cours 2

pré-requis à l'étude des unités fonctionnelles d'une machine

plan

- codage
- numération
- algèbre de Boole
- circuits combinatoires
- circuits séquentiels

principe du codage

principe du codage

$I = \{i_1, \ldots, i_m\}$ ensemble d'informations

principe du codage

$I = \{i_1, \ldots, i_m\}$ ensemble d'informations

$A = \{a_1, \ldots, a_n\}$ alphabet

principe du codage

$I = \{i_1, \ldots, i_m\}$	ensemble d'informations
$A = \{a_1, \ldots, a_n\}$	alphabet
a_i	caractères de A

principe du codage

$I = \{i_1, \ldots, i_m\}$	ensemble d'informations
$A = \{a_1, \ldots, a_n\}$	alphabet
a_i	caractères de A
$a_1 a_3 a_4 a_8$	mot de A

principe du codage

$I = \{i_1, \ldots, i_m\}$	ensemble d'informations
$A = \{a_1, \ldots, a_n\}$	alphabet
a_i	caractères de A
$a_1 a_3 a_4 a_8$	mot de A
$\lvert A \rvert$	base du codage

principe du codage

$I = \{i_1, \ldots, i_m\}$	ensemble d'informations
$A = \{a_1, \ldots, a_n\}$	alphabet
a_i	caractères de A
$a_1 a_3 a_4 a_8$	mot de A
$\lvert A \rvert$	base du codage

coder I : associer à chaque $i \in I$ un mot de A

exemple

$I = \{\text{ain, ainse, allier, } \ldots\}$

$A = \{0, \ldots, 9\}$

$|A| = 10$

ain est associé à 1

loir-et-cher est associé à 41

. . .

quelques propriétés

codage *redondant*

quelques propriétés

codage *redondant* : numéro de téléphone

quelques propriétés

codage *redondant* : numéro de téléphone

codage non redondant

quelques propriétés

codage *redondant* : numéro de téléphone

codage non redondant : numéro de sécurité sociale

quelques propriétés

codage *redondant* : numéro de téléphone

codage non redondant : numéro de sécurité sociale

codage à longueur fixe

quelques propriétés

codage *redondant* : numéro de téléphone

codage non redondant : numéro de sécurité sociale

codage à longueur fixe : code postal

quelques propriétés

codage *redondant* : numéro de téléphone

codage non redondant : numéro de sécurité sociale

codage à longueur fixe : code postal

codage à longueur variable

quelques propriétés

codage *redondant* : numéro de téléphone

codage non redondant : numéro de sécurité sociale

codage à longueur fixe : code postal

codage à longueur variable : alphabet morse

quelques codes

quelques codes

– codage des nombres

quelques codes

– codage des nombres

– codage des données alphanumériques

quelques codes

– codage des nombres
– codage des données alphanumériques
– codage détecteur d'erreur

notation positionnelle

la représentation d'un nombre est un codage

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

dix,

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

dix, 10,

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

dix, 10, **********,

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

dix, 10, **********, 1010,

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

dix, 10, **********, 1010, A,

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

dix, 10, **********, 1010, A, X,

notation positionnelle

la représentation d'un nombre est un codage

ne pas confondre un nombre avec sa représentation !

la quantité dix peut être représentée par

dix, 10, **********, 1010, A, X, etc...

représentation d'un nombre dans une base b

$$a_n a_{n-1} \ldots a_1 a_{0_b} = \sum_{i=0}^{n} a_i \times b^i$$

a_n,a_{n-1}, . . . : poids forts

. . . ,a_1,a_0 : poids faibles

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

$$01101_2$$

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

$$01101_2 = 0 \times 2^4$$

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

$$01101_2 = 0 \times 2^4 + 1 \times 2^3$$

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

$$01101_2 = 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2$$

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

$$01101_2 = 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1$$

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

$$01101_2 = 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

représentation d'un nombre en base 2

alphabet de 2 caractères binaires (appelés *bit*)

$$\{0,1\}$$

$$\begin{aligned} 01101_2 &= 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 8_{10} + 4_{10} + 1_{10} \\ &= 13_{10} \end{aligned}$$

représentation d'un nombre en base 16

alphabet de 16 caractères *hexadécimaux*

$$\{0,\ldots,9,A,B,C,D,E,F\}$$

$$\begin{aligned} 01B_{16} &= 0 \times 16^2 + 1 \times 16^1 + B \times 16^0 \\ &= 16_{10} + 11_{10} \\ &= 27_{10} \end{aligned}$$

code binaire naturel/pur

basé sur la représentation des nombres en base 2

0	000
1	001
2	010
⋮	⋮
6	110
7	111

code Décimal Codé Binaire (BCD/DCB)

chaque chiffre est codé sur 4 bits

0	0000
1	0001
2	0011
⋮	⋮
10	0001 0000
11	0001 0001

code de Gray

distance de 1 entre deux mots de code consécutif

0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

détection d'erreurs

un code peut être utilisé pour *détecter* des erreurs

exemple : ajout d'un bit pour maintenir la parité

– 0110 devient 01100
– 1011 devient 10111

bit de parité

0110 à transmettre

bit de parité

0110 à transmettre

calcul du bit de parité : 0

bit de parité

0110 à transmettre

calcul du bit de parité : 0

01100 transmis

bit de parité

0110 à transmettre

calcul du bit de parité : 0

01100 transmis

erreur de transmission

bit de parité

0110 à transmettre

calcul du bit de parité : 0

01100 transmis

erreur de transmission

01110 arrivé

bit de parité

0110 à transmettre

calcul du bit de parité : 0

01100 transmis

erreur de transmission

01110 arrivé

recalcul du bit de parité : 1

bit de parité

0110 à transmettre

calcul du bit de parité : 0

01100 transmis

erreur de transmission

01110 arrivé

recalcul du bit de parité : 1

comparaison avec le bit de parité arrivé : erreur détectée

codage des données alphanumériques

code *ASCII* : 7 bits + 1 bit de parité (un octet)

bits 654 \ bits 3210	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
010	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

unicode : code sur 16 bits contenant pratiquement tous les alphabets existants

système de numération

système de numération décimal

système de numération

système de numération décimal

système de numération binaire

systèmes de numération

systèmes de numération

– entiers naturels

systèmes de numération

– entiers naturels

– entiers relatifs

systèmes de numération

– entiers naturels
– entiers relatifs
– réels

base b vers base 10

$a_n a_{n-1} \ldots a_1 a_0$ exprimé en base b (noté $a_n a_{n-1} \ldots a_1 a_{0_b}$)

vers une représentation en base 10 :

$$a_n \times b^n + a_{n-1} \times b^{n-1} + \ldots + a_1 \times b + a_0$$

partie fractionnaire :

$$a_1 \times b^{-1} + a_2 \times b^{-2} + \ldots + a_n \times b^{-n}$$

base 10 vers base b

pour A entier

$$A_{10} = a_n \times b^n + a_{n-1} \times b^{n-1} + \ldots + a_1 \times b + a_0$$

$$= ((\ldots(a_n \times b + a_{n-1}) \times b + \ldots) \times b + a_1) \times b + a_0$$

a_0 est le reste de la division entière de A par la base b

base 10 vers base b

pour la partie fractionnaire

$$A_{10} = a_1 \times b^{-1} + a_2 \times b^{-2} + \ldots + a_n \times b^{-n}$$

$$(a_1 + (a_2 + (\ldots + (a_{n-1} + a_n \times b - 1)b^{-1} \ldots)b^{-1})b^{-1})b^{-1}$$

a_1 est la partie entière de la multiplication de A par b

principe de conversion

partie entière :

principe de conversion

partie entière :

– divisions entières successives par la base

principe de conversion

partie entière :

– divisions entières successives par la base

– lecture du reste

principe de conversion

partie entière :

– divisions entières successives par la base

– lecture du reste

partie fractionnaire :

principe de conversion

partie entière :

– divisions entières successives par la base

– lecture du reste

partie fractionnaire :

– multiplications successives par la base

principe de conversion

partie entière :

– divisions entières successives par la base
– lecture du reste

partie fractionnaire :

