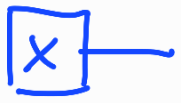
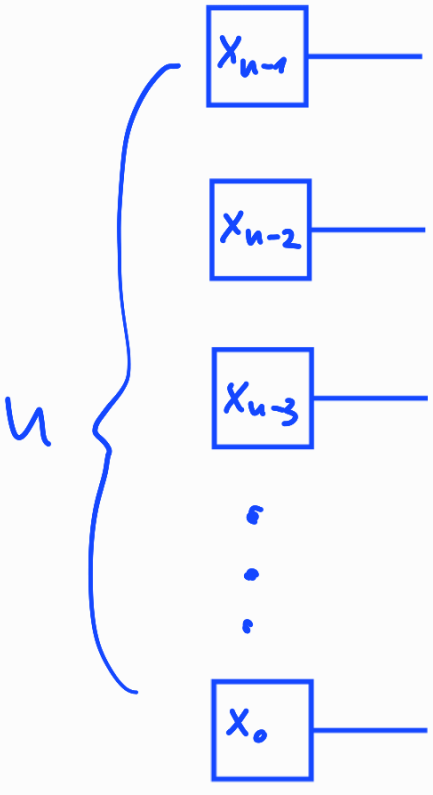
Classical bits, gates, circuit circuits

classical bit $\hat{=}$ classical 2-state system:



$x \in \{0,1\} = \mathbb{Z}_2$

n-bit register:



state:

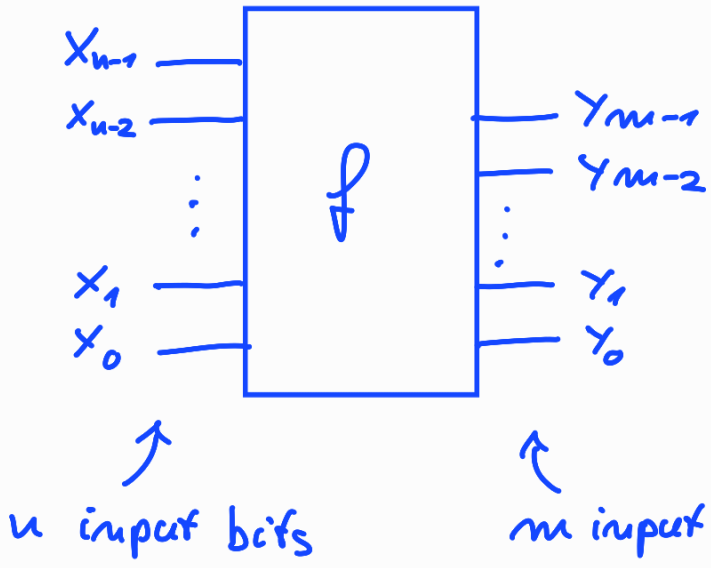
$X = (x_{n-1}, x_{n-2}, \dots, x_0) \in \mathbb{Z}_2^n$

n bit binary number:

$00\cdots00 = 0$
$00\cdots01 = 1$
$00\cdots10 = 2$
$00\cdots11 = 3$
$\vdots \qquad \vdots$
$11\cdots11 = 2^n - 1$

$\rightarrow \mathbb{Z}_2^n \hat{=} \{0, 1, 2, \dots, 2^{n-1}\}$

Logigic gate:

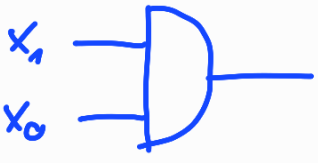


implements m-valued boolean function:

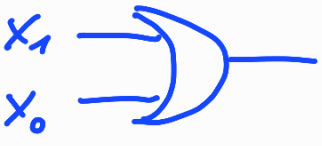
$$f : \mathbb{Z}_2^{\,n} \to \mathbb{Z}_2^{\,m}$$

$$x \mapsto y = f(x)$$

elementary gates:

AND:  x_1 AND x_0

NOT:  NOT x

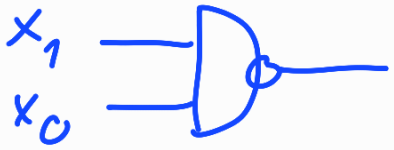
OR:  x_1 OR x_0

XOR : 

