Outils de Simulation Logico-Temporelle

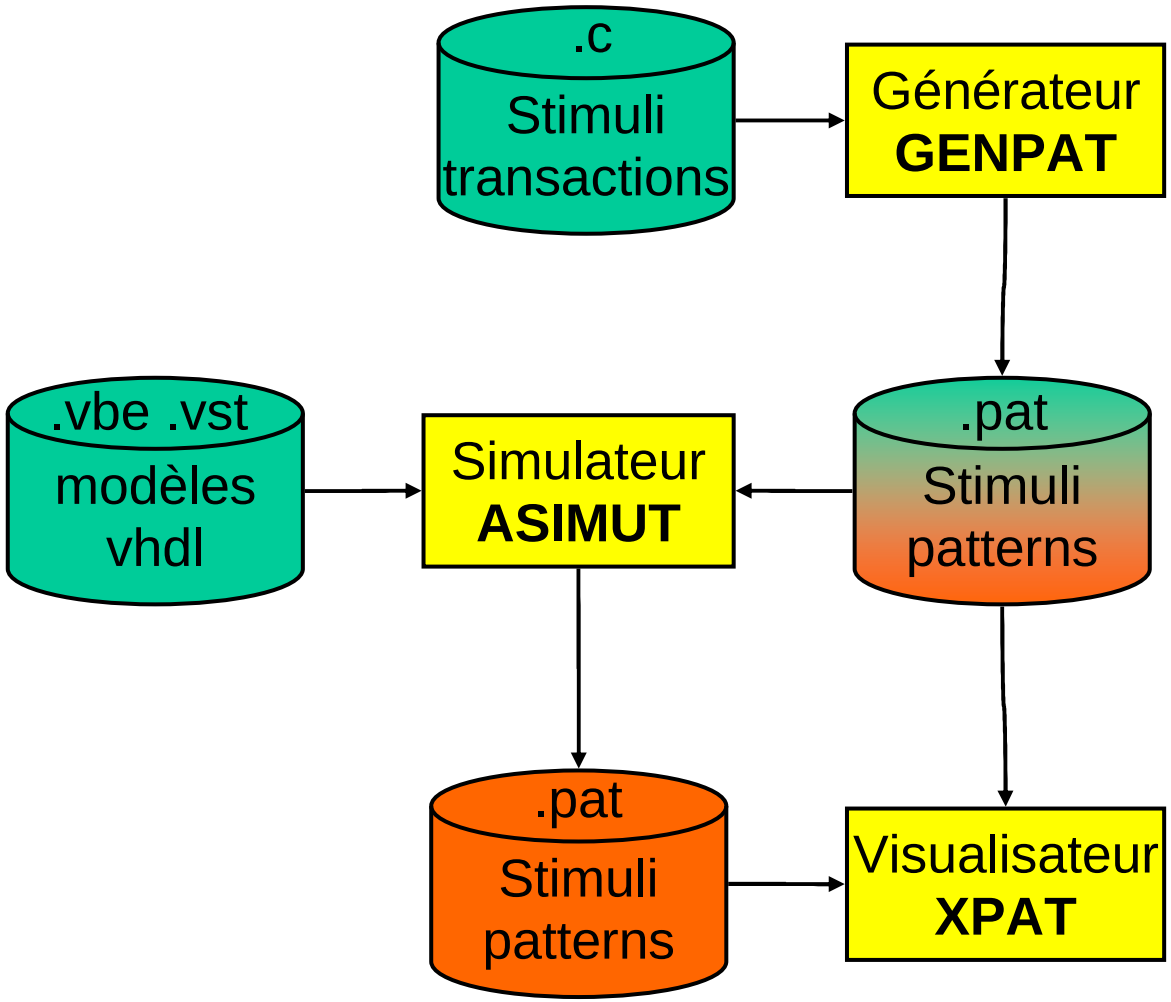
asimut
genpat
xpat

Université Pierre et Marie Curie
Master ACSI
Outils pour la Conception VLSI

Plan du cours

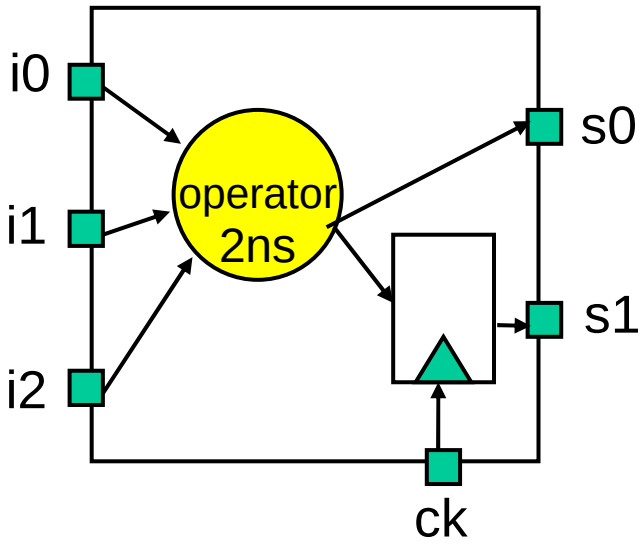
- Aperçu de la chaine de compilation
- Principe de la simulation à événements discrets
  - Modélisation
  - Exemple de simulation d'un circuit combinatoire
  - Extension aux bus et aux registres