– multiplications successives par la base
– lecture de la partie entière

exemple

25_{10}

exemple

$25_{10}\ /2 = 12_{10}$ reste 1

exemple

$25_{10}\ /2 = 12_{10}$ reste 1

$12_{10}/2 = 6_{10}$ reste 0

exemple

$25_{10}\ /2 = 12_{10}$ reste 1

$12_{10}/2 = 6_{10}$ reste 0

$6_{10}/2 = 3_{10}$ reste 0

exemple

25_{10} $/2 = 12_{10}$ reste 1

$12_{10}/2 = 6_{10}$ reste 0

$6_{10}/2 = 3_{10}$ reste 0

$3_{10}/2 = 1_{10}$ reste 1

exemple

25_{10} $/2 = 12_{10}$ reste 1

$12_{10}/2 = 6_{10}$ reste 0

$6_{10}/2 = 3_{10}$ reste 0

$3_{10}/2 = 1_{10}$ reste 1

$1_{10}/2 = 0_{10}$ reste 1

exemple

$25_{10}\ /2 = 12_{10}$ reste 1

$12_{10}/2 = 6_{10}$ reste 0

$6_{10}/2 = 3_{10}$ reste 0

$3_{10}/2 = 1_{10}$ reste 1

$1_{10}/2 = 0_{10}$ reste 1

donc

$25_{10} = 11001_2$

exemple

$0{,}78125_{10}$

exemple

$0{,}78125_{10} \times 2 = 1{,}5625_{10}$ partie entière 1

exemple

$0{,}78125_{10} \times 2 = 1{,}5625_{10}$ partie entière 1

$0{,}5625_{10} \times 2 = 1{,}125_{10}$ partie entière 1

exemple

$0{,}78125_{10} \times 2 = 1{,}5625_{10}$ partie entière 1

$0{,}5625_{10} \times 2 = 1{,}125_{10}$ partie entière 1

$0{,}125_{10} \times 2 = 0{,}25_{10}$ partie entière 0

exemple

$0{,}78125_{10} \times 2 = 1{,}5625_{10}$ partie entière 1

$0{,}5625_{10} \times 2 = 1{,}125_{10}$ partie entière 1

$0{,}125_{10} \times 2 = 0{,}25_{10}$ partie entière 0

$0{,}25_{10} \times 2 = 0{,}5_{10}$ partie entière 0

exemple

$0{,}78125_{10} \times 2 = 1{,}5625_{10}$ partie entière 1

$0{,}5625_{10} \times 2 = 1{,}125_{10}$ partie entière 1

$0{,}125_{10} \times 2 = 0{,}25_{10}$ partie entière 0

$0{,}25_{10} \times 2 = 0{,}5_{10}$ partie entière 0

$0{,}5_{10} \times 2 = 1_{10}$ partie entière 1

exemple

$0{,}78125_{10} \times 2 = 1{,}5625_{10}$ partie entière 1

$0{,}5625_{10} \times 2 = 1{,}125_{10}$ partie entière 1

$0{,}125_{10} \times 2 = 0{,}25_{10}$ partie entière 0

$0{,}25_{10} \times 2 = 0{,}5_{10}$ partie entière 0

$0{,}5_{10} \times 2 = 1_{10}$ partie entière 1

donc

$0{,}78125_{10} = 0{,}11001_{2}$

représentations des nombres négatifs

– signe et valeur absolue

représentations des nombres négatifs

– signe et valeur absolue

– notations complémentées

représentations des nombres négatifs

– signe et valeur absolue
– notations complémentées
– notations excédentaires

signe et valeur absolue

Sur n bits

signe : bit de poids fort (0 : positif, 1 : négatif)

valeur absolue : $n - 1$ bits

intervalle de valeurs représentées : $[-2^{n-1} + 1, 2^{n-1} - 1]$

exemple sur 3 bits : $[-3, +3]$

000 0

exemple sur 3 bits : $[-3, +3]$

000	0
001	1

exemple sur 3 bits : $[-3, +3]$

000	0
001	1
010	2

exemple sur 3 bits : $[-3, +3]$

000	0
001	1
010	2
011	3

exemple sur 3 bits : $[-3, +3]$

000	0
001	1
010	2
011	3
100	-0

exemple sur 3 bits : $[-3, +3]$

000	0
001	1
010	2
011	3
100	-0
101	-1

exemple sur 3 bits : $[-3, +3]$

000	0
001	1
010	2
011	3
100	-0
101	-1
110	-2

exemple sur 3 bits : $[-3, +3]$

000	0
001	1
010	2
011	3
100	-0
101	-1
110	-2
111	-3

complément à 1

sur n bits

inversion de chaque bit de la valeur absolue

$|x| + (-|x|) = 2^n - 1$

intervalle de valeurs représentées : $[-2^{n-1} + 1, 2^{n-1} - 1]$

exemple sur 3 bits : $[-3,3]$

000 0

exemple sur 3 bits : $[-3,3]$

000	0
001	1

exemple sur 3 bits : $[-3,3]$

000	0
001	1
010	2

exemple sur 3 bits : $[-3,3]$

000	0
001	1
010	2
011	3

exemple sur 3 bits : $[-3,3]$

000	0
001	1
010	2
011	3
100	-3

exemple sur 3 bits : $[-3,3]$

000	0
001	1
010	2
011	3
100	-3
101	-2

exemple sur 3 bits : $[-3,3]$

000	0
001	1
010	2
011	3
100	-3
101	-2
110	-1

exemple sur 3 bits : $[-3,3]$

000	0
001	1
010	2
011	3
100	-3
101	-2
110	-1
111	-0

complément à 2

sur n bits

complément à $1 + 1$

$|x| - |x| = 2^n$

intervalle de valeurs représentées : $[-2^{n-1}, 2^{n-1} - 1]$

exemple sur 3 bits : $[-4,3]$

000 0

exemple sur 3 bits : $[-4,3]$

000	0
001	1

exemple sur 3 bits : $[-4,3]$

000	0
001	1
010	2

exemple sur 3 bits : $[-4,3]$

000	0
001	1
010	2
011	3

exemple sur 3 bits : $[-4,3]$

000	0
001	1
010	2
011	3
100	-4

exemple sur 3 bits : $[-4,3]$

000	0
001	1
010	2
011	3
100	-4
101	-3

exemple sur 3 bits : $[-4,3]$

000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2

exemple sur 3 bits : $[-4,3]$

000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

notation excédentaire

sur n bits

ajout de la valeur d'un excès

souvent translation de 2^{n-1}

avec un excès de 2^{n-1}

$-|x| = 2^{n-1} - |x|$

intervalle de valeurs représentées : $[-2^{n-1}, 2^{n-1} - 1]$

$x < 0$ représentatble si $x \geq$ - excès

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000 -4

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000	-4
001	-3

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000	-4
001	-3
010	-2

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000	-4
001	-3
010	-2
011	-1

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000	-4
001	-3
010	-2
011	-1
100	0

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000	-4
001	-3
010	-2
011	-1
100	0
101	1

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000	-4
001	-3
010	-2
011	-1
100	0
101	1
110	2

exemple sur 3 bits, excédent $2^2 = 4$: $[-4,3]$

000	-4
001	-3
010	-2
011	-1
100	0
101	1
110	2
111	3

représentations des réels

– virgule fixe

représentations des réels

– virgule fixe
– couple (numérateur, dénominateur)

représentations des réels

– virgule fixe
– couple (numérateur, dénominateur)
– virgule flottante

virgule flottante

un réel $\pm m \times b^e$ est représenté par

virgule flottante

un réel $\pm m \times b^e$ est représenté par

– un signe $\pm$

virgule flottante

un réel $\pm m \times b^e$ est représenté par

– un signe $\pm$

– une mantisse m

virgule flottante

un réel $\pm m \times b^e$ est représenté par

– un signe $\pm$

– une mantisse m

– un exposant e

virgule flottante

un réel $\pm m \times b^e$ est représenté par

– un signe $\pm$

– une mantisse m

– un exposant e

– une base b

virgule flottante

un réel $\pm m \times b^e$ est représenté par

– un signe $\pm$

– une mantisse m

– un exposant e

– une base b

représentation la plus utilisée

virgule flottante

un réel $\pm m \times b^e$ est représenté par

– un signe $\pm$

– une mantisse m

– un exposant e

– une base b

représentation la plus utilisée

infinité de représentations possibles d'un même nombre

représentation normalisée

pour une base b

la mantisse $m \in [1,b[$

représentation particulière pour zéro

précision et intervalle de valeurs représentées dépendent

– du nombre de bits de la mantisse
– du nombre de bits de l'exposant

norme IEEE754

– la mantisse $m \in [1.0_2, 10.0_2[$

norme IEEE754

– la mantisse $m \in [1.0_2, 10.0_2[$

– le 1 à gauche de la virgule n'est pas représenté

norme IEEE754

- la mantisse $m \in [1.0_2, 10.0_2[$
- le 1 à gauche de la virgule n'est pas représenté
- nombres codés sur
  - 32 bits (simple précision)

norme IEEE754

- la mantisse $m \in [1.0_2, 10.0_2[$
- le 1 à gauche de la virgule n'est pas représenté
- nombres codés sur
  - 32 bits (simple précision)
    exposant codé avec un excès de 127

norme IEEE754

- la mantisse $m \in [1.0_2, 10.0_2[$
- le 1 à gauche de la virgule n'est pas représenté
- nombres codés sur
  - 32 bits (simple précision)
    exposant codé avec un excès de 127
  - 64 bits (double précision)

norme IEEE754

- la mantisse $m \in [1.0_2, 10.0_2[$
- le 1 à gauche de la virgule n'est pas représenté
- nombres codés sur
  - 32 bits (simple précision)
    exposant codé avec un excès de 127
  - 64 bits (double précision)
    exposant est codé avec un excès de 1023

norme IEEE754

- la mantisse $m \in [1.0_2, 10.0_2[$
- le 1 à gauche de la virgule n'est pas représenté
- nombres codés sur
  - 32 bits (simple précision)
    exposant codé avec un excès de 127
  - 64 bits (double précision)
    exposant est codé avec un excès de 1023

c'est la norme la plus utilisée pour représenter les réels

simple précision

$$(-1)^S \times 1,M \times 2^{E-127}$$

S 1 bits	exposant E 8 bits	mantisse M 23 bits

exemple

-5_{10} est codé par

1 100 0000 1 010 0000 0000 0000 0000 0000

c'est à dire $-1 \times 1{,}25 \times 2^{129-127}$

précision

la représentation machine des réels sur une machine se base sur un nombre fini de valeurs

précision

la représentation machine des réels sur une machine se base sur un nombre fini de valeurs

c'est donc une représentation approchée !