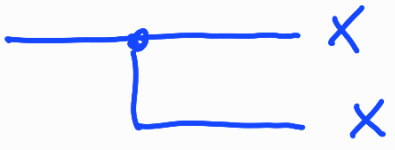
x_1 OR x_0 = $x_1 \oplus x_0$

↑

„ + mod 2 "

NAND : 

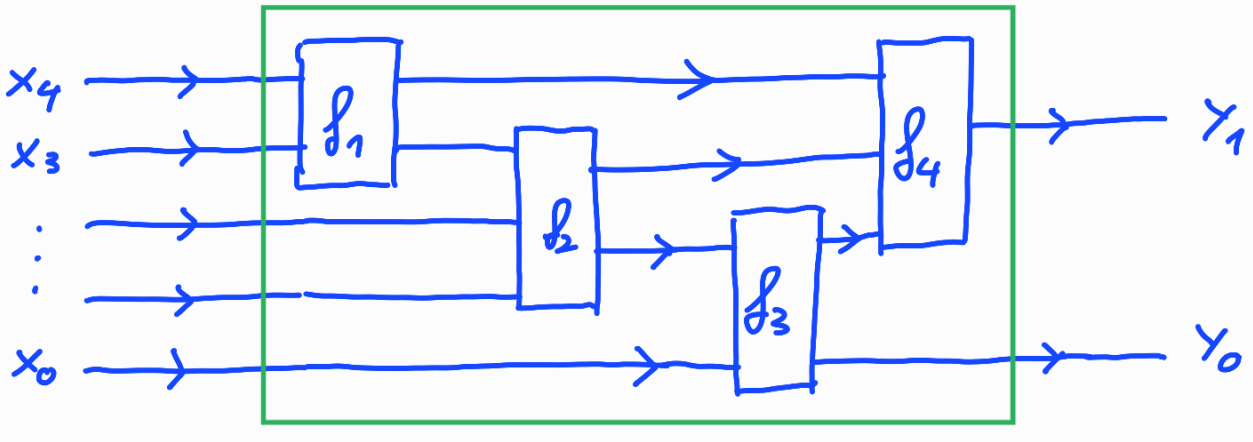
NOT (x_1 AND x_0)

BRANCH : x 

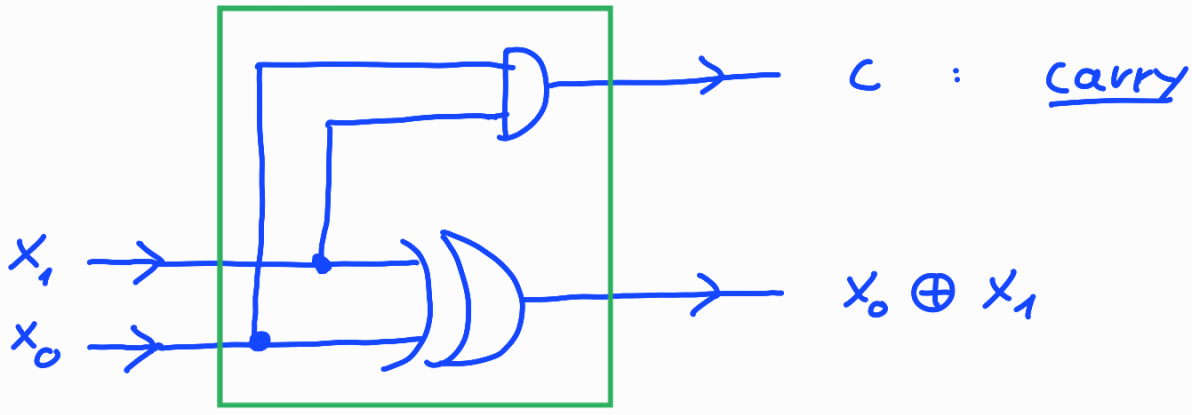
<u>circuit</u> = <u>directed, acyclic graph</u>

edge ≙ wire

knot ≙ gate



e.g. <u>half-adder</u> circuit:

