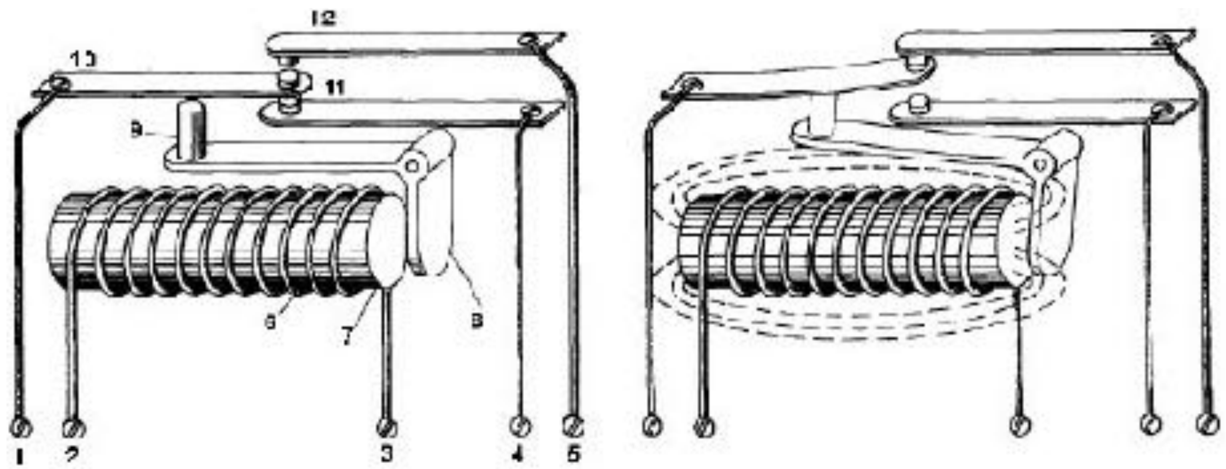
Ada Lovelace

Electricity!

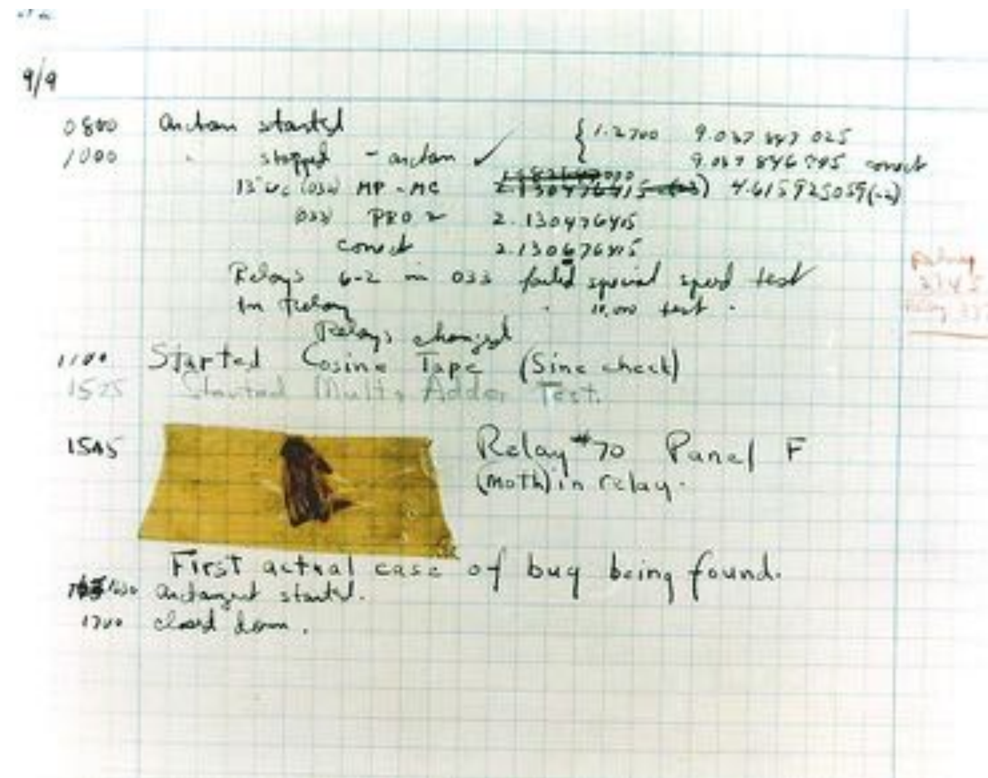
20th century and the first electronic computers