précision

la représentation machine des réels sur une machine se base sur un nombre fini de valeurs

c'est donc une représentation approchée !

précision de la représentation = différence entre les mantisses de deux nombres réels consécutifs

IEEE 754, simple précision : la précision est de 2^{-23}

valeurs particulières avec IEEE 754

E	M	valeur représentée
max	0	$\pm\infty$
max	$\neq 0$	NaN
0	0	0
0	$\neq 0$	nombres dénormalisés $(-1)^S \times 0,M \times 2^{-126}$

algèbre de Boole

algèbre de Boole

due à Georges Boole (1815-1864)

algèbre de Boole

due à Georges Boole (1815-1864)

une algèbre de Boole est la donnée de

algèbre de Boole

due à Georges Boole (1815-1864)

une algèbre de Boole est la donnée de

– un ensemble E

algèbre de Boole

due à Georges Boole (1815-1864)

une algèbre de Boole est la donnée de

– un ensemble E
– deux éléments particuliers de E : 0 et 1

algèbre de Boole

due à Georges Boole (1815-1864)

une algèbre de Boole est la donnée de

– un ensemble E
– deux éléments particuliers de E : 0 et 1
– deux opérations binaires sur E : $+$ et .

algèbre de Boole

due à Georges Boole (1815-1864)

une algèbre de Boole est la donnée de

– un ensemble E
– deux éléments particuliers de E : 0 et 1
– deux opérations binaires sur E : $+$ et .
– une opération unaire sur E : -

axiomes de l'algèbre de Boole

soient a,$b \in E$

axiomes de l'algèbre de Boole

soient a,$b \in E$

commutativité a+b=b+a ab=ba

axiomes de l'algèbre de Boole

soient a,$b \in E$

commutativité	a+b=b+a	ab=ba
associativité	(a+b)+c = a+(b+c)	(ab)c=a(bc)

axiomes de l'algèbre de Boole

soient a,$b \in E$

commutativité	a+b=b+a	ab=ba
associativité	(a+b)+c = a+(b+c)	(ab)c=a(bc)
distributivité	a(b+c)=ab+ac	a+(bc)=(a+b)(a+c)

axiomes de l'algèbre de Boole

soient a,$b \in E$

commutativité	a+b=b+a	ab=ba
associativité	(a+b)+c = a+(b+c)	(ab)c=a(bc)
distributivité	a(b+c)=ab+ac	a+(bc)=(a+b)(a+c)
éléments neutres	a+0=a	a1=a

axiomes de l'algèbre de Boole

soient a,$b \in E$

commutativité	a+b=b+a	ab=ba
associativité	(a+b)+c = a+(b+c)	(ab)c=a(bc)
distributivité	a(b+c)=ab+ac	a+(bc)=(a+b)(a+c)
éléments neutres	a+0=a	a1=a
complémentation	a+$\overline{a}$=1	a$\overline{a}$=0

théorèmes

idempotence $a+a=a$ $aa=a$

théorèmes

idempotence	a+a=a	aa=a
absorption	a+ab=a	a(a+b)=a

théorèmes

idempotence	a+a=a	aa=a
absorption	a+ab=a	a(a+b)=a
De Morgan (dualité)	$\overline{a+b}=\overline{a}.\overline{b}$	$\overline{ab}=\overline{a}+\overline{b}$

théorèmes

idempotence	a+a=a	aa=a
absorption	a+ab=a	a(a+b)=a
De Morgan (dualité)	$\overline{a+b}=\overline{a}.\overline{b}$	$\overline{ab}=\overline{a}+\overline{b}$
éléments absorbants	a+1=1	a0=0

démonstration de l'idempotence

$$aa$$

démonstration de l'idempotence

$$aa = aa + 0$$

démonstration de l'idempotence

$$\begin{aligned} aa &= aa + 0 \\ &= aa + a\overline{a} \end{aligned}$$

démonstration de l'idempotence

$$\begin{aligned} aa &= aa + 0 \\ &= aa + a\overline{a} \\ &= a(a + \overline{a}) \end{aligned}$$

démonstration de l'idempotence

$$\begin{aligned} aa &= aa + 0 \\ &= aa + a\overline{a} \\ &= a(a + \overline{a}) \\ &= a1 \end{aligned}$$

démonstration de l'idempotence

$$
\begin{aligned}
aa &= aa + 0 \\
&= aa + a\overline{a} \\
&= a(a + \overline{a}) \\
&= a1 \\
&= a
\end{aligned}
$$

l'algèbre de Boole minimale

$E = \{0,1\}$ pouvant être interprétée

l'algèbre de Boole minimale

$E = \{0,1\}$ pouvant être interprétée

– 1 est interprété "vrai"

l'algèbre de Boole minimale

$E = \{0,1\}$ pouvant être interprétée

– 1 est interprété "vrai"

– 0 est interprété "faux"

l'algèbre de Boole minimale

$E = \{0,1\}$ pouvant être interprétée

– 1 est interprété "vrai"

– 0 est interprété "faux"

– $+$ est interprété "ou" (disjonction logique)

l'algèbre de Boole minimale

$E = \{0,1\}$ pouvant être interprétée

– 1 est interprété "vrai"

– 0 est interprété "faux"

– $+$ est interprété "ou" (disjonction logique)

– . est interprété "et" (conjonction logique)

l'algèbre de Boole minimale

$E = \{0,1\}$ pouvant être interprétée

– 1 est interprété "vrai"
– 0 est interprété "faux"
– $+$ est interprété "ou" (disjonction logique)
– . est interprété "et" (conjonction logique)
– $-$ est interprété "non" (négation logique)

autre interprétation

– 1 est interprété “le courant passe”

autre interprétation

– 1 est interprété “le courant passe”

– 0 est interprété “le courant ne passe pas”

autre interprétation

- 1 est interprété “le courant passe”
- 0 est interprété “le courant ne passe pas”
- $+$ est interprété
  “le courant passe si au moins une des entrées est à 1”

autre interprétation

- 1 est interprété “le courant passe”
- 0 est interprété “le courant ne passe pas”
- $+$ est interprété
  “le courant passe si au moins une des entrées est à 1”
- . est interprété
  “le courant passe si les 2 entrées sont à 1”

autre interprétation

- 1 est interprété “le courant passe”
- 0 est interprété “le courant ne passe pas”
- $+$ est interprété
  “le courant passe si au moins une des entrées est à 1”
- . est interprété
  “le courant passe si les 2 entrées sont à 1”
- $-$ est interprété “inverse”

tables de vérité

description d'une opération logique

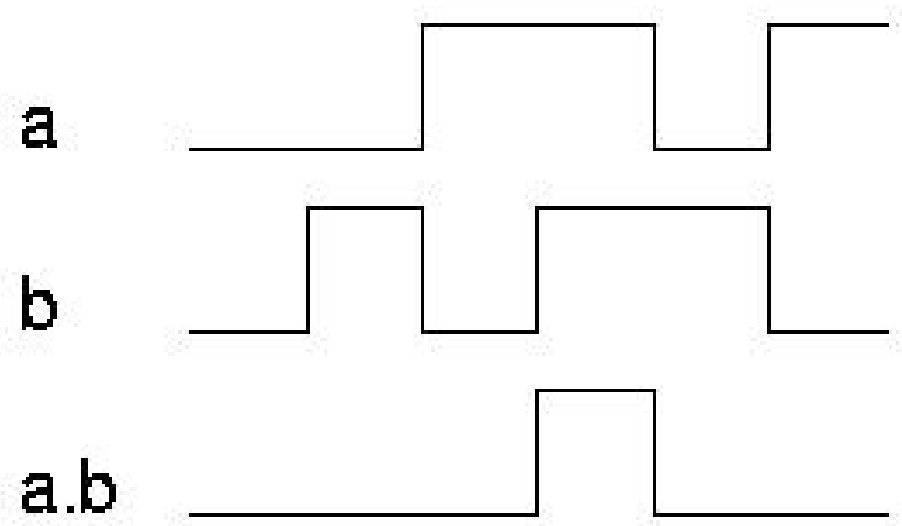
a	$\overline{a}$
0	1
1	0

a	b	$a+b$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	ab
0	0	0
0	1	0
1	0	0
1	1	1