  - simulation sans délai
- Le format pat
- Le simulateur ASIMUT
- Le générateur de patterns GENPAT
- Le visualisateur de patterns XPAT

Chaine de simulation

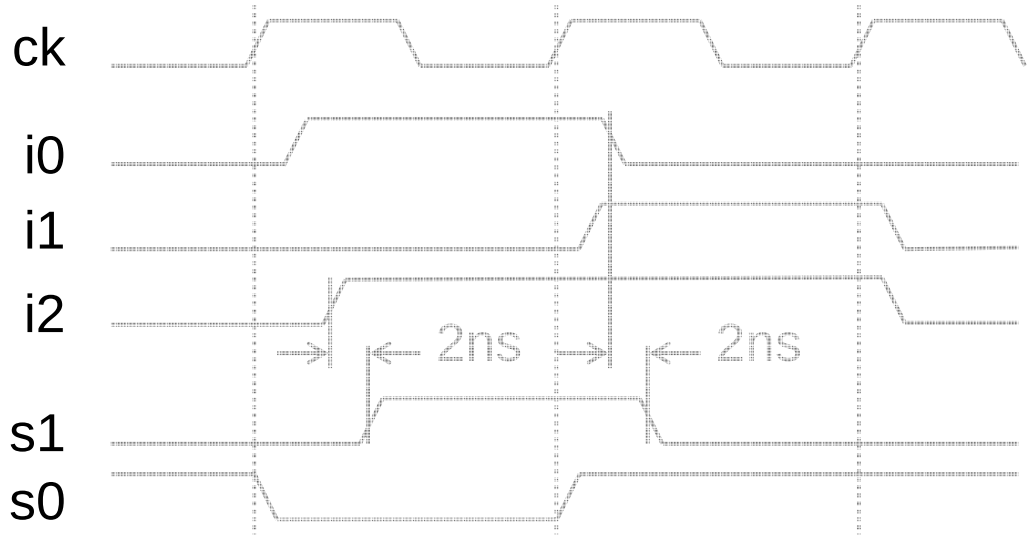


simulation de circuits numériques

Les simulateurs à événements discrets permettent de simuler des systèmes matériels constitués d'un ensemble de composants matériels interconnectés par des signaux.