Electricity!

20th century and the first electronic computers

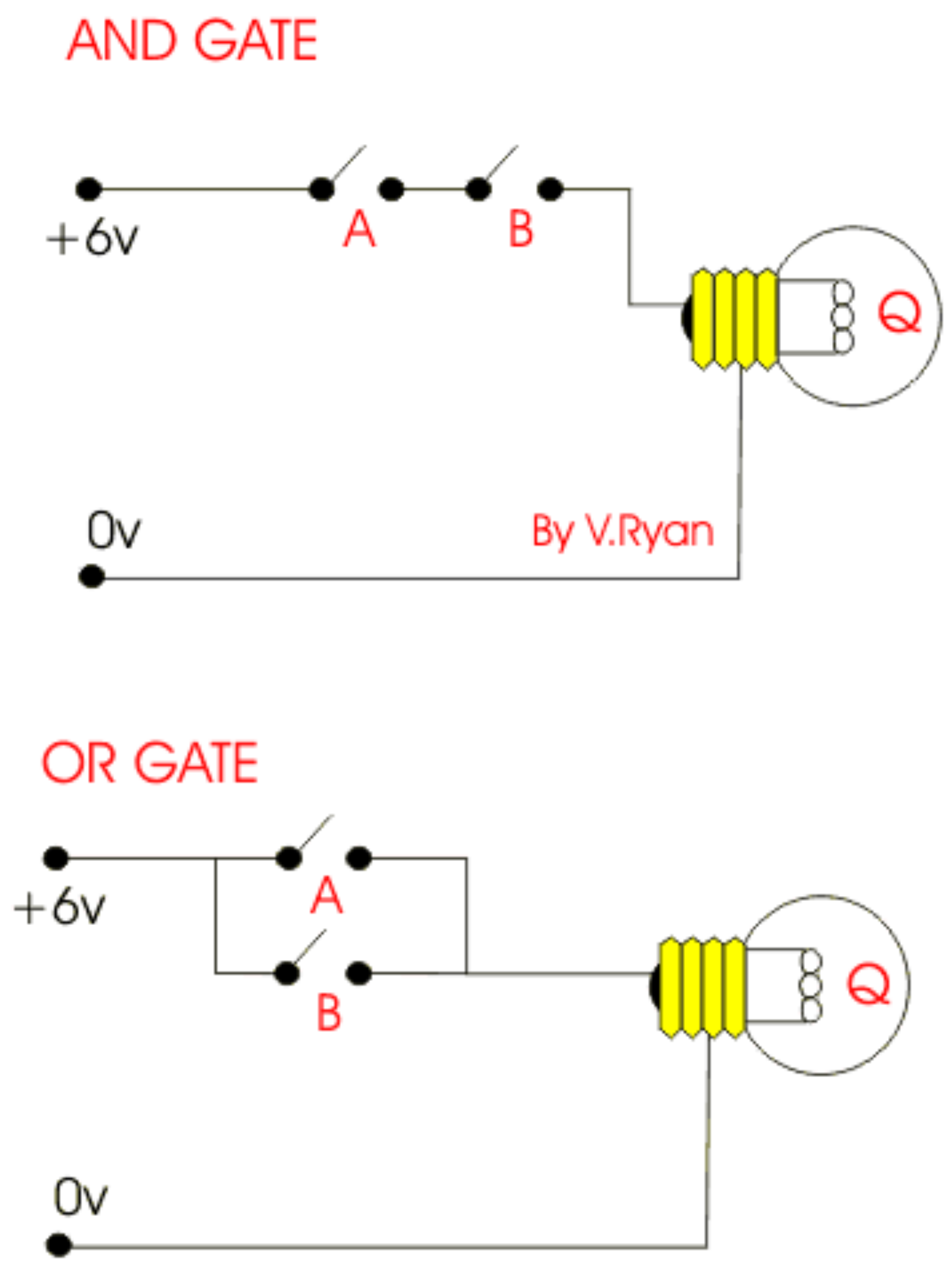


Relays

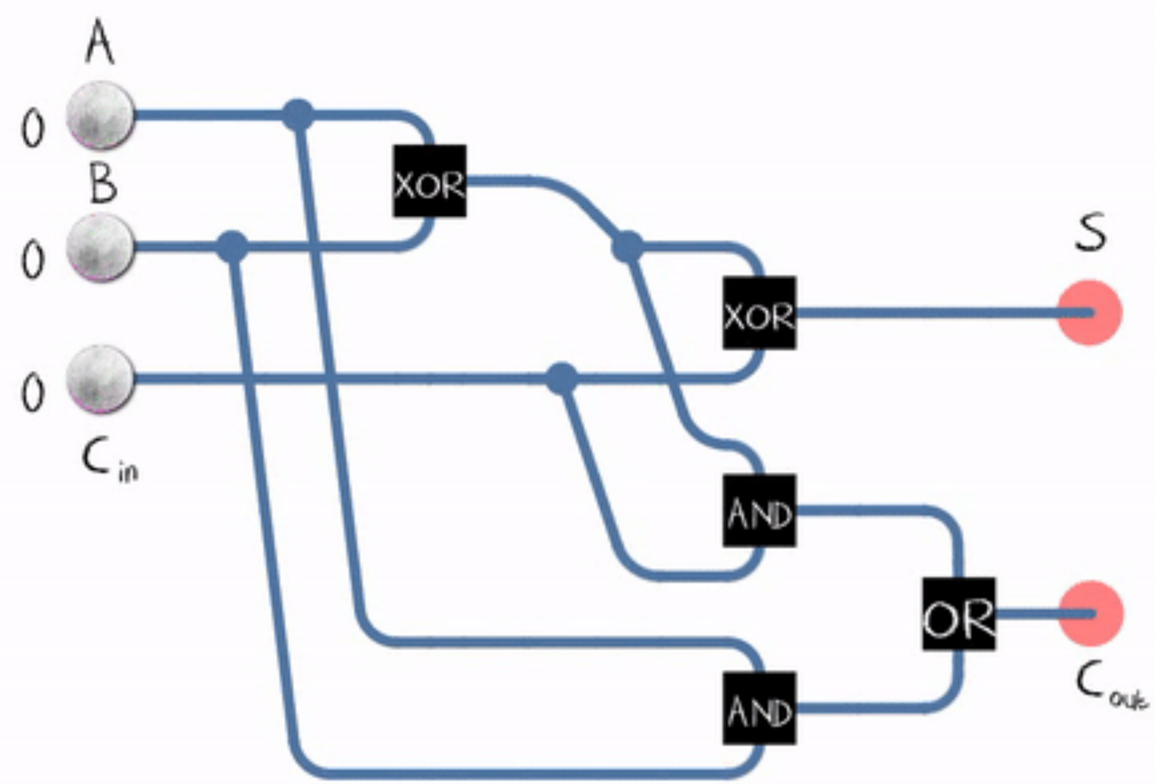


The first computer bugs

