5. Machine à états

Les machines à état permettent de décrire des systèmes séquentiels dont l'**évolution** est plus **complexe** que les compteurs ou les registres.

Il est remarquable de constater que le concept relatif aux automates (au sens machines à état) se retrouvent désormais dans des applications diverses :

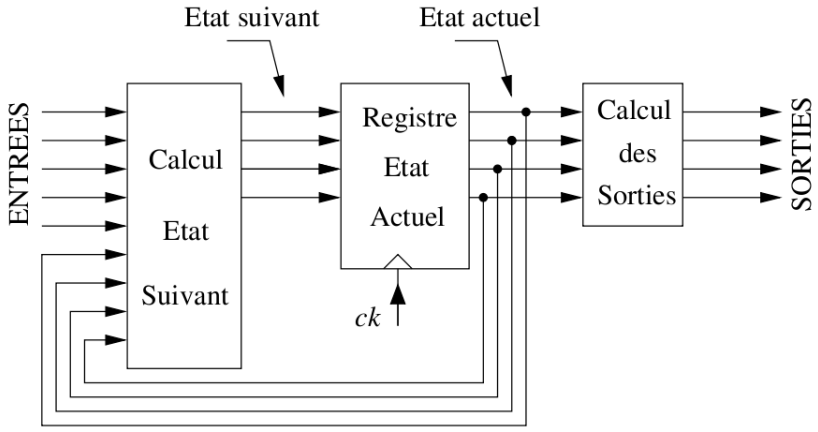
- circuits numériques ;
- automatismes industriels ;
- processeurs ou microcontroleurs ;
- prgrammes informatiques.

5.1. Modèles de machine à états

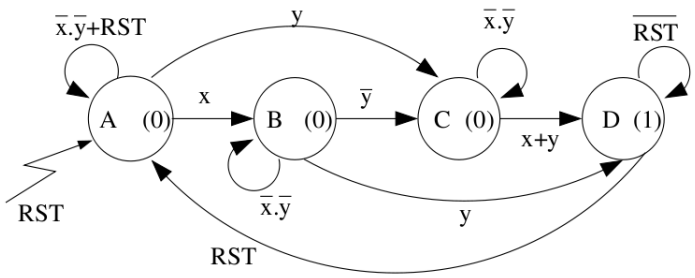
Pour représenter ces automates, qu'ils soient matériel ou logiciel, il existe deux architectures différentes : la machine de **MOORE** (synchrone) et la machine de **MEALY** (asynchrone).

5.1.1 Modèle de Moore - synchrone

Dans une **machine de Moore**, la sortie ne dépend que de l'état de la machine. Les sorties sont alors synchrones avec les transitions d'état et les fronts d'horloge.



Voici un exemple de diagramme d'états que l'on pourrait associé à une machine de Moore :

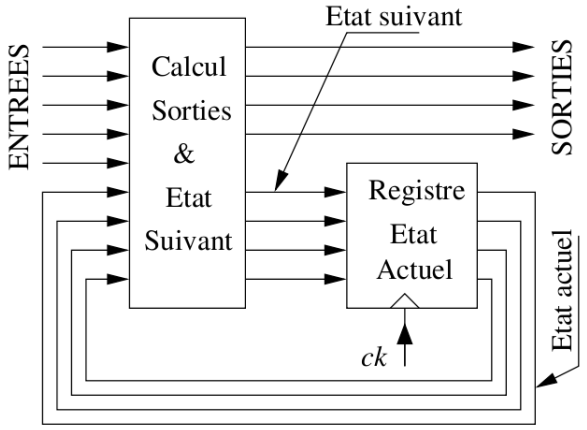


Dans une telle machine, les sorties étant fonction exclusivement de l'état du système, leurs valeurs sont indiquées dans les cercles.

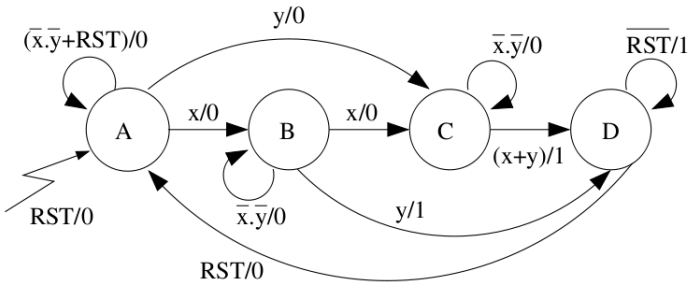
5.1.2 Modèle de Mealy - asynchrone

Dans une **machine de Mealy**, la sortie est calculée en fonction de l'état présent et de la valeur présente des entrées : les sorties peuvent alors changer immédiatement après un changement des entrées, indépendamment de l'horloge.