circuit numérique



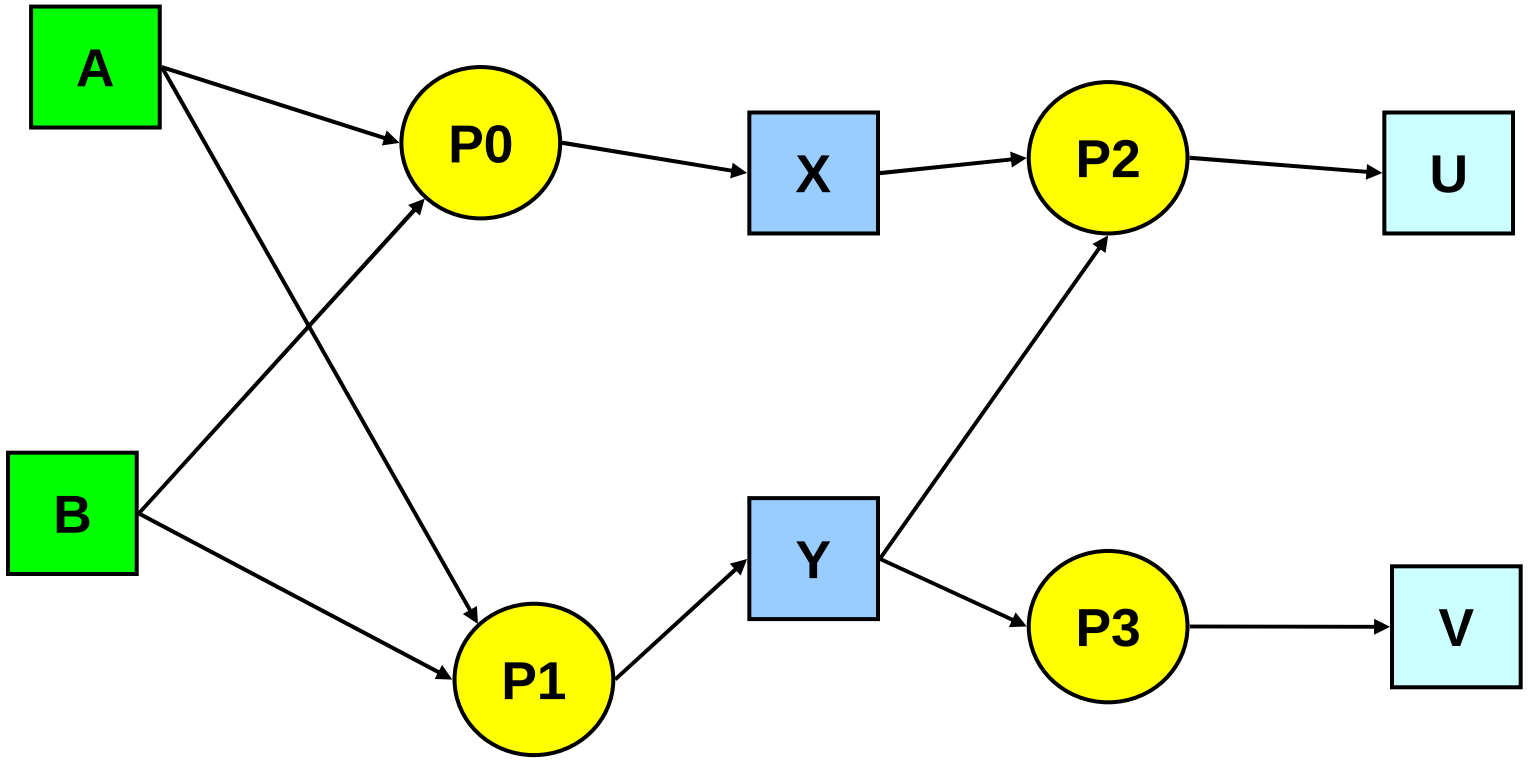
chronogramme des signaux

Plan du cours

- Aperçu de la chaine de compilation
- Principe de la simulation à événements discrets
  - Modélisation
  - Exemple de simulation d'un circuit combinatoire
  - Extension aux bus et aux registres
  - simulation sans délai
- Le format pat
- Le simulateur ASIMUT
- Le générateur de patterns GENPAT
- Le visualisateur de patterns XPAT

Modélisation du circuit

- Le système à simuler est modélisé comme un ensemble de processus interconnectés par des signaux



Processus

- Un processus est une description comportementale qui définit comment des événements sur les signaux d'entrées provoquent des événements sur les signaux de sorties.

- Le comportement peut être modélisé par
  - un programme séquentiel
  - une expression Booléenne multi-niveaux
  - une table de vérité
  - un arbre de décision binaire (BDD)

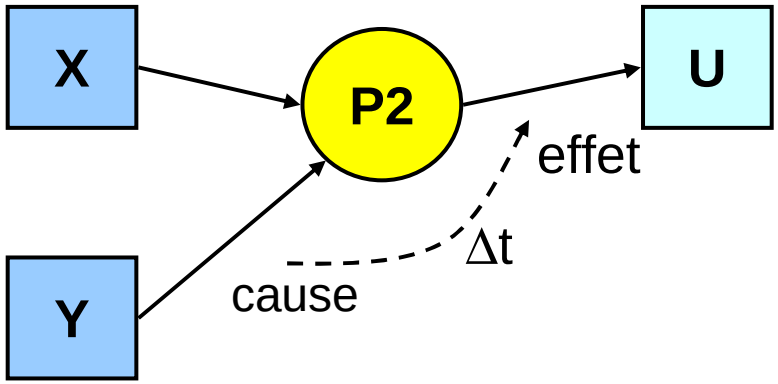
Signaux

- Les signaux représentent l'état des fils reliant les processus.
  - les valeurs possibles dépendent du simulateur
    - **0**
    - **1**
    - **U indéfinie**

    } **ASIMUT**

    - haute impédance
    - 0 faible
    - 1 faible
    - conflit électrique
    - …
- Un état du système est défini par l'ensemble des valeurs de signaux du système.

