CHAPITRE EN4

Machines à états finis

SOMMAIRE

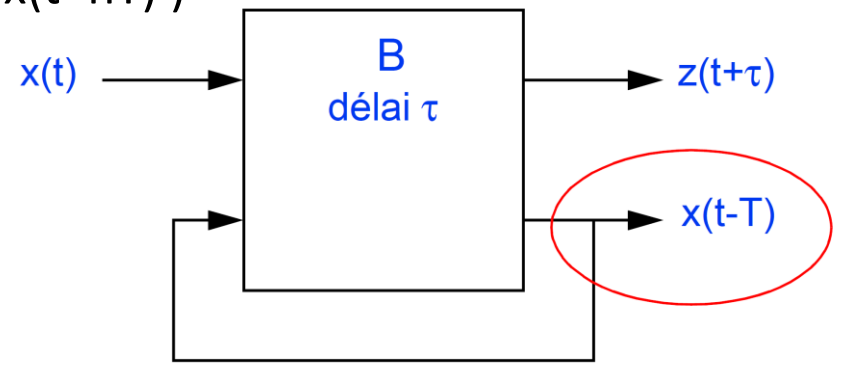
CLASSIFICATION DES CIRCUITS – Combinatoire vs Séquentiel

- **Circuit combinatoire:** les sorties du circuit à l'instant t ne dépendent que de l'état de ses entrées au même instant

$z(t) = A(x(t))$

- **Circuit séquentiel:** les sorties du circuit à l'instant t dépendent de l'état de ses entrées au même instant ainsi que de l'état de ses entrées aux instants précédents (t-nT) (effet de mémoire)

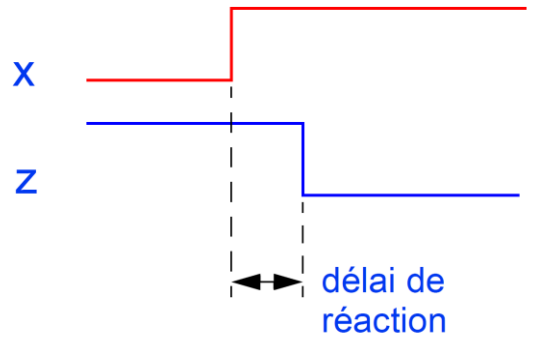
$z(t) = B(x(t), x(t-T), x(t-2T), ..., x(t-nT))$



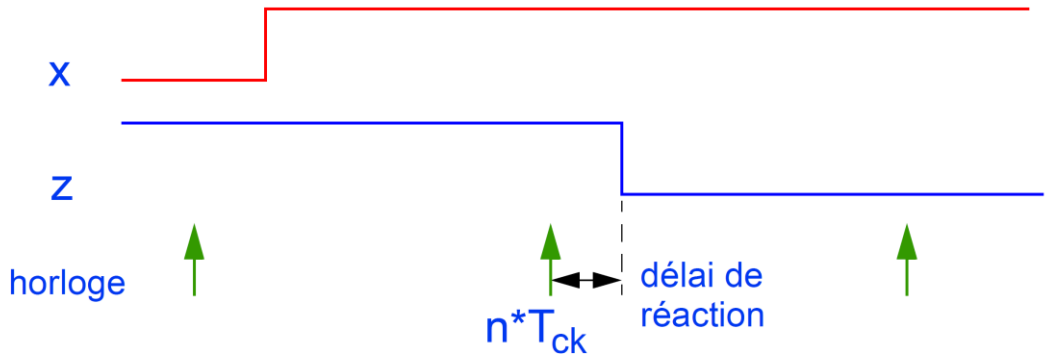
ceci n'est pas une sortie mais une variable interne

CLASSIFICATION DES CIRCUITS – *Synchrone vs Asynchrone*

- **Circuit asynchrone :** les sorties du circuit peuvent changer d'état à tout instant



- **Circuit synchrone :** les sorties du circuit ne peuvent changer d'état qu'à des instants particuliers (synchronisation par une horloge de période T_{ck})



DESCRIPTION D'UNE MEF – *Exemple 1*

I = {0, 1, 2, 3} c'est l'ensemble des valeurs en entrée (2 bits sur x)
O = {a, b} c'est l'ensemble des valeurs en sortie (1 bit sur z)
S = {S0, S1} c'est l'ensemble des états possibles (1 élément mémoire)

S(t+T) =S0 si S(t) = S0 et (x(t)=0 ou x(t)=2)
=S0 si S(t) = S1 et (x(t)=1 ou x(t)=3)
=S1 dans les autres cas

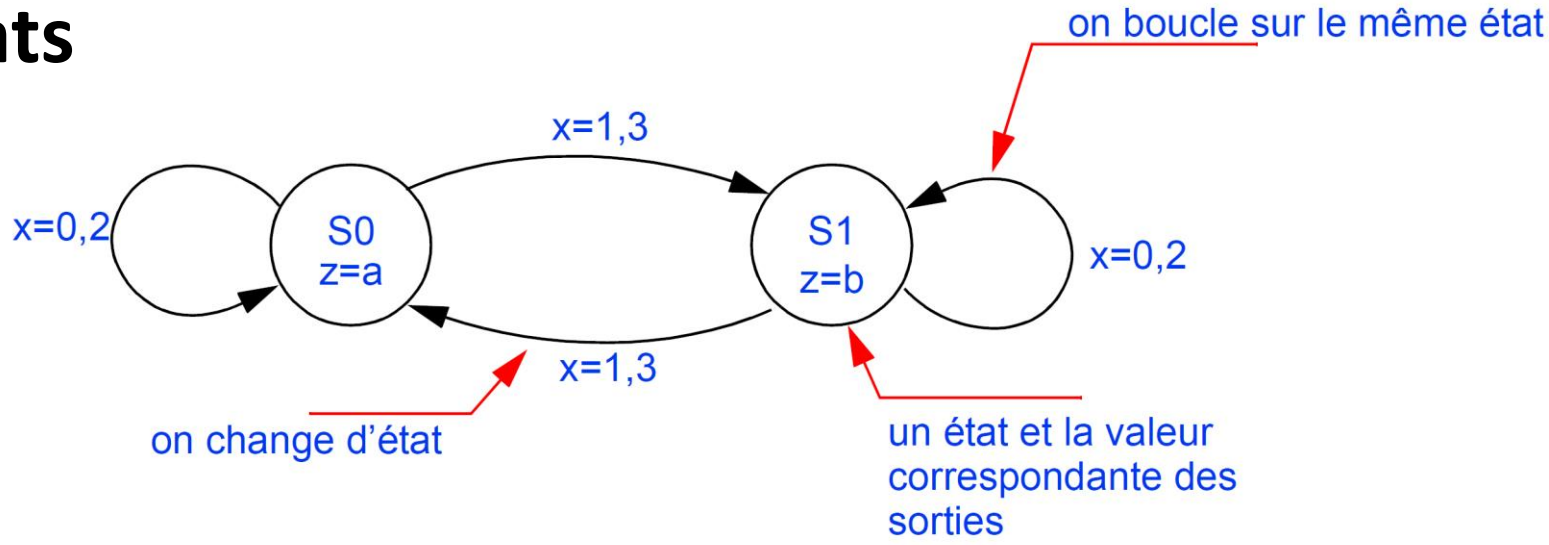
c'est la description de l'enchaînement des états: on connaît l'état suivant (t+T) à partir de l'état présent (t)