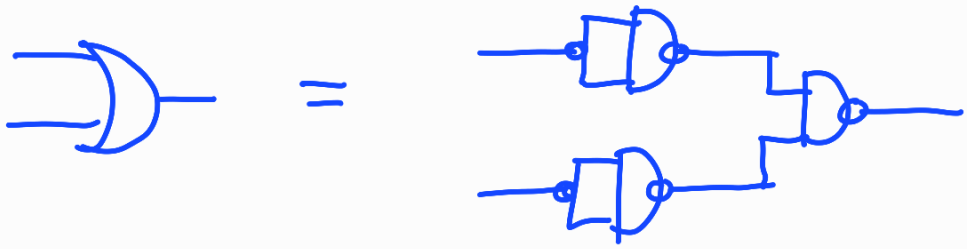


<u>Thm.</u>: Every boolean function $f: \mathbb{Z}_2^{\,n} \to \mathbb{Z}_2^{\,m}$ can be implemented by a circuit made of NAND and BRANCH gates only: $\{NAND, BRANCH\}$ is a <u>universal set</u> of logic gates.

Examples:

NOT: 

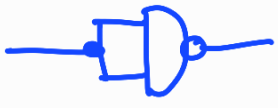
AND: 

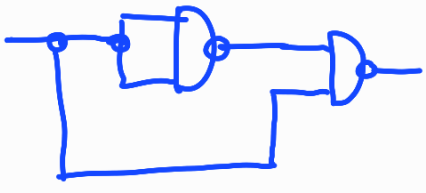
OR :  =

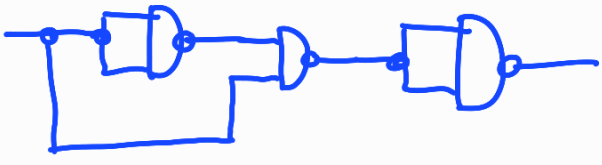
we prove Thm. by induction,

w. l. o. g. $m = 1$.

$n = 1$: $f_1 = id$: 

$f_2 = NOT$: 

$f_3 = 1$: 

$f_4 = 0$:  ✓

$n \to n+1$:

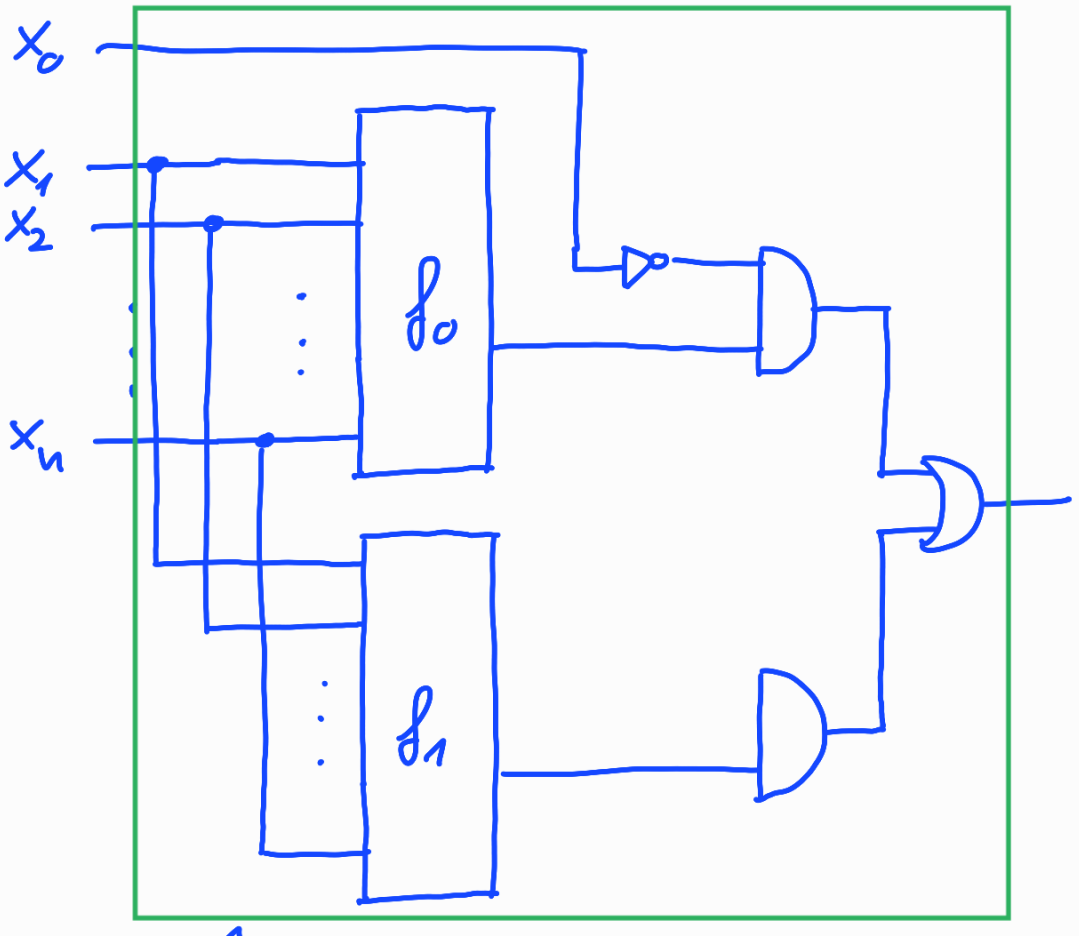
$$f : \mathbb{Z}_2^{n+1} \to \mathbb{Z}_2$$