opération et

chronogramme



ou exclusif

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

ou exclusif

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

$a \oplus b$ vaut 1 quand

ou exclusif

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

$a \oplus b$ vaut 1 quand $a = 0$ et $b = 1$

ou exclusif

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

$a \oplus b$ vaut 1 quand $a = 0$ et $b = 1$ ou $a = 1$ et $b = 0$

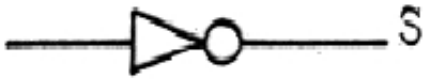
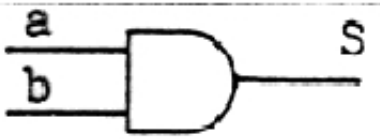
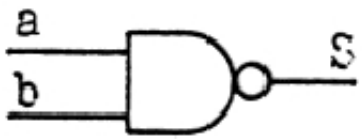
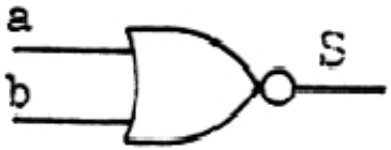
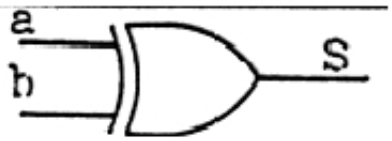
ou exclusif

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

$a \oplus b$ vaut 1 quand $a = 0$ et $b = 1$ ou $a = 1$ et $b = 0$

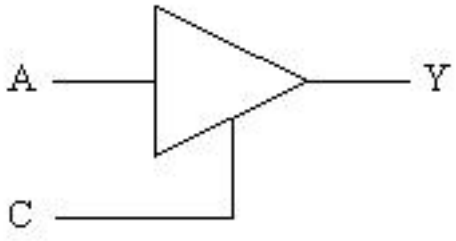
$a \oplus b = \overline{a}b + a\overline{b}$

portes logiques

	Ampli inverseur
	Opérateur ET (AND) $S = a.b$
	Opérateur OU (OR) $S = a + b$
	Opérateur NON ET (NAND) $S = \overline{a.b} = \overline{a} + \overline{b}$
	Opérateur NON OU (NOR) $S = \overline{a + b} = \overline{a}.\overline{b}$
	Opérateur OU Exclusif (XOR) $S = a\overline{b} + \overline{a}b$

porte 3 états

C	A	Y
1	0	0
1	1	1
0	X	déconnectée



principalement utilisée pour connecter une sortie sur une ligne commune à plusieurs circuits (bus)

portes logiques

mises en œuvres des opérations logiques de base

portes logiques

mises en œuvres des opérations logiques de base

les portes NAND et NOR sont

portes logiques

mises en œuvres des opérations logiques de base

les portes NAND et NOR sont

– complètes :
  on peut réaliser n'importe quelle fonction boléenne
  avec l'une ou l'autre

portes logiques

mises en œuvres des opérations logiques de base

les portes NAND et NOR sont

– complètes :
  on peut réaliser n'importe quelle fonction boléenne
  avec l'une ou l'autre
– faciles à fabriquer

portes logiques

mises en œuvres des opérations logiques de base

les portes NAND et NOR sont

- complètes :
  on peut réaliser n'importe quelle fonction boléenne avec l'une ou l'autre
- faciles à fabriquer
- à la base de la plupart des circuits intégrés des ordinateurs actuels

expression booléenne

terme construit à partir de

expression booléenne

terme construit à partir de

– variables booléennes

expression booléenne

terme construit à partir de

– variables booléennes
– d'opérateur booléens

expression booléenne

terme construit à partir de

– variables booléennes
– d'opérateur booléens

exemple

expression booléenne

terme construit à partir de

– variables booléennes
– d'opérateur booléens

exemple

a

expression booléenne

terme construit à partir de

– variables booléennes
– d'opérateur booléens

exemple

a

$\overline{c}$

expression booléenne

terme construit à partir de

– variables booléennes

– d'opérateur booléens

exemple

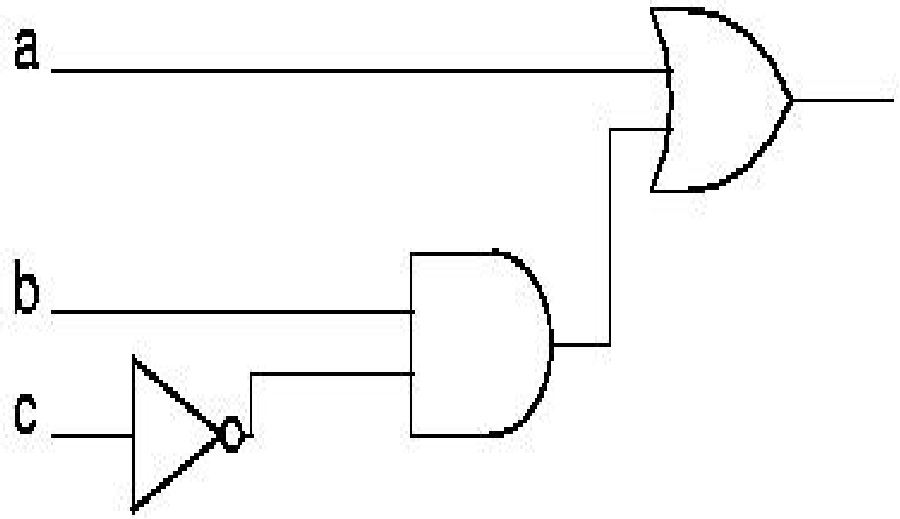
a

$\overline{c}$

$a + b\overline{c}$

expression booléenne

peut être exprimée (réalisée) par des portes logiques



fonction booléenne

fonction binaire à variables binaires

de $\{0,1\}^n$ dans $\{0,1\}$

peut être décrite par

fonction booléenne

fonction binaire à variables binaires

de $\{0,1\}^n$ dans $\{0,1\}$

peut être décrite par

– sa table de vérité

fonction booléenne

fonction binaire à variables binaires

de $\{0,1\}^n$ dans $\{0,1\}$

peut être décrite par

– sa table de vérité

– une expression booléenne

exemple : fonction booléenne majorité

a	b	c	majorité
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

majorité(a,b,c)

exemple : fonction booléenne majorité

a	b	c	majorité
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\text{majorité}(a,b,c) \quad = \quad \overline{a}bc$$

exemple : fonction booléenne majorité

a	b	c	majorité
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\begin{aligned} \text{majorité}(a,b,c) &= \overline{a}bc \\ &+ a\overline{b}c \end{aligned}$$

exemple : fonction booléenne majorité

a	b	c	majorité
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\begin{aligned} \text{majorité}(a,b,c) &= \overline{a}bc \\ &+ a\overline{b}c \\ &+ ab\overline{c} \end{aligned}$$

exemple : fonction booléenne majorité

a	b	c	majorité
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\begin{aligned} \text{majorité}(a,b,c) &= \overline{a}bc \\ &+ a\overline{b}c \\ &+ ab\overline{c} \\ &+ abc \end{aligned}$$

simplificaton d'expressions booléennes

f_1 et f_2 fonctions booléennes équivalentes ssi

$$\forall x_1,\ldots,x_n \in \{0,1\}, f_1(x_1,\ldots,x_n) = f_2(x_1,\ldots,x_n)$$

simplifier l'expression booléenne = minimiser le coût de réalisation

exemple : majorité$(a,b,c) = bc + ac + ab$

circuits logiques combinatoires

circuits dont la fonction de sortie s'exprime par une expression logique des variables d'entrées

circuits logiques combinatoires

circuits dont la fonction de sortie s'exprime par une expression logique des variables d'entrées

quelques circuits combinatoires intervenant dans la réalisation des composants d'un ordinateur

circuits logiques combinatoires

circuits dont la fonction de sortie s'exprime par une expression logique des variables d'entrées

quelques circuits combinatoires intervenant dans la réalisation des composants d'un ordinateur

– décodeur

circuits logiques combinatoires

circuits dont la fonction de sortie s'exprime par une expression logique des variables d'entrées

quelques circuits combinatoires intervenant dans la réalisation des composants d'un ordinateur