z(t) =a si S(t)=S0
=b si S(t)=S1

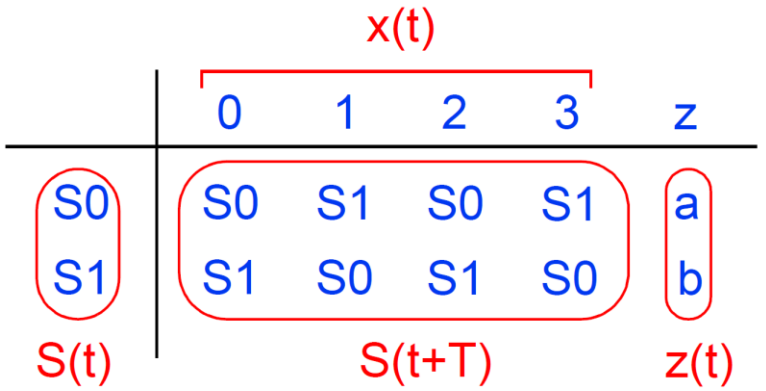
c'est la description de la valeur des sorties pour chaque état présent (t)

DESCRIPTION D'UNE MEF – *Graphe d'états & Table d'états*

- **Graphe d'états**



- **Table d'états**



Remarque: la sortie z(t) ne dépend que de S(t)

DESCRIPTION D'UNE MEF – *Exemple 2*

I = {a, b}

O = {p, q}

z(t)=q si [x(t-2T)=a et x(t-T)=b et x(t)=b]

=p autrement

Remarque : la sortie z(t) dépend de S(t) et de x(t)

DESCRIPTION D'UNE MEF – *Exemple 2*

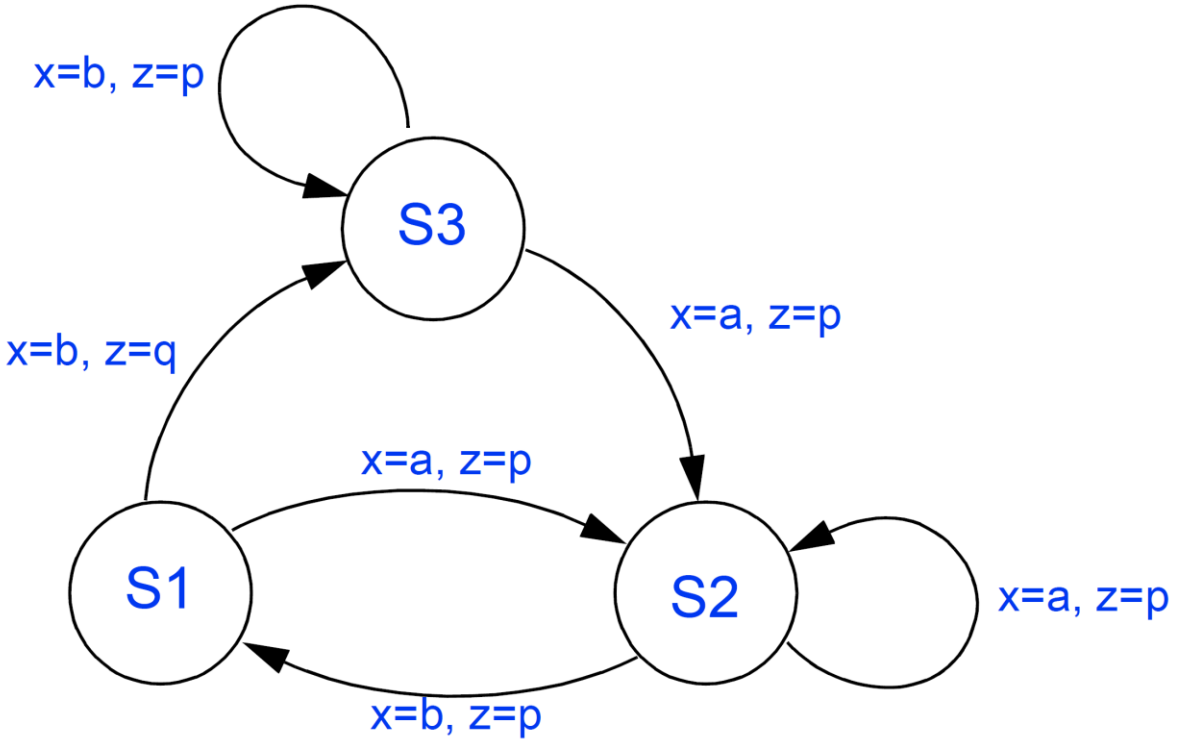
I = {a, b}

O = {p, q}

z(t)=q si [x(t-2T)=a et x(t-T)=b et x(t)=b]