$\to$ $f_0, f_1 : \mathbb{Z}_2^{n} \to \mathbb{Z}_2$ def. by

$$f_0(x_1, \dots, x_n) = f(0, x_1, \dots, x_n)$$

$$f_1(x_1, \dots, x_n) = f(1, x_1, \dots, x_n)$$

by inductive hypothesis f_0 and f_1 can be implemented with NAND and BRANCH; and so f:

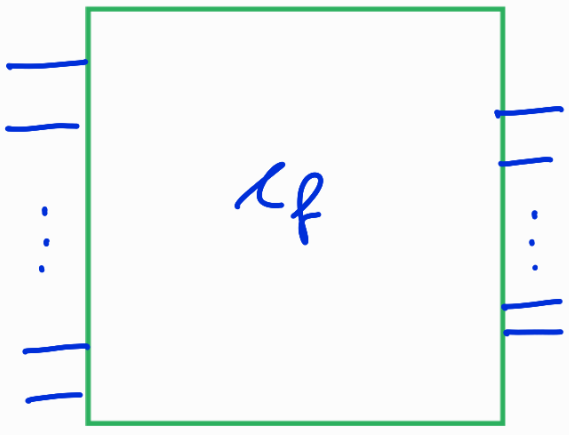


↑ circuit for f !

circuit model of computation:

computation $\longrightarrow$ circuit

$f: \mathbb{Z}_2^n \to \mathbb{Z}_2^m$



$\rightsquigarrow$ computational cost of comp. f:

$r(f)$ = number of NAND and BRANCH needed to build circuit c_f

- emulation of c_f on CPU-computer requires

$$r' = \text{poly}(r(f))$$

CPU-operations!

- CPU - computer $\underline{\text{key}}$ of $T'(f)$

  CPU - operations (to compute f) can be implemented in a circuit with

  $$T = \text{poly}(T'(f))$$

  NAND & BRANCH gates !

→ circuits and CPU - computers are $\underline{\text{polynomially equivalent}}$!

→

Reversible Computation

circuits made of irreversible gates (as AND, OR, XOR, NAND, ...) makes computation an irreversible process !

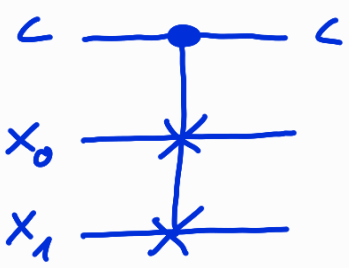
→ energy dissipation (1960!)

→ incompatible with unitary and thus reversible dynamics of a quantum system ("computer") (1990!)