– décodeur
– multiplexeur

décodeur

permet d'envoyer un signal à une sortie choisie

- n lignes d'entrées
- 2^n lignes de sortie

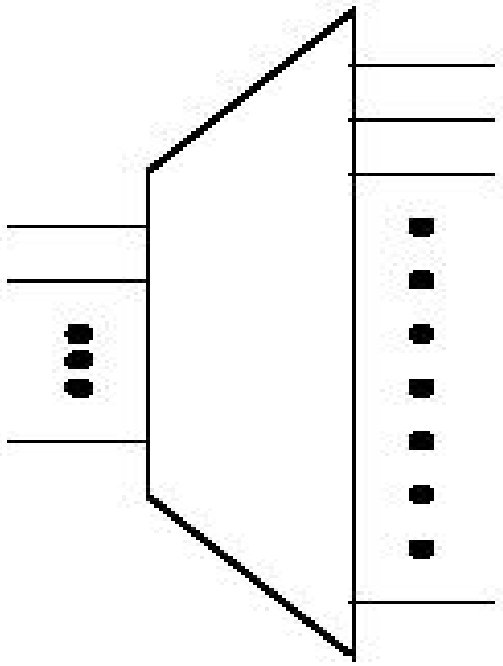
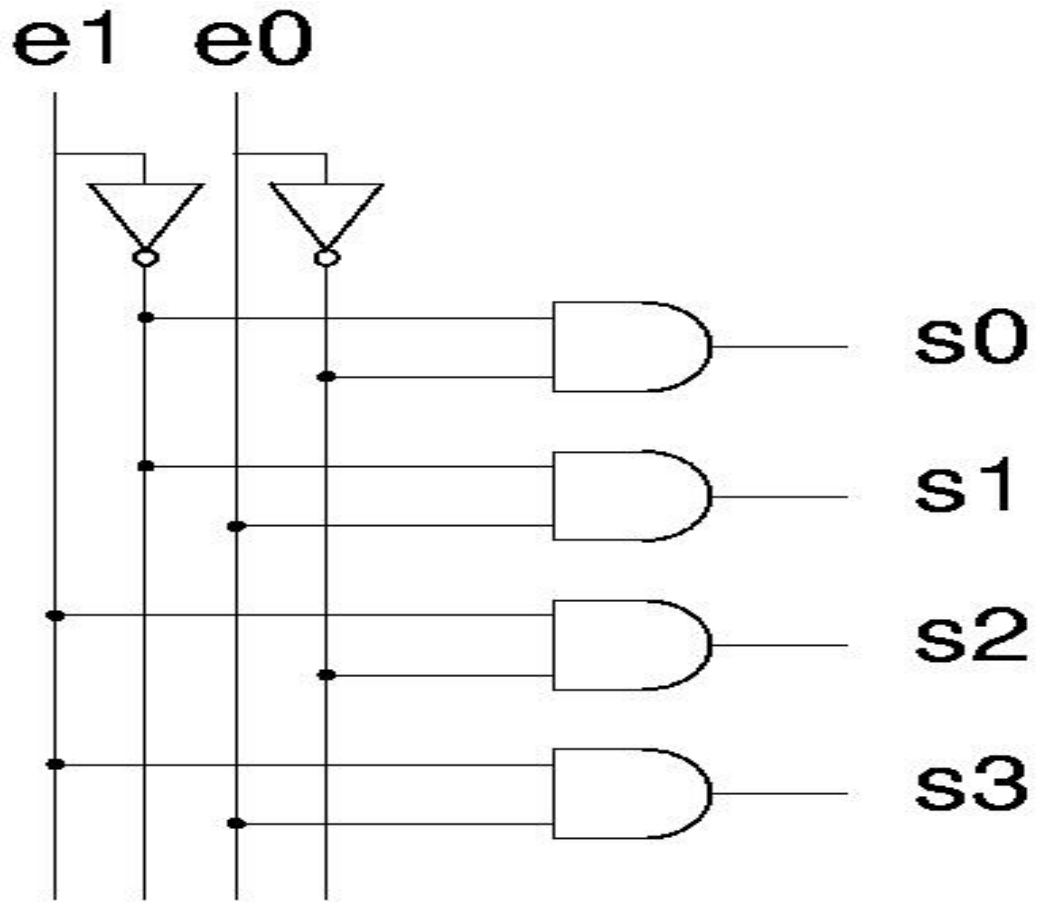


table de vérité

décodeur "2 vers 4" $(n = 2)$

e_1	e_0	s_0	s_1	s_2	s_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

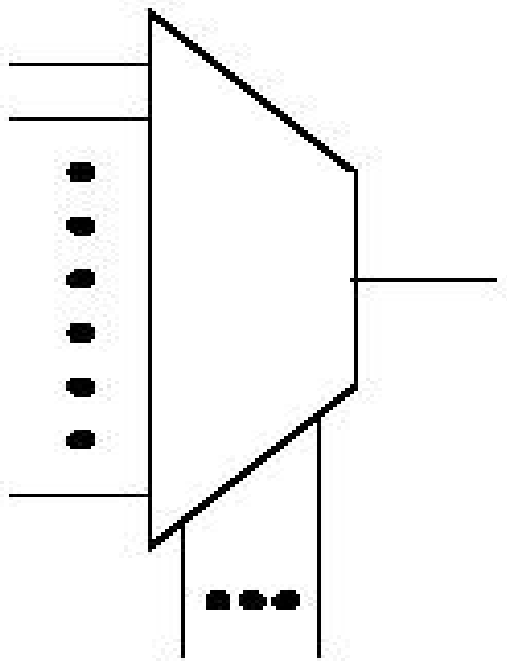
réalisation d'un décodeur 2 vers 4



multiplexeur

permet de sélectionner une entrée parmi plusieurs

- 2^n entrées
- 1 sortie
- n lignes de sélection



multiplexeur “4 voies” ($n = 2$)

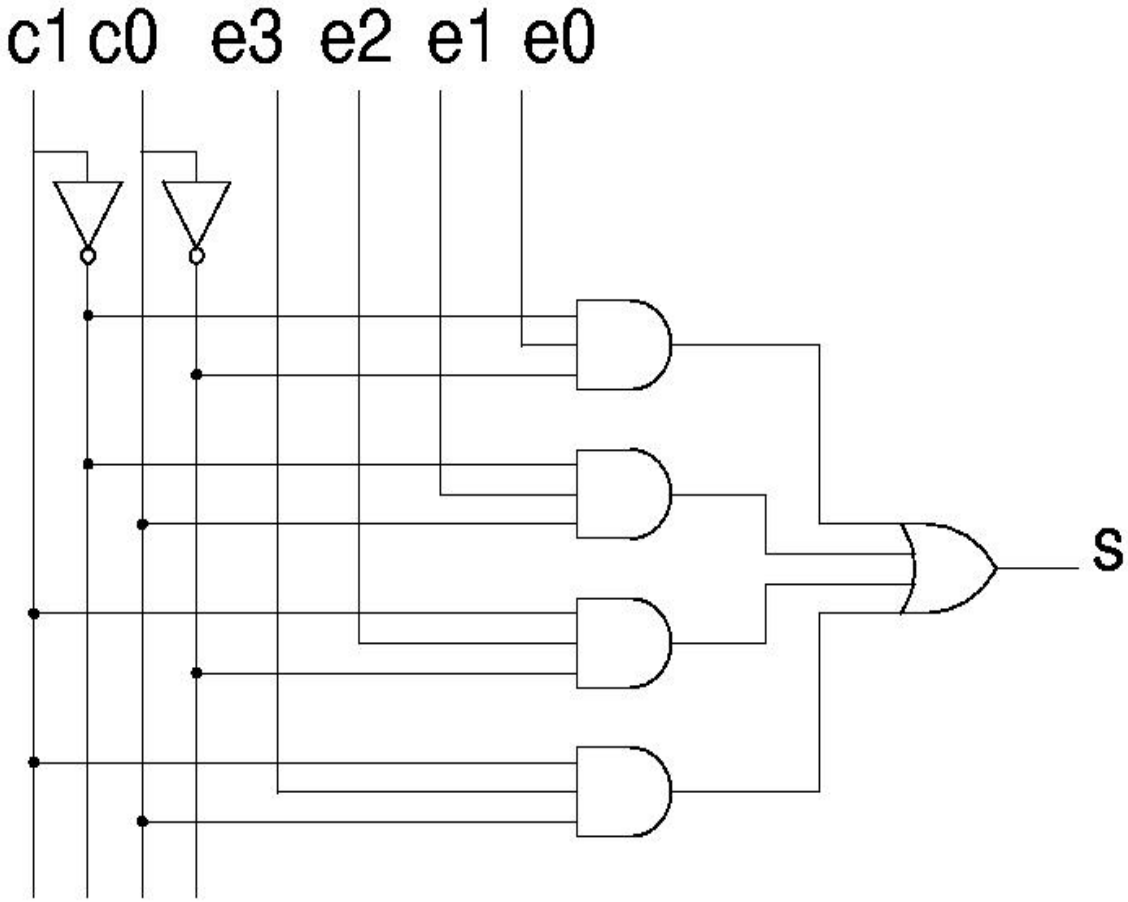


table de vérité

multiplexeur 4 voies

e_3	e_2	e_1	e_0	c_1	c_0	s
–	–	–	x	0	0	x
–	–	x	–	0	1	x
–	x	–	–	1	0	x
x	–	–	–	1	1	x