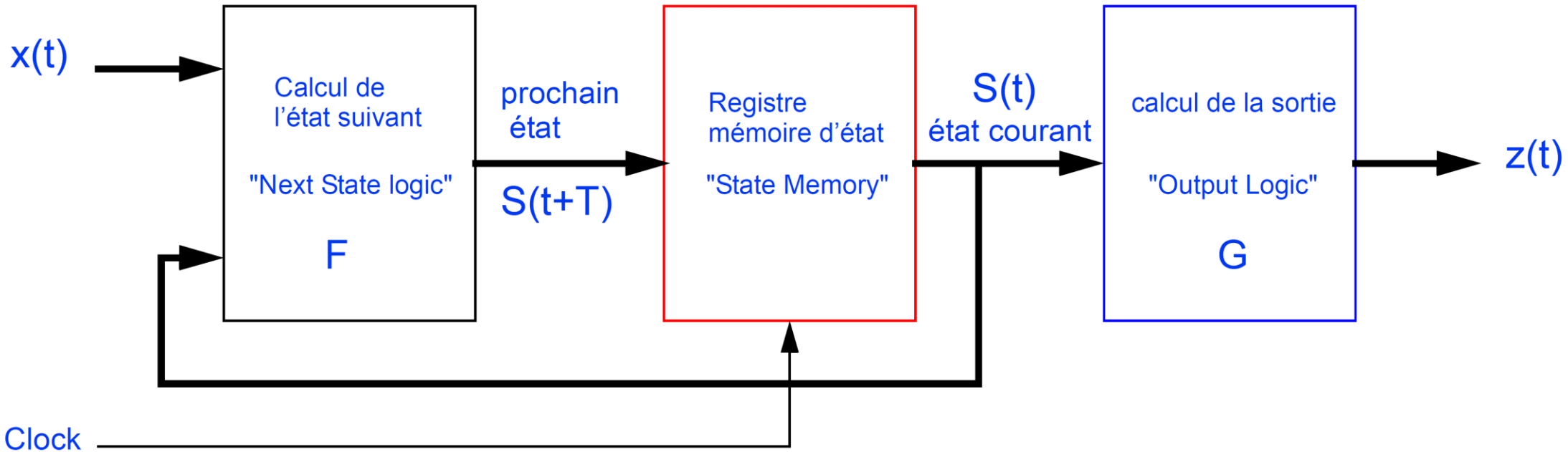
=p autrement

	a	b
S1	S2,p	S3,q
S2	S2,p	S1,p
S3	S2,p	S3,p



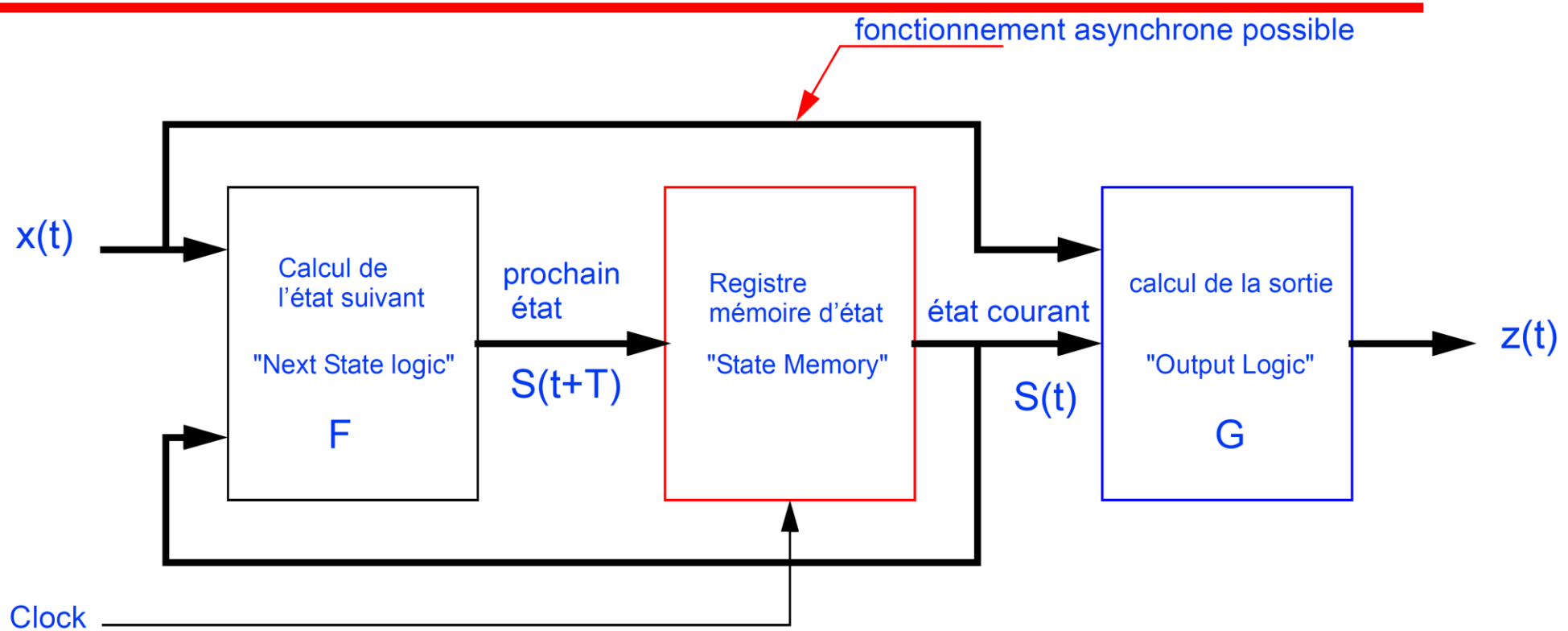
Remarque : la sortie z(t) dépend de S(t) et de x(t)

STRUCTURES D'UNE MEF – Machine de MOORE



- La sortie z(t) ne dépend que de S(t) (c.f. exemple 1)
- La sortie change d'état sur le front (ou le niveau) actif de l'horloge
- Un élément de mémoire = une bascule
- Une bascule stocke 2 états
- Pour N états, il faut M bascules tel que $N \leq 2^M$

STRUCTURES D'UNE MEF – Machine de MEALY



- La sortie z(t) dépend de S(t) et de x(t) (c.f. exemple 2)
- La sortie change d'état sur le front (ou le niveau) actif de l'horloge ainsi que sur un changement de x(t): fonctionnement asynchrone possible