Ces systèmes sont alors totalement **asynchrones**. Ils sont plus rapides que les machines de Moore, mais beaucoup plus instables et difficiles à concevoir.



Voici un exemple de diagramme d'états que l'on pourrait associé à une machine de Mealy :



5.2. Conception et synthèse de machines à états

Afin d'illustrer la synthèse d'une machine à états, nous prendrons comme exemple un **détecteur de séquence**. Un tel système est très souvent utilisé autour de nous : digicode, détecteur d'erreurs CRC...

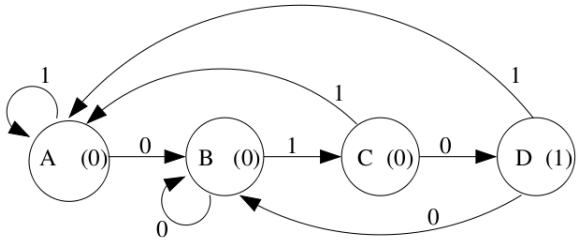
Le système à réaliser a une entrée E et une sortie S. E reçoit des bits en série, cadencés par une horloge.

Chaque fois que la séquence *"010"* se présente en entrée, la sortie S devra passer à '1' dès le dernier bit détecté, puis retourner à '0' au bit suivant, quel qu'il soit.

5.2.1 Diagramme d'états

La première étape lors de la conception d'une machine à états est la compréhension du cahier des charges, qui passe souvent par la représentation sous forme d'un **diagramme d'états**.

Dans le cas précédent, l'étude du cahier des charges amène à réaliser une machine à *4 états distincts*. On peut associer le fonctionnement décrit précédemment au diagramme d'états suivant :



Il est ensuite nécessaire d'utiliser un codage particulier pour les états, qui seront stockés dans un registre d'état.

4 états nécessitent ici **2 bits d'état** (donc 2 bascules).

On pourra utiliser un codage simple des états, par exemple le code de Gray : $A = 00$, $B = 01$, $C = 11$ et $D = 10$.

5.2.2 Synthèse structurelle

Pour pouvoir plus facilement trouver la structure complète du système, c'est à dire la structure de la fonction combinatoire de calcul de l'état suivant ainsi que celle du calcul des sorties, il est préférable de passer par la **table des transitions**.

Actuel	Suivant		Sortie
	0	1	S
A	B	A	0
B	B	C	0
C	D	A	0
D	B	A	1

Actuel	Suivant		Sortie
	0	1	S
00	01	00	0
01	01	11	0
11	10	00	0
10	01	00	1

Dans le cas de l'utilisation de bascules D, les valeurs des entrées D des bascules sont directement donnés par les codes de l'état suivant.

On en déduit alors (après synthèse d'un système combinatoire et simplification) les expressions de D_1 et de D_2 en fonction de Q_1 , Q_2 et E ainsi que l'expression de S.

$$D_1 = E \cdot Q1 \cdot Q2 + E \cdot Q1 \cdot Q2$$

$$D_2 = E \cdot Q1 + E \cdot Q2 + Q1 \cdot Q2$$

$$S = Q1 \cdot Q2$$

5.2.3 Synthèse comportementale (VHDL)

L'autre possibilité, à partir du diagramme d'états, est de décrire le **comportement du système** à l'aide d'un **langage de description de haut niveau**, tel que le VHDL.

La traduction du diagramme d'états précédent est donnée par la suite.

```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity detect_seq is
  port
  (
    E, CLK: in STD_LOGIC;
    s:      out STD_LOGIC
  );
end detect_seq;

architecture mach_etat of detect_seq is
signal ETAT: STD_LOGIC_VECTOR(1 downto 0);

mach: process (CLK)
begin
  if (CLK'event and CLK='1') then
    case ETAT is
    when "00" =>
      if E='0' then ETAT <= "01";
      else ETAT <= "00";
      end if;
    when "01" =>
      if E='0' then ETAT <= "01";
      else ETAT <= "11";
      end if;
    when "10" =>
      if E='0' then ETAT <= "01";
      else ETAT <= "00";
      end if;
    when others =>
      if E='0' then ETAT <= "10";
      else ETAT <= "00";
      end if;
    end case;
  end if;
end process mach;

S <= '1' when ETAT = "10" else '0';

end mach_etat;
```