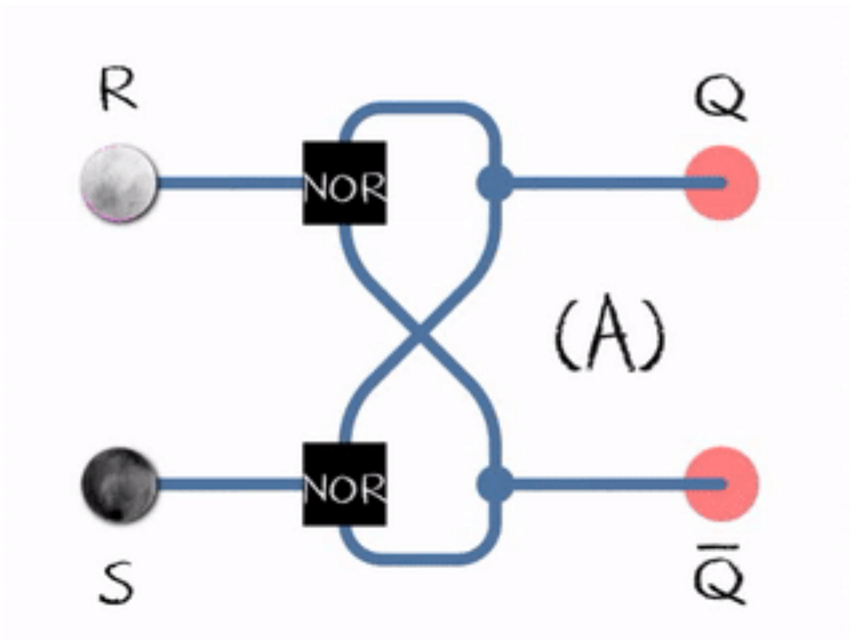
Electromechanical Computers



Logic with Switches (boolean logic and logic gates)



Logic Gates into Binary adders (and Calculators)

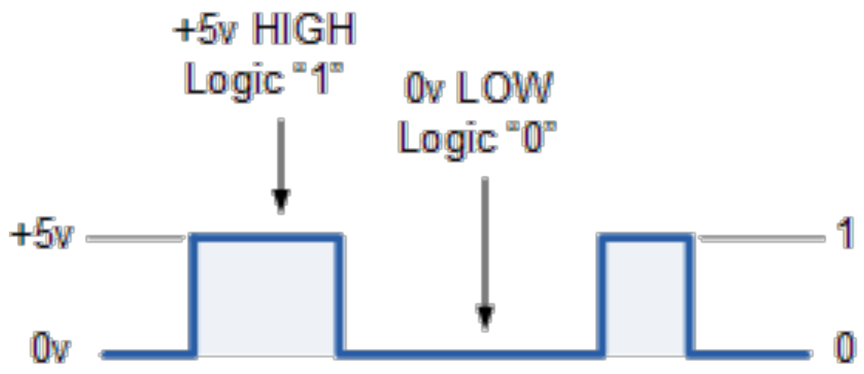


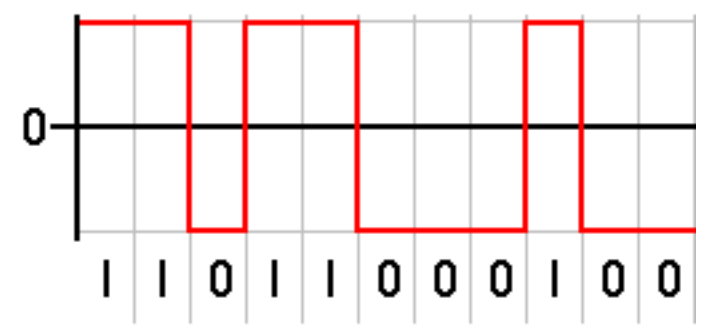
Volatile memory

non volatile memory

Data Storage and Memory

Iphone 7 A10 CPU: 3.3 Billion Transistors





Electricity into Binary

What are Bits all about?