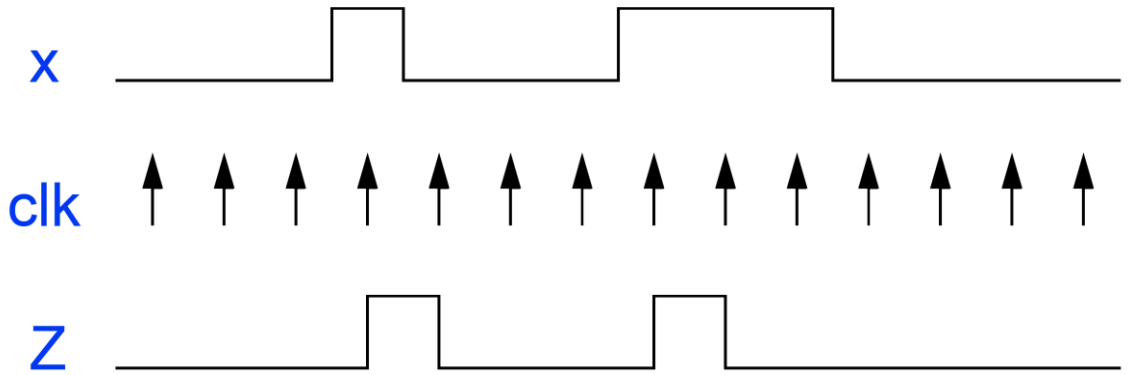
SYNTHÈSE D'UNE MEF – *Spécifications, graphe d'états*

I = {0,1}

O = {0,1}

z(t)=1 si [x(t-T)=0 et x(t)=1]

=0 autrement



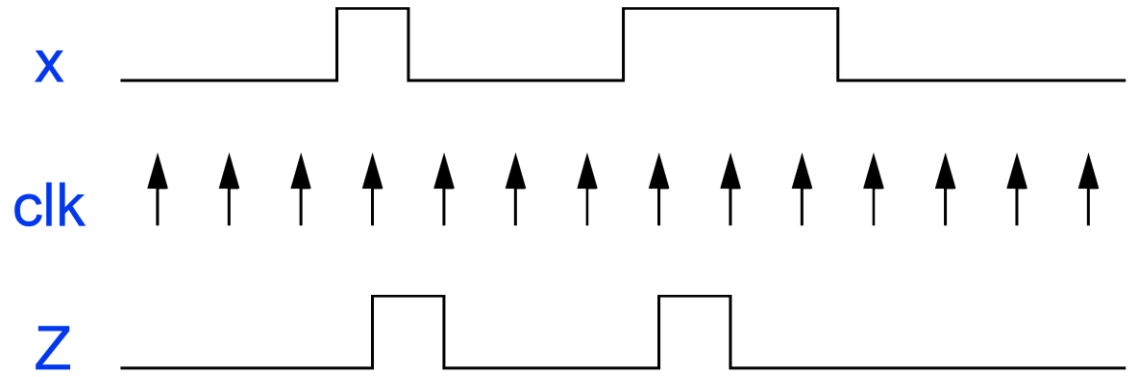
SYNTHÈSE D'UNE MEF – *Spécifications, graphe d'états*

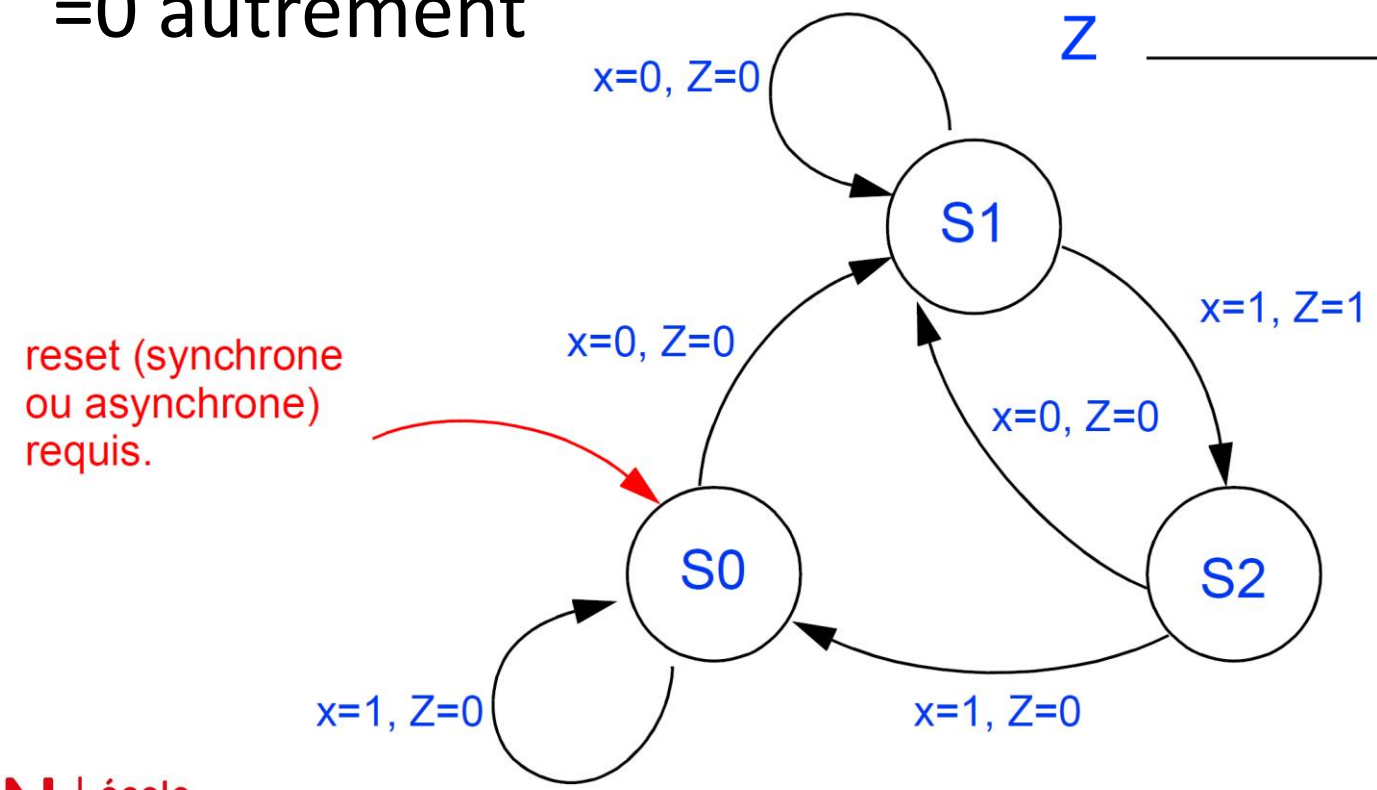
I = {0,1}

O = {0,1}

z(t)=1 si [x(t-T)=0 et x(t)=1]