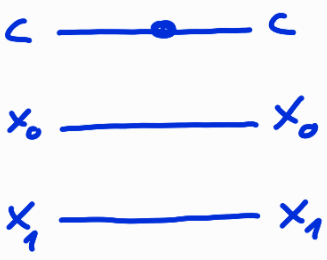
fortunately, there exist universal reversible gates → reversible computation!

1) CSWAP : controlled swap

(Fredkin-gate)

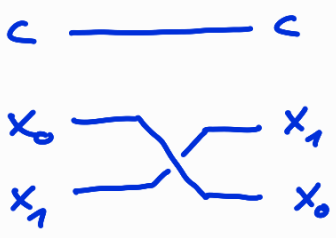


$c = 0$:



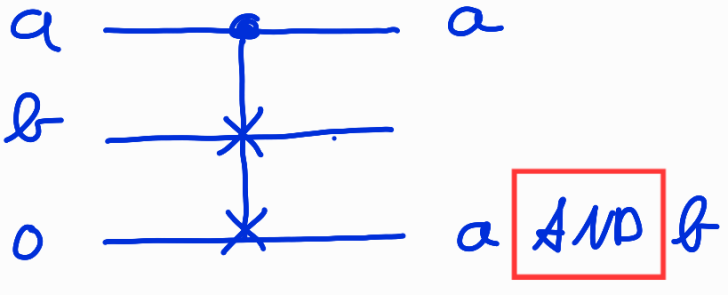
no swap !

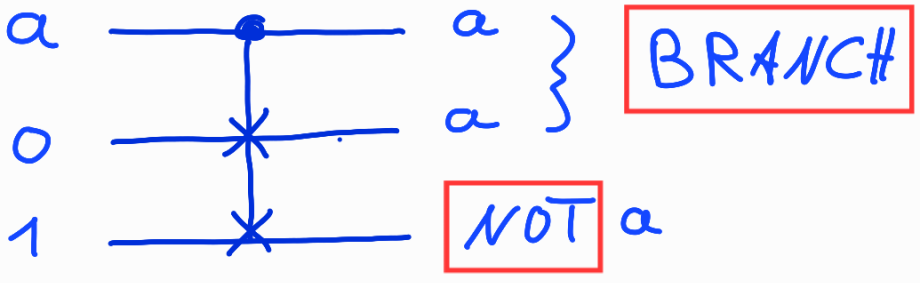
$c = 1$:



swap !

obviously reversible and also universal :



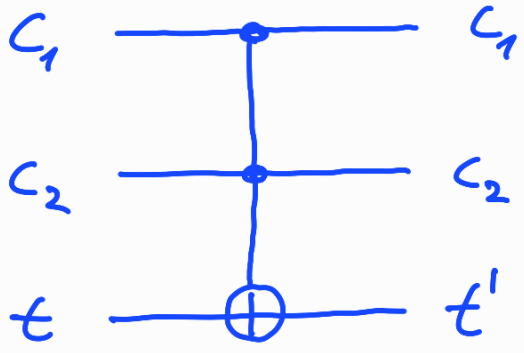


→ NAND , BRANCH : universality !

another universal + reversible gate:

2) CCNOT : doubly controlled NOT

(Toffoli-gate)



(universality: problem 6. on sheet 2)

Conclusion:

by use of reversible, universal gates any computation can be done reversibly, – without dissipating energy!

reversible computation of non-injective

(= irreversible) function $f: \mathbb{Z}_2^n \to \mathbb{Z}_2^m$?!

→ switch to injective (= reversible)

function $\tilde{f}: \mathbb{Z}_2^{n+m} \to \mathbb{Z}_2^{n+m}$

def. by

$$\tilde{f}(x,y) = (x, y \oplus f(x)) \quad !$$

- reversible:

$$\tilde{f} \circ \tilde{f}\,(x,y) = \tilde{f}(x, y \oplus f(x))$$

$$= (x, y \oplus \underbrace{f(x) \oplus f(x)}_{\| \; 0})$$

$$= (x,y)$$

$$\to \tilde{f} = \tilde{f}^{-1}$$

- on input $(x,0)$:

$$\tilde{f}(x,0) = (x, f(x))$$

✓