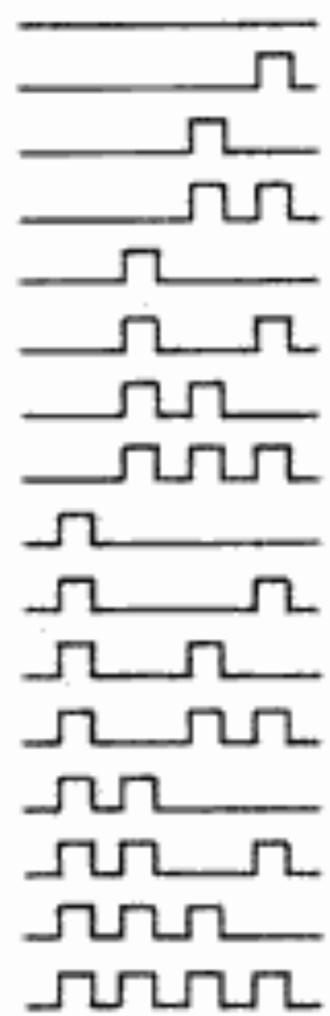
STATE A	STATE B
1	0
TRUE	FALSE
HIGH	LOW
ON	OFF
OPEN	CLOSED
ACTIVE	INACTIVE
UP	DOWN

States

DECIMAL NUMBER	BINARY EQUIVALENT 2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