=0 autrement





Machine de MOORE

SYNTHÈSE D'UNE MEF – *Mémoire d'états*

- **Quelle taille pour la mémoire d'états?**

  Un état va être représenté en mémoire par une combinaison binaire:

  le registre doit comporter au moins N bascules telles que nombre d'état <= 2^N

  Notre exemple: 2 bascules

- **Comment gérer les états non utilisés?**

  Notre exemple: 2^2 états possibles - 3 états utilisés = 1 états non utilisés = danger potentiel

  Stratégie à coût minimal: on considère que les états non utilisés ne peuvent jamais survenir (!), l'état suivant est donc indifférent

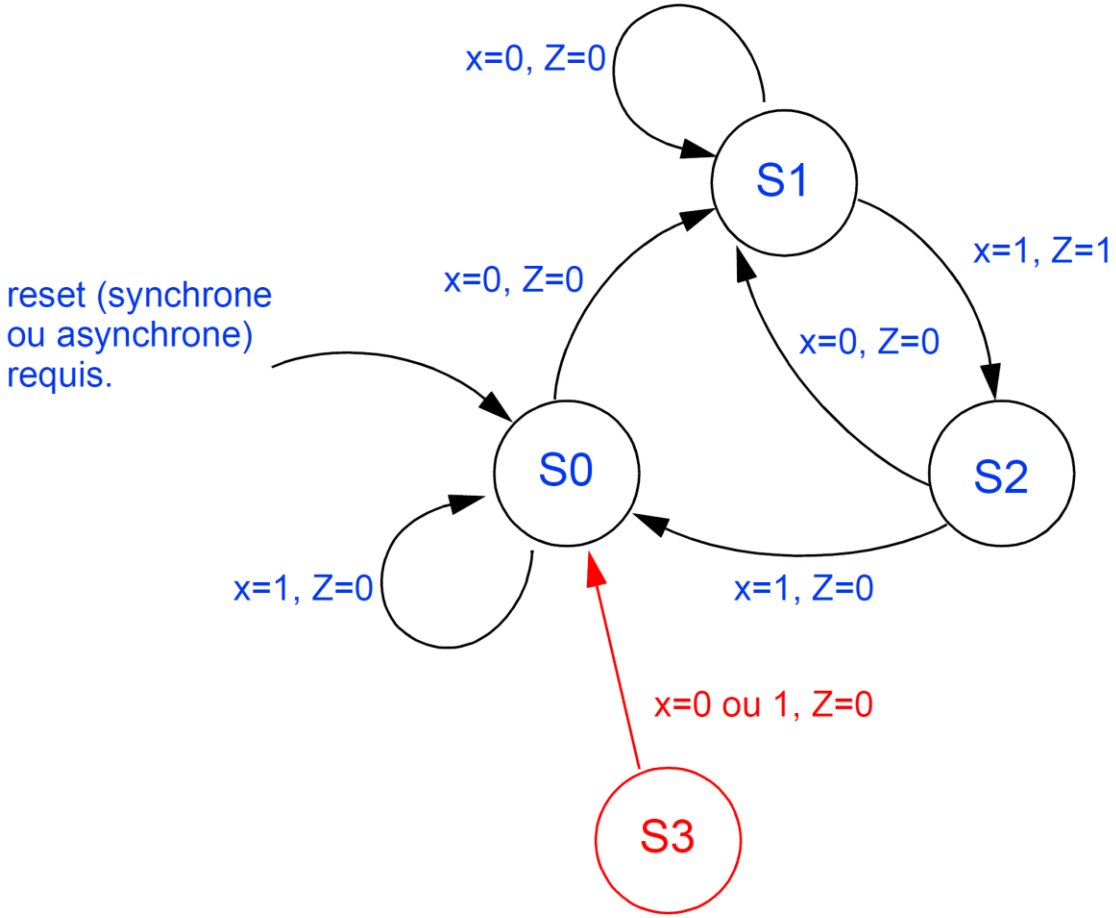
  => simplification possible des équations par la suite

  Stratégie à risque minimal: on affecte à chaque état non utilisé un état suivant

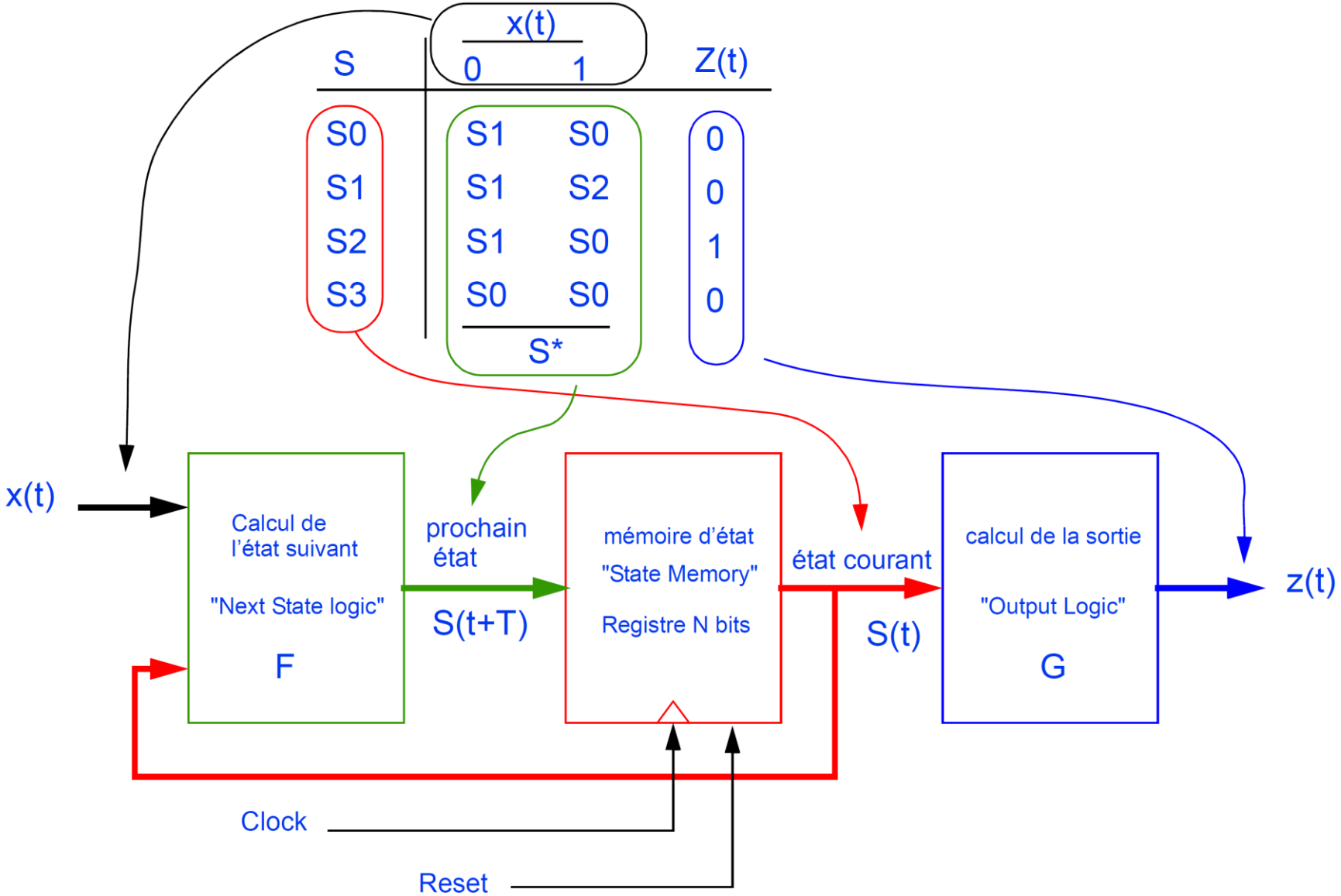
  adéquat à déterminer en fonction de l'application