circuits séquentiels

circuits séquentiels

circuits de mémorisation

circuits séquentiels

circuits de mémorisation

leurs sorties dépendent

circuits séquentiels

circuits de mémorisation

leurs sorties dépendent

– de l'état des variables d'entrée

circuits séquentiels

circuits de mémorisation

leurs sorties dépendent

– de l'état des variables d'entrée
– de l'état antérieur de certaines variables de sortie

circuits séquentiels

un circuit séquentiel possède

circuits séquentiels

un circuit séquentiel possède

– des entrées E

circuits séquentiels

un circuit séquentiel possède

– des entrées E

– des sorties S

circuits séquentiels

un circuit séquentiel possède

– des entrées E
– des sorties S
– un état interne Q

circuits séquentiels

un circuit séquentiel possède

– des entrées E
– des sorties S
– un état interne Q

il est défini par deux fonctions

circuits séquentiels

un circuit séquentiel possède

– des entrées E
– des sorties S
– un état interne Q

il est défini par deux fonctions

– $S = f(E,Q)$ indique la nouvelle sortie

circuits séquentiels

un circuit séquentiel possède

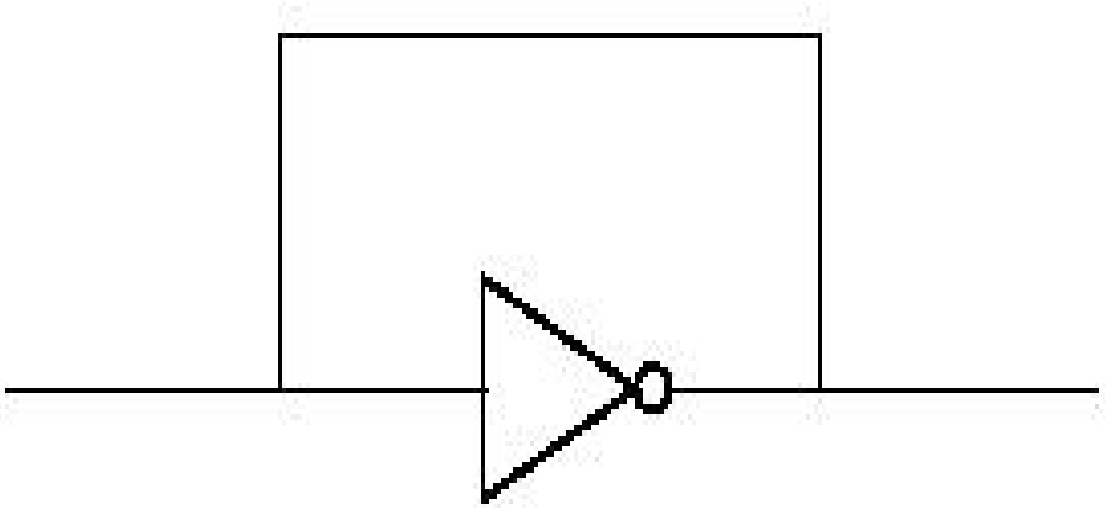
– des entrées E
– des sorties S
– un état interne Q

il est défini par deux fonctions

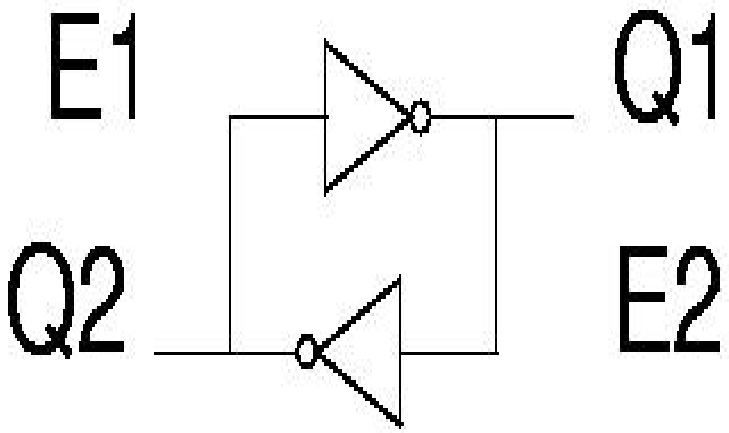
– $S = f(E,Q)$ indique la nouvelle sortie
– $Q' = g(E,Q)$ indique le nouvel état

notion d'état stable

circuit astable oscillant constament entre 0 et 1



bistable

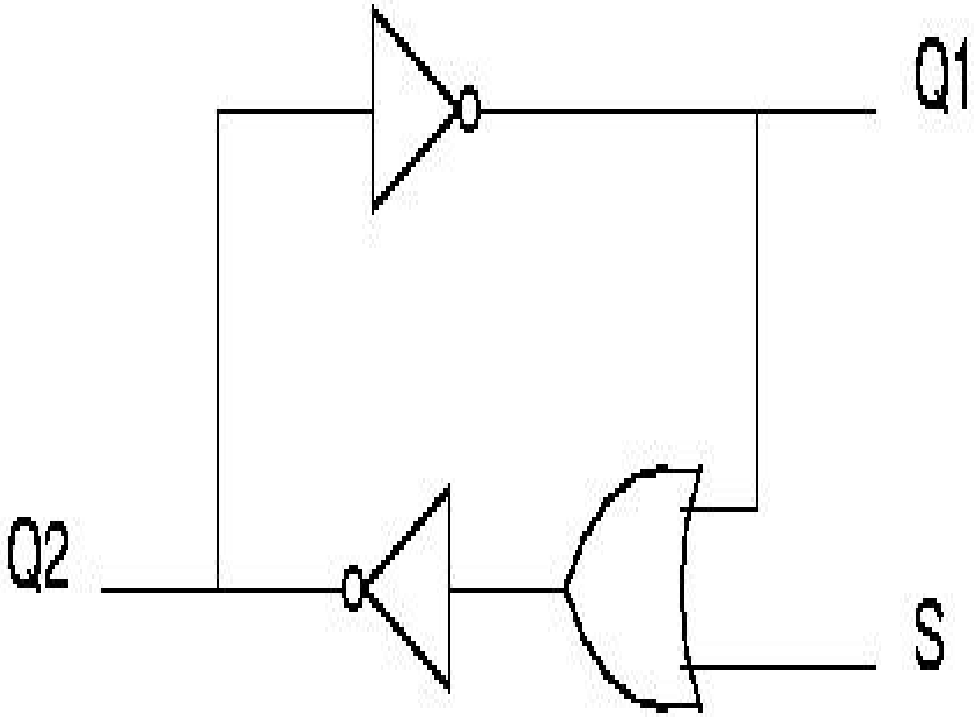


2 états *stables* différents

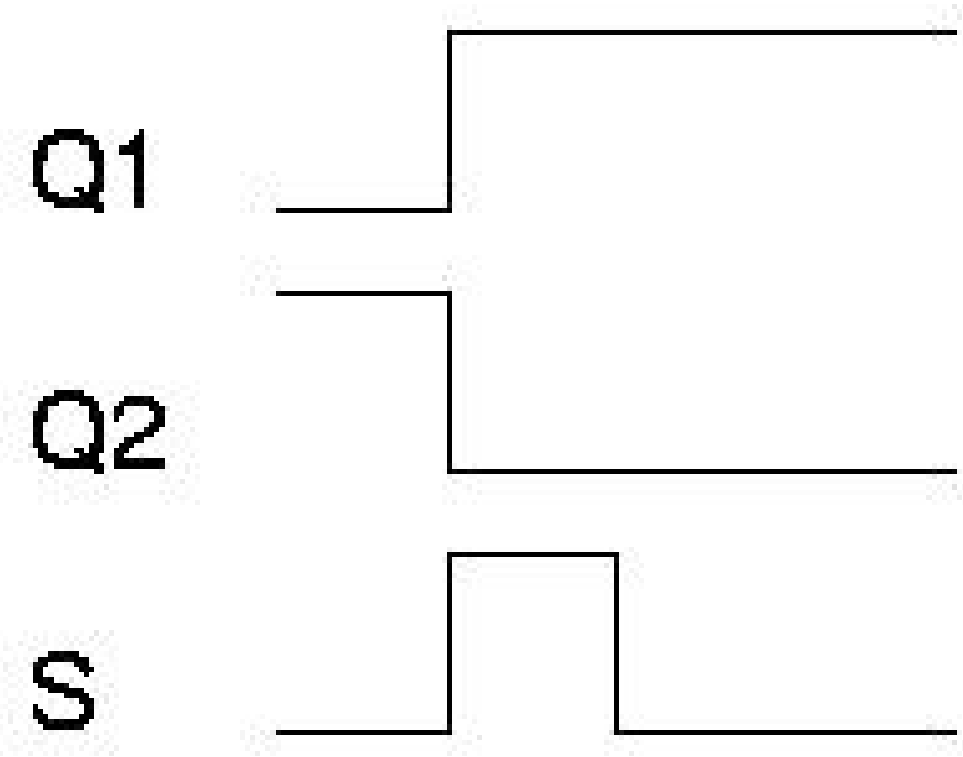
– $Q_1 = 0$ et $Q_2 = 1$
– $Q_1 = 1$ et $Q_2 = 0$