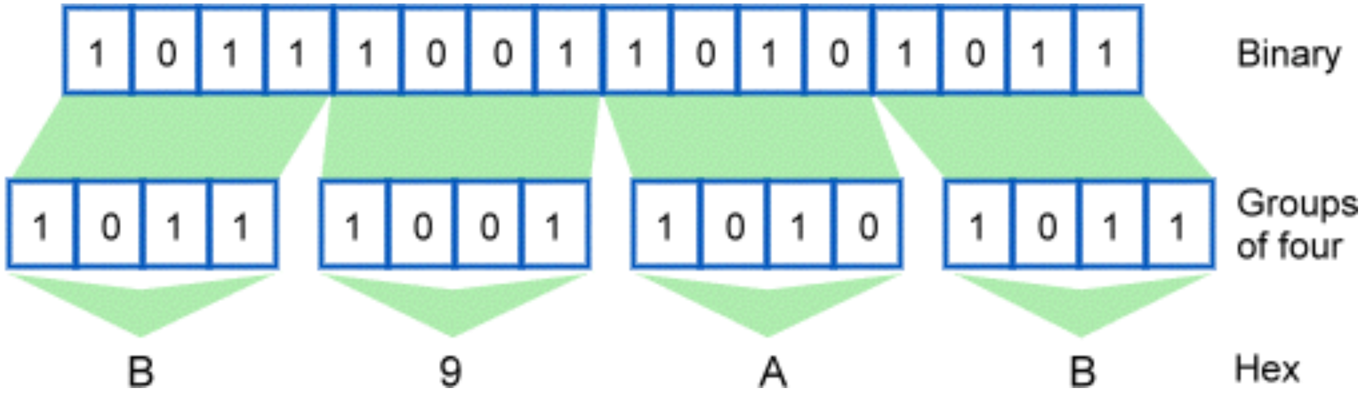
PULSE - CODE WAVEFORMS

2^3 2^2 2^1 2^0



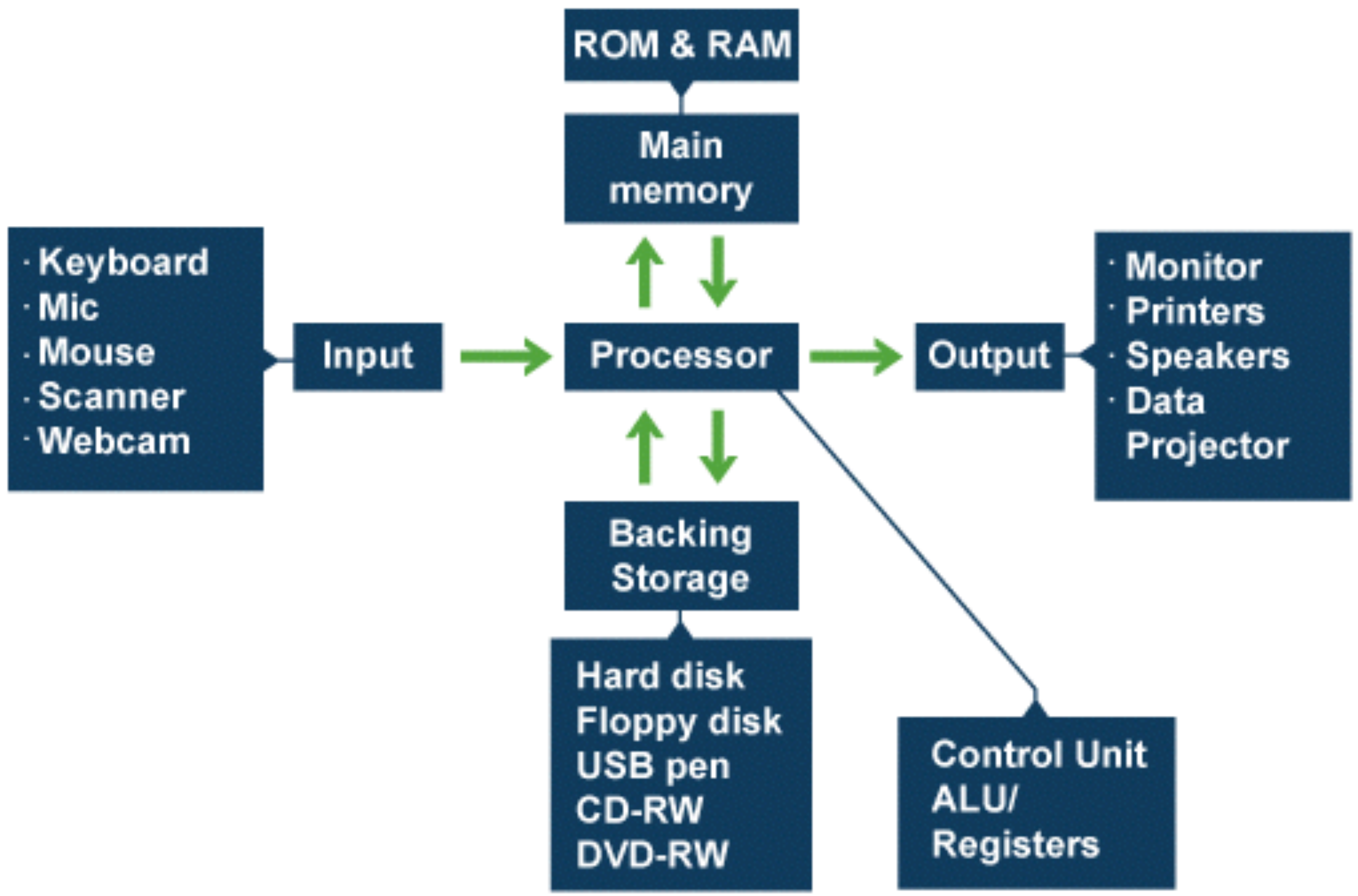
Counting in Binary

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F



Hexadecimal

Hexadecimal Colour



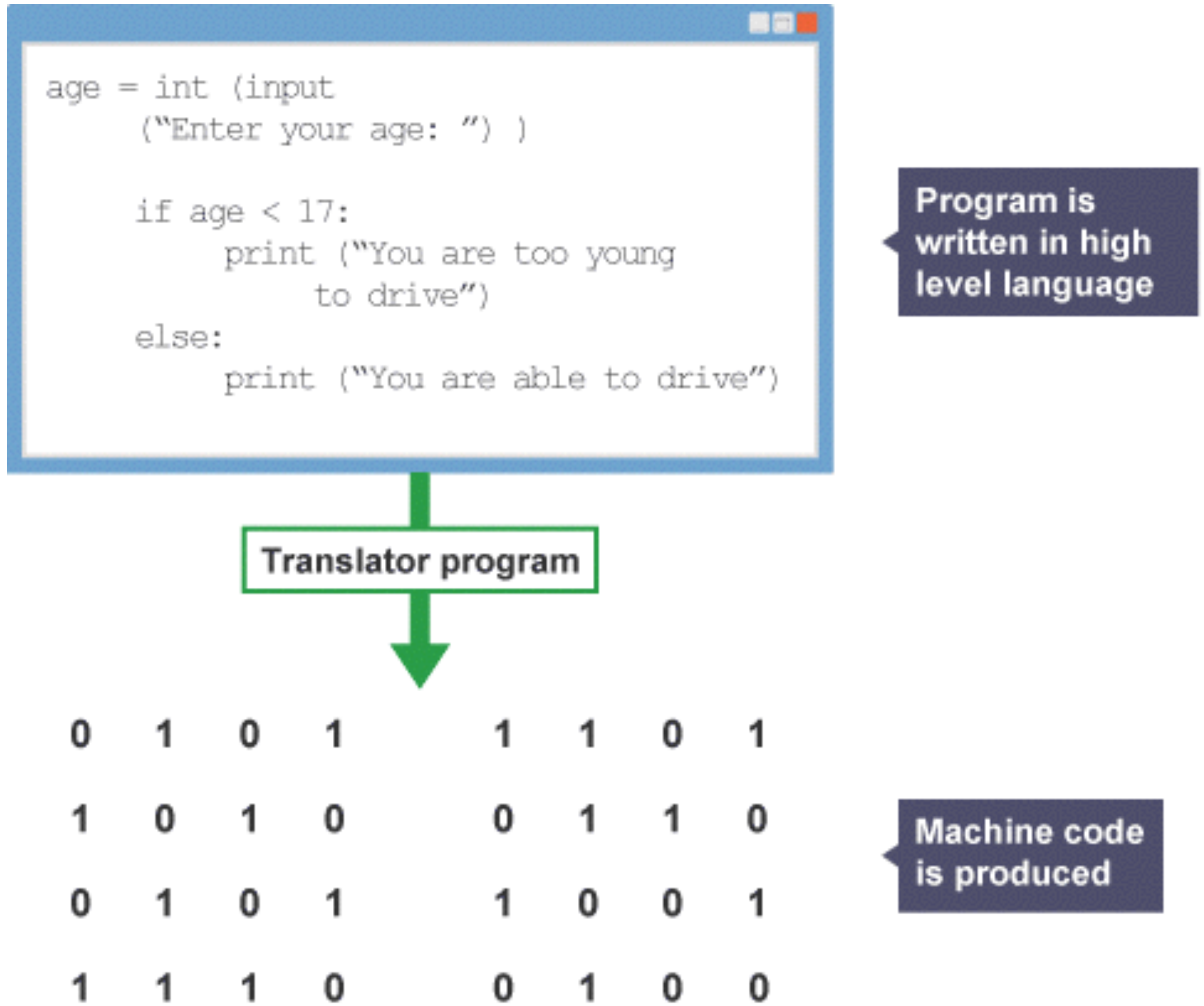
Fitting it all together

```
def add5(x):
   return x+5

def dotwrite(ast):
   nodename = getNodename()
   label=symbol.sym_name.get(int(ast[0]),ast[0])
   print '    %s [label="%s' % (nodename, label),
   if isinstance(ast[1], str):
      if ast[1].strip():
         print '= %s"];' % ast[1]
      else:
         print '"]'
    else:
       print '"];'
       children = []
       for  in n, childenumerate(ast[1:]):
          children.append(dotwrite(child))
       print ,'    %s -> {' % nodename
       for  in :namechildren
          print '%s' % name,
```