SYNTHÈSE D'UNE MEF – Gestion des états non utilisés

Notre exemple: l'état non utilisé (S3) pourrait pointer vers "S0"



SYNTHÈSE D'UNE MEF – Correspondance graphe-synoptique



SYNTHÈSE D'UNE MEF – *Equations logiques de F et G*

Dans cet exemple:
états codés en
binaire naturel

S (Q1-Q0)		x = 0	x = 1	Z
S0	0 0	0 1	0 0	0
S1	0 1	0 1	1 0	0
S2	1 0	0 1	0 0	1
S3	1 1	0 0	0 0	0

S* (D1-D0)

Bloc G

S (Q1-Q0)	Z
0 0	0
0 1	0
1 0	1
1 1	0

Bloc F: 2 équations

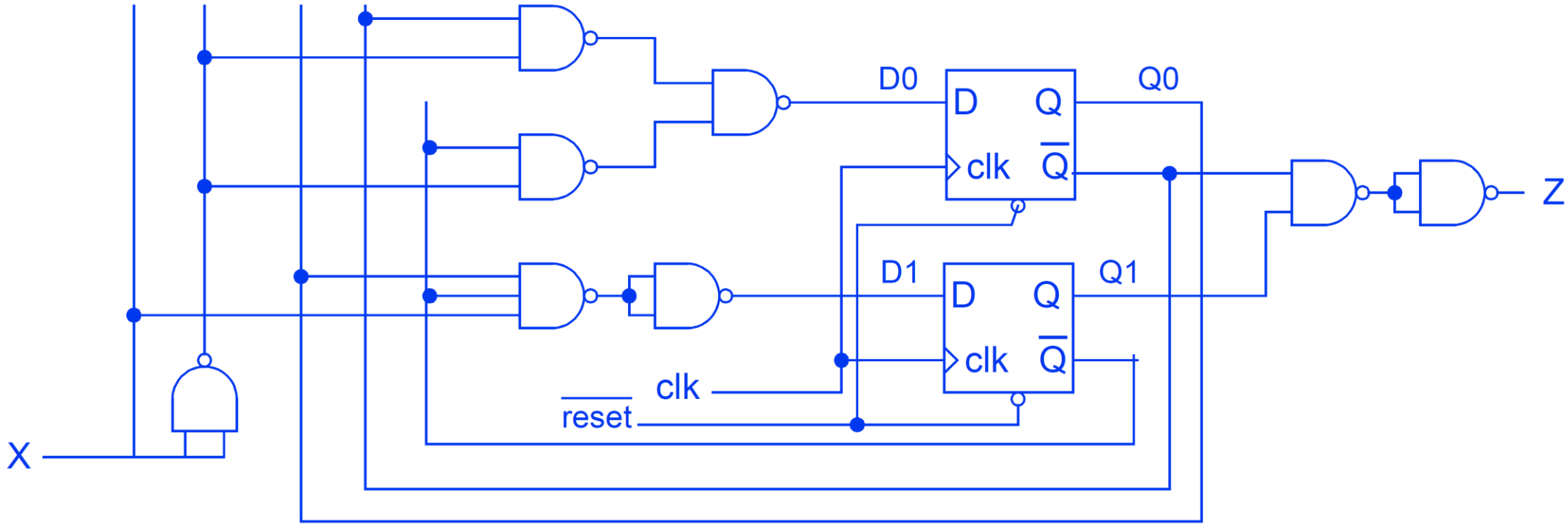
S (Q1-Q0)	x = 0	x = 1
0 0	0	0
0 1	0	1
1 0	0	0
1 1	0	0

S* (D1)

S (Q1-Q0)	x = 0	x = 1
0 0	1	0
0 1	1	0
1 0	1	0
1 1	0	0

S* (D0)

SYNTHÈSE D'UNE MEF – Schéma



SYNTHÈSE D'UNE MEF – *Langage comportemental (VHDL)*

```
entity mef2 is

        port(
                clk     : in    std_logic;
                x       : in    std_logic;
                reset   : in    std_logic;
                z       : out   std_logic
        );

end entity;

architecture rtl of mef2 is

        -- Build an enumerated type for the state machine
        type state_type is (s0, s1, s2, s3);

        -- Register to hold the current state
        signal state   : state_type;
```

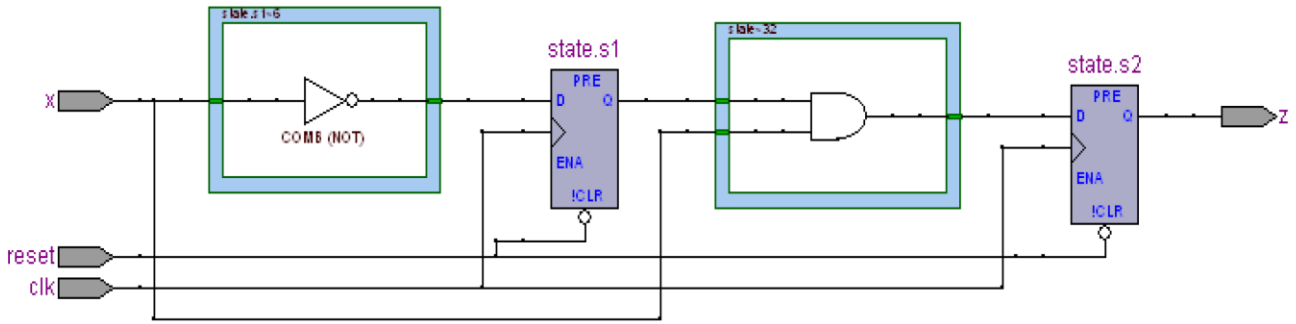
VHDL: **V**ery high speed integrated circuits **H**ardware **D**escription **L**anguage