bistable avec commande Set

S commande la mise à 1



chronogramme



bascule RS

ajout de commandes reset (R) et set (S) au bistable

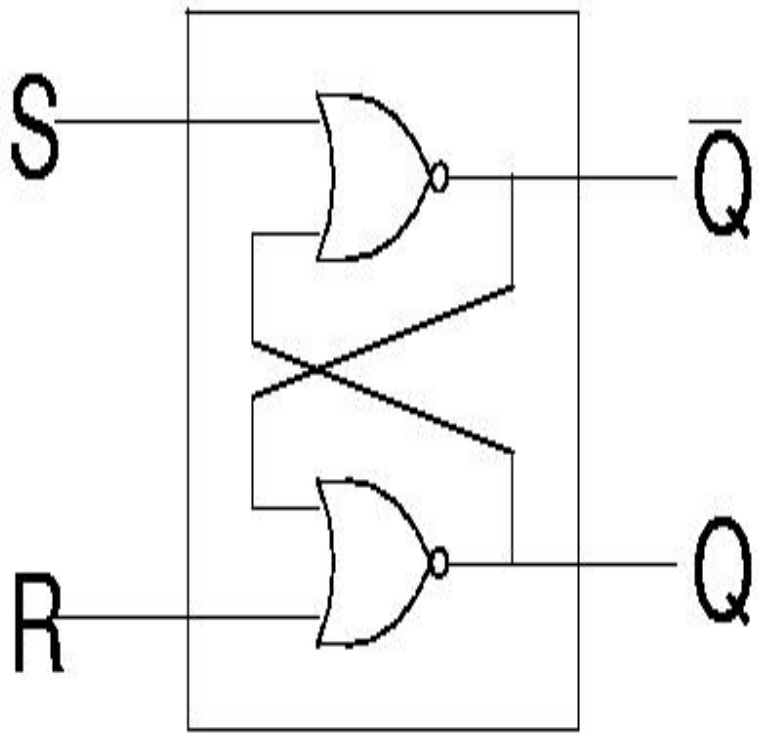
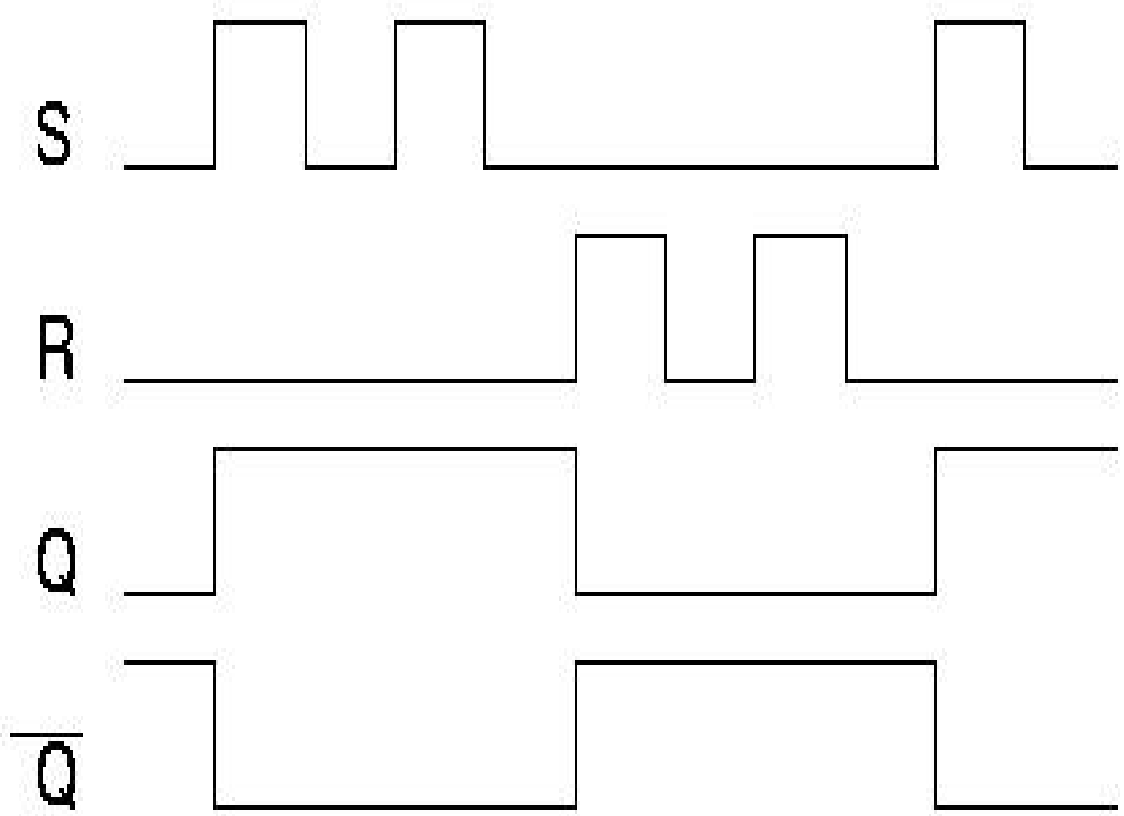