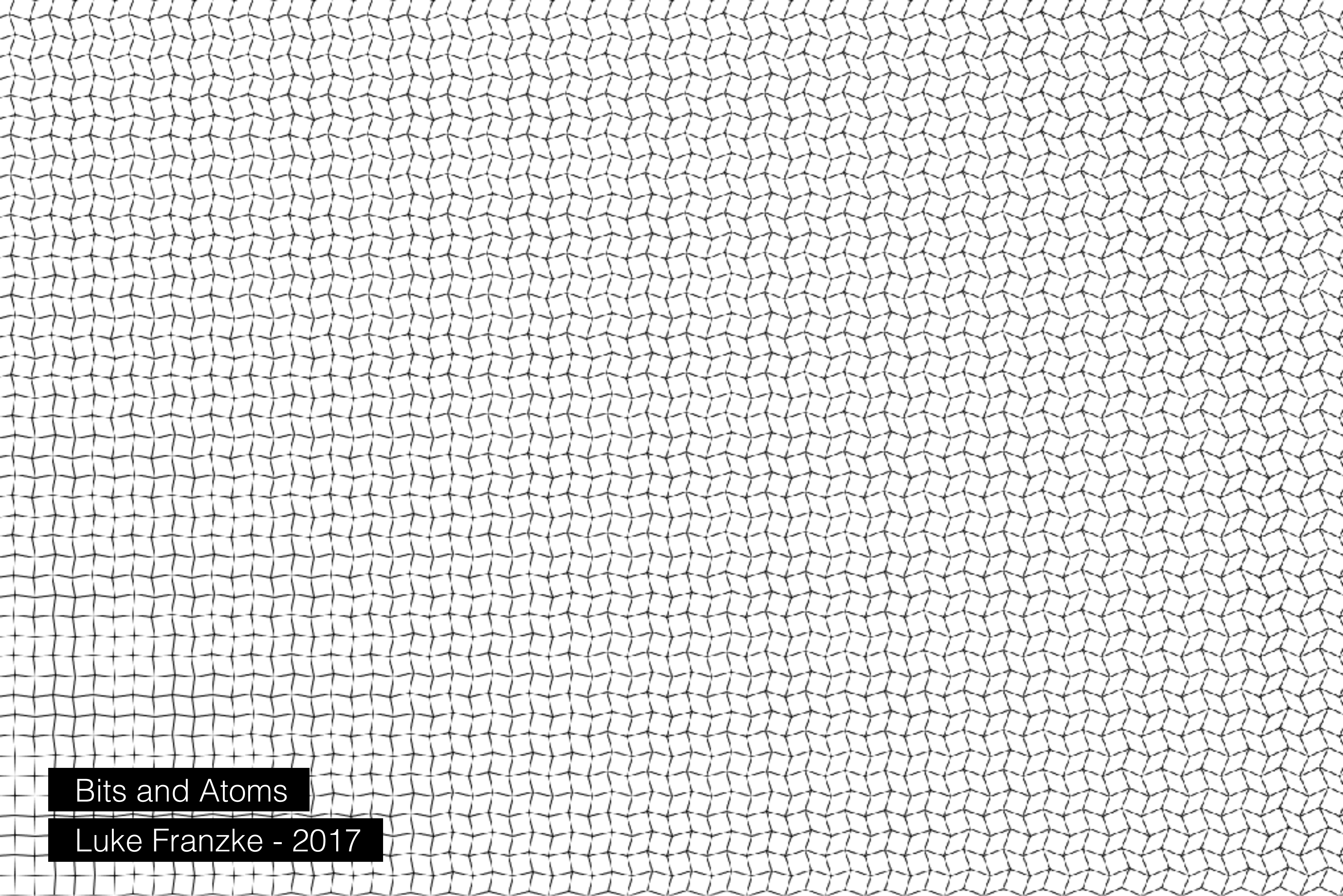
Coding Languages

Syntax and instructions



Coding Languages

Compiling and Interpreting


Bits and Atoms
Luke Franzke - 2017