SYNTHÈSE D'UNE MEF – Langage comportemental (VHDL)

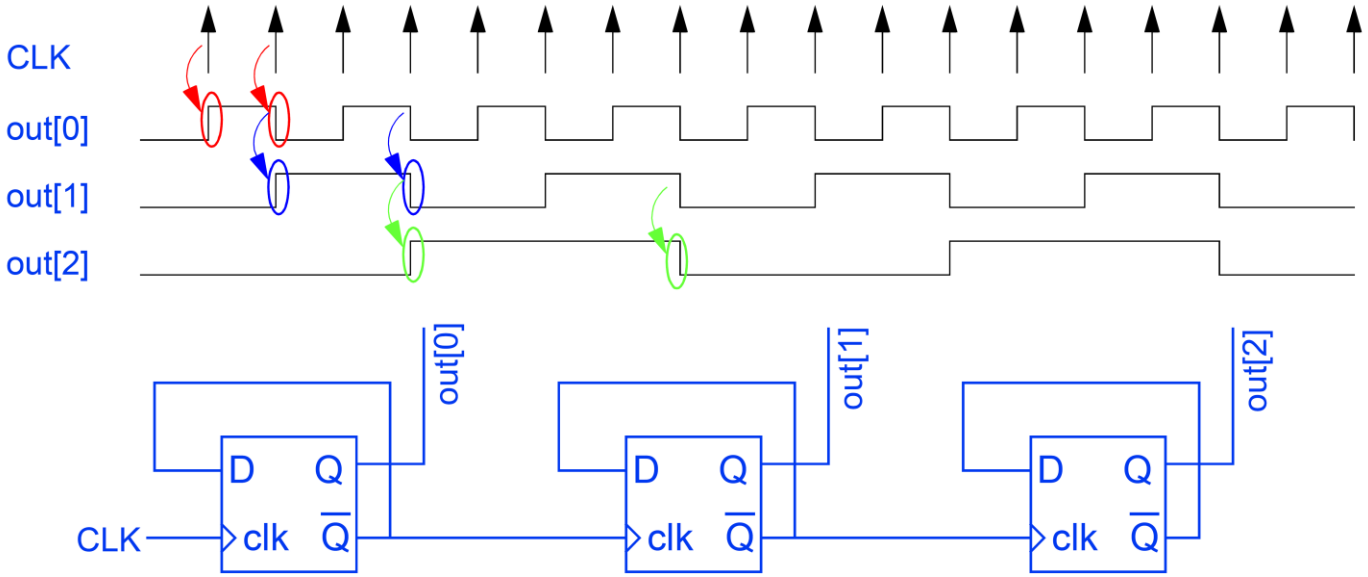
```
begin

    -- Logic to advance to the next state
    process (clk)
    begin
        if reset = '1' then
            state <= s0;
        elsif (rising_edge(clk)) then
            case state is
                when s0=>
                    if x = '1' then
                    state <= s0;
                    else
                    state <= s1;
                    end if;
                when s1=>
                    if x = '1' then
                    state <= s2;
                    else
                    state <= s1;
                    end if;
                when s2=>
                    if x = '1' then
                    state <= s0;
                    else
                    state <= s1;
                    end if;
                when s3 =>
                    state <= s0;

            end case;
        end if;
    end process;
```

SYNTHÈSE D'UNE MEF – *Langage comportemental (VHDL)*

```
-- Output depends solely on the current state
process (state)
begin
        case state is
                when s0 =>
                        z <= '0';
                when s1 =>
                        z <= '0';
                when s2 =>
                        z <= '1';
                when s3 =>
                        z <= '0';
        end case;
end process;

end rtl;
```