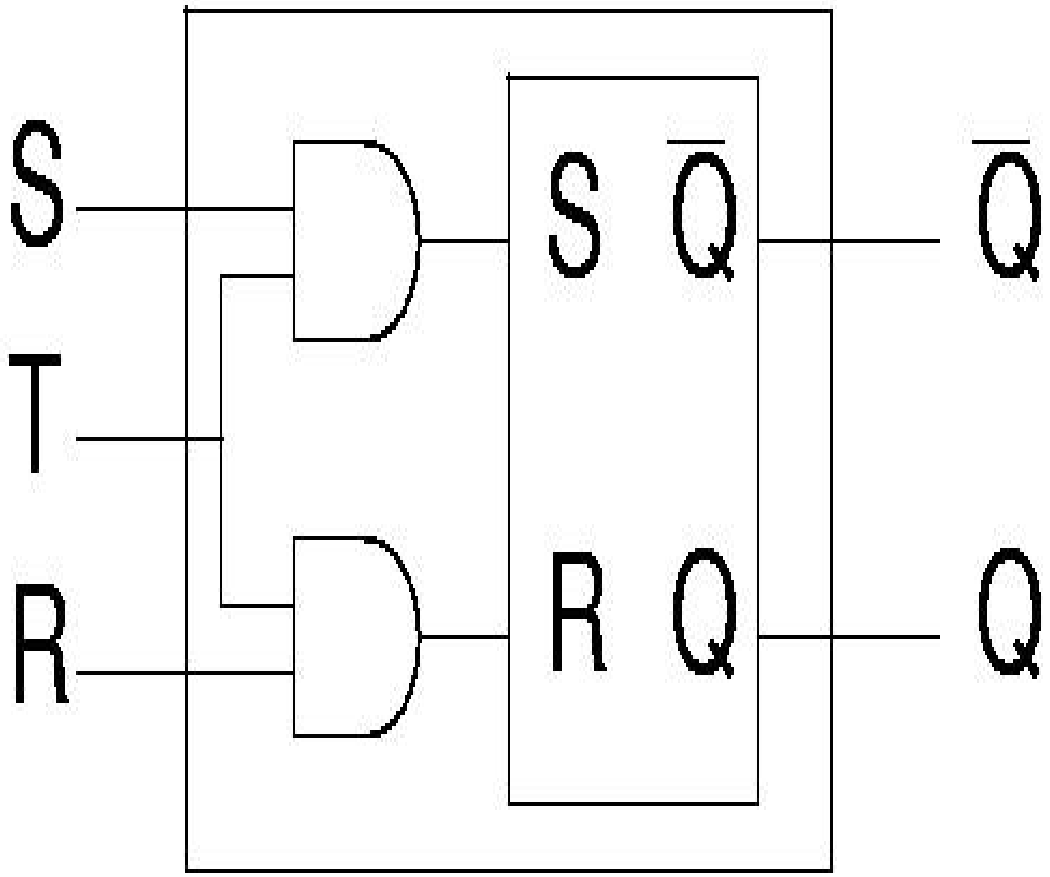
table de vérité

R	S	Q_i	Q_{i+1}
0	0	x	x
0	1	x	1
1	0	x	0
1	1	x	interdit

chronogramme



bascule RST



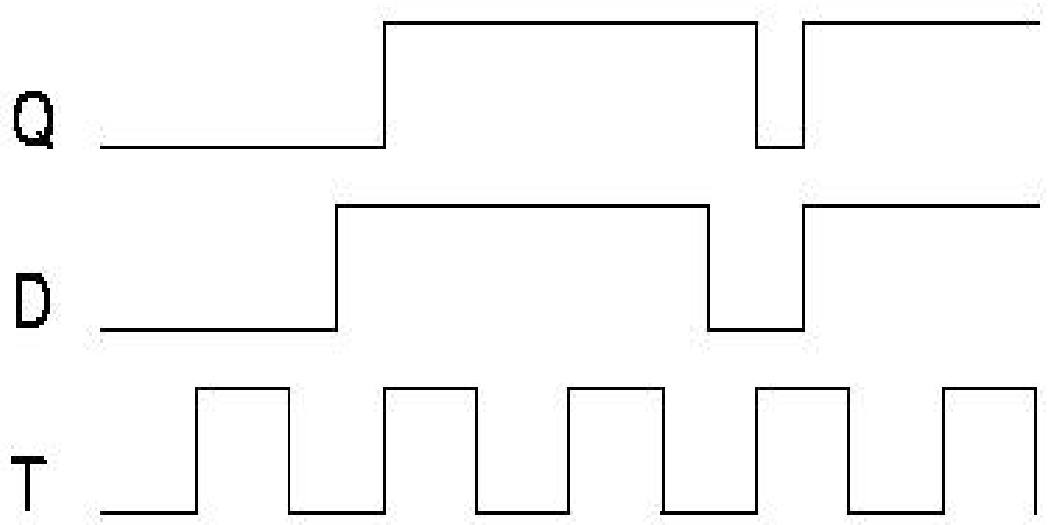
bascule RST

l'entrée d'horloge permet de valider les signaux R et S

– si T = 1, fonctionnement comme une bascule RS

– si T = 0, conservation de l'état courant

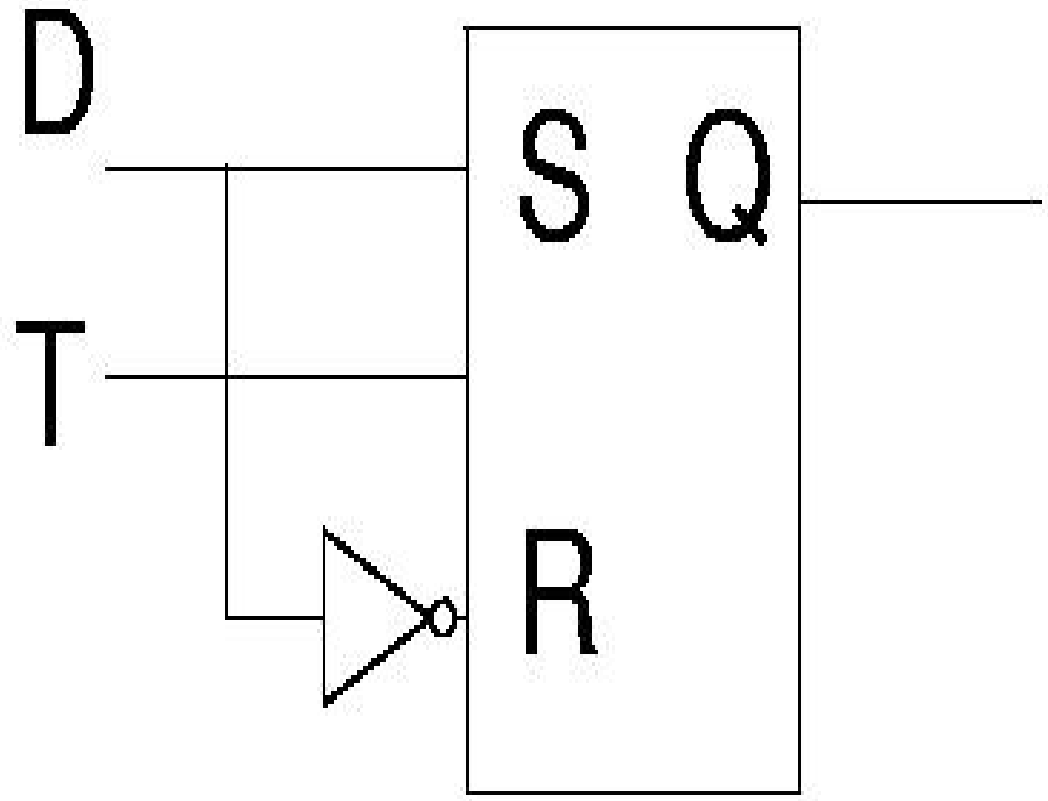
bascule D (latch)



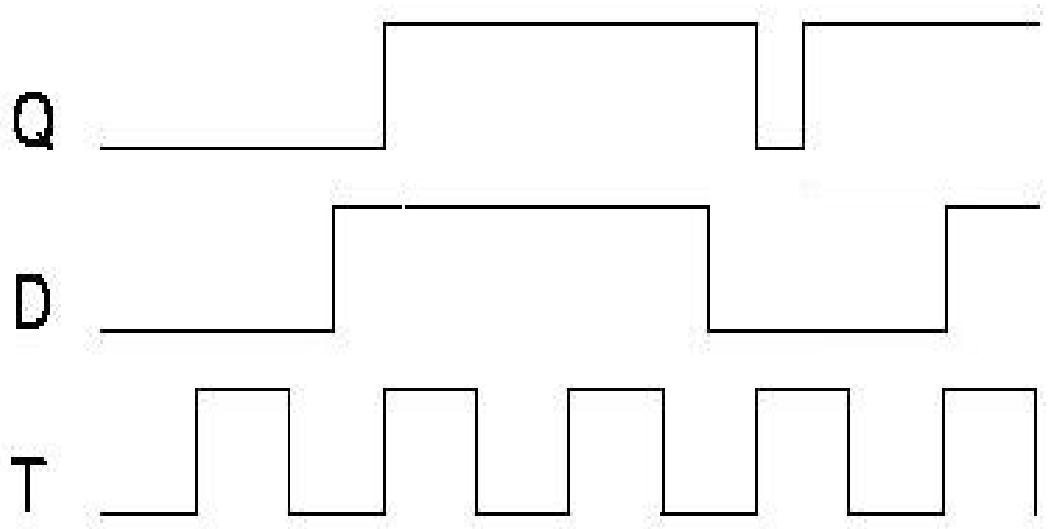
pas d'état instable

utilisée pour la conception des mémoires

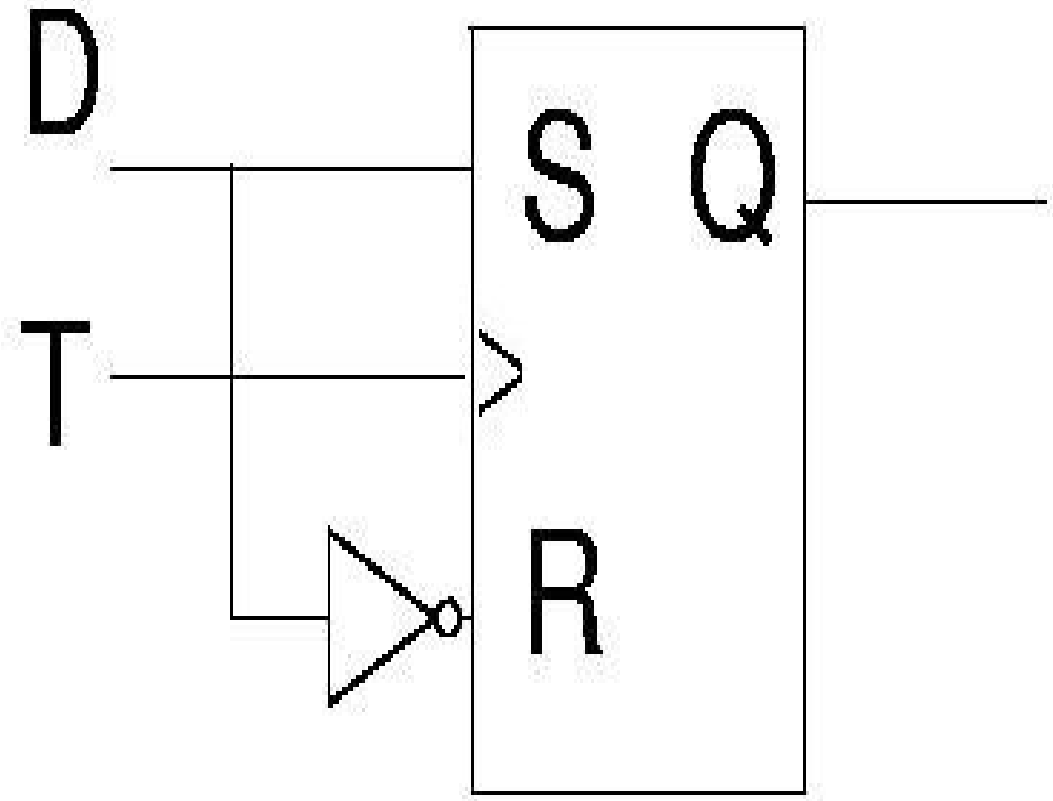
bascule D



bascule D sensible sur front d'horloge



bascule D sensible sur front d'horloge



bascule JK

J	K	Q_i	Q_{i+1}
0	0	x	x
0	1	x	0
1	0	x	1
1	1	x	$\overline{x}$

bascule JK