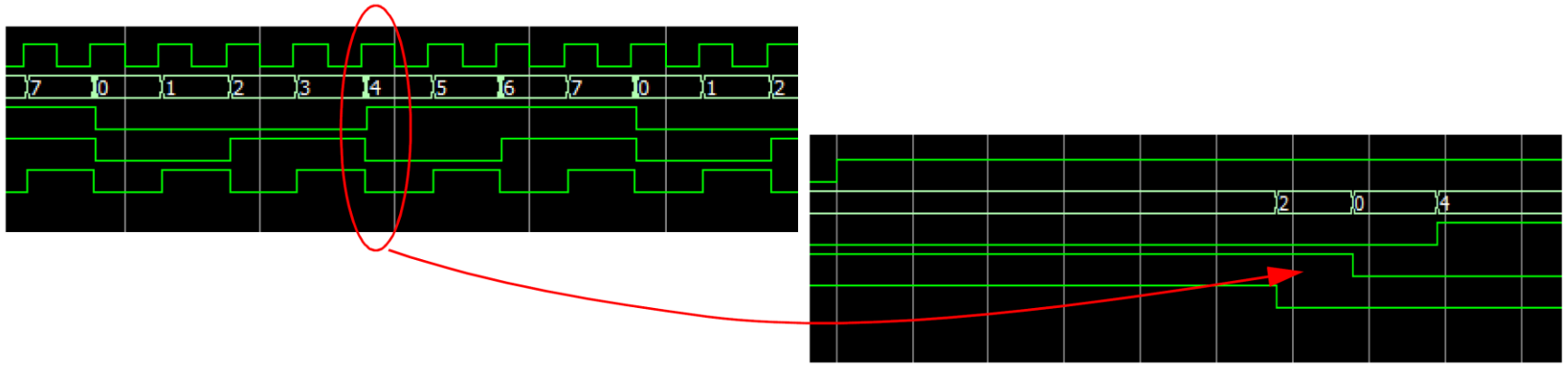
LES COMPTEURS – Compteurs asynchrones



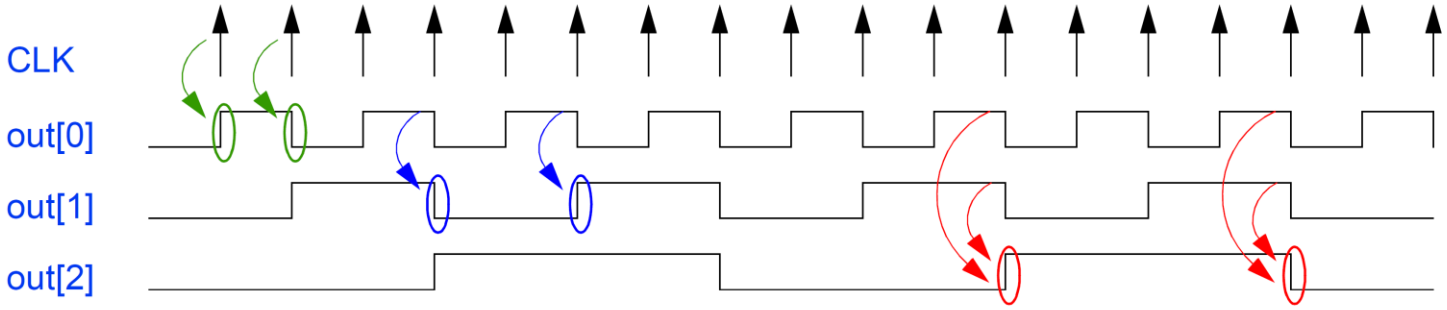
Bascule D avec D relié à Q = bascule T (Toggle)

clk relié à Q = décompteur

Glitches!



LES COMPTEURS – Compteurs synchrones



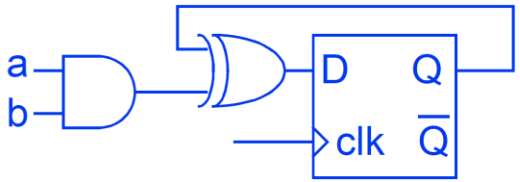
out[0] change d'état à chaque front d'horloge (Toggle)

out[1] change d'état quand out[0] est à 1

out[2] change d'état quand out[1] et out[0] sont à 1

De façon générale:

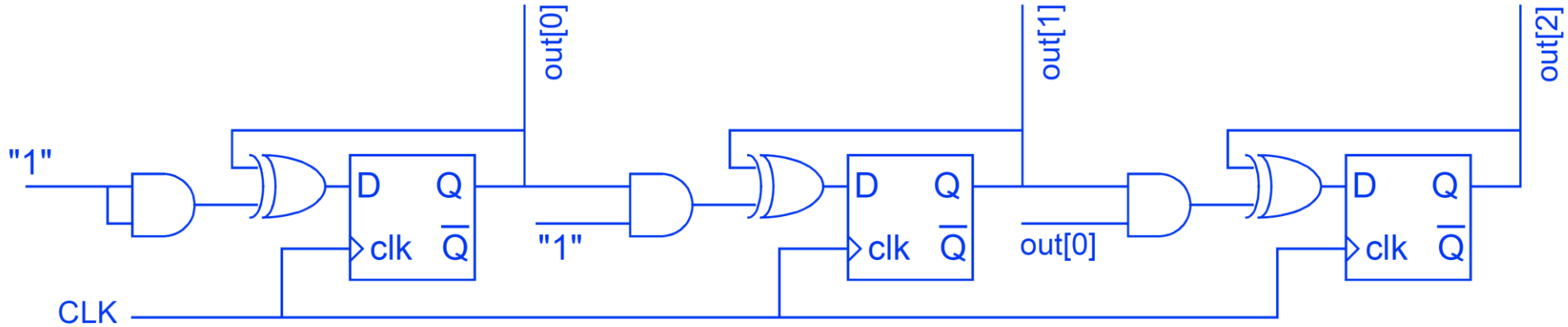
out[n] change d'état quand out[n-1] et out[n-2] sont à 1

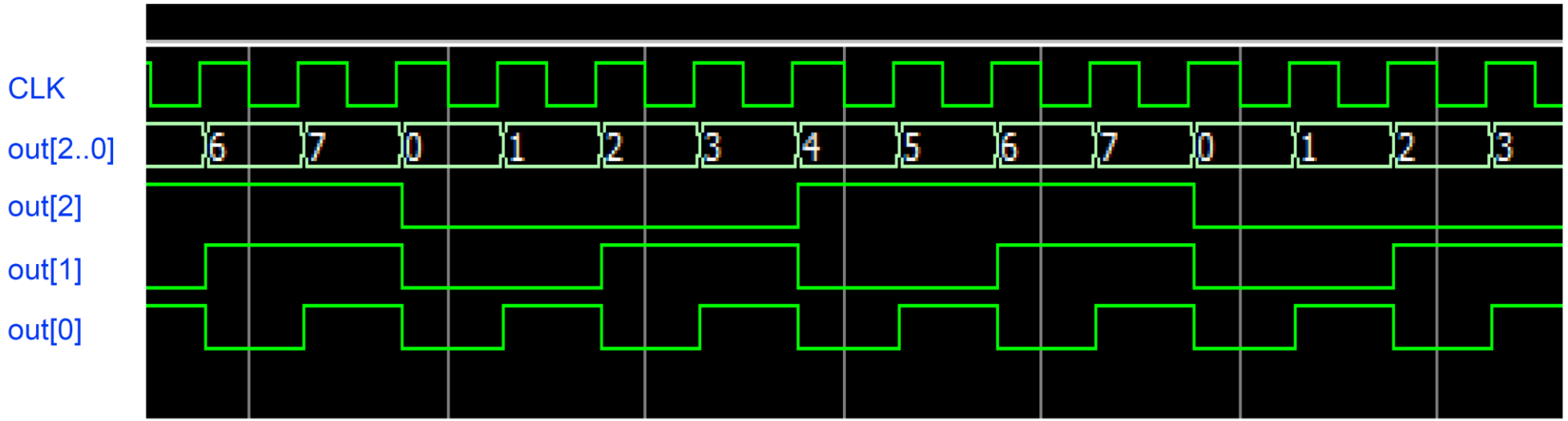


D = $\overline{Q}$ (Toggle) quand a=b=1

D = Q (pas de changement) sinon

LES COMPTEURS – Compteurs synchrones





TITRE