Principe de causalité



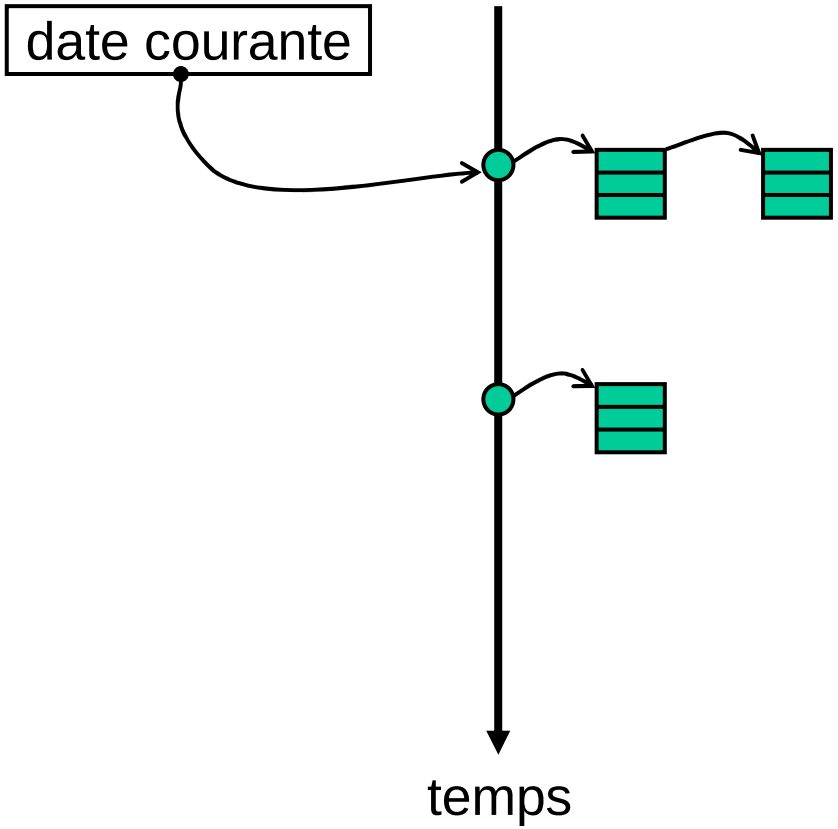
- Le changement d'un signal de sortie est toujours précédé par le changement d'un signal d'entrée :
  ➔ **c'est le principe de causalité**

  Ici, c'est de changement d'état de Y qui peut provoquer un changement d'état de U.
- On appelle **liste de sensibilité** d'un processus le sous-ensemble de ses signaux d'entrée qui oblige la réévaluation de ce processus.

Échéancier

Un événement sur un signal de la liste de sensibilité provoque une transaction à faire dans le futur sur le signal de sortie.

- Une transaction est définie par :
  - une date
  - un nom de signal
  - une valeur
- Les transactions sont stockées dans un échéancier.
- Un échéancier est défini par :
  - la liste ordonnée par date de toutes les transactions
  - la date courante



Structure de données

P : l'ensemble de tous les processus

S : l'ensemble de tous les signaux

E : l'échéancier = liste des transactions par ordre chronologique + date courante

Processus :

- les comportements, *-- pour l'évaluation*
- la liste des signaux en entrée, *-- sur ces signaux*
- la liste des signaux en sorties, *-- pour l'ajout des transactions*
- les délais de calcul. *-- à la bonne date*

Signal :

- la liste des processus émetteurs, *-- le plus souvent 1 seul*
- la liste des processus destination, *-- pour savoir ce qu'il faut réévaluer*
- Une valeur. *-- le plus souvent Booléenne*
- une fonction de résolution *-- lorsque plusieurs valeurs sont attribuées à la même date*

Transaction :

- date *-- l'instant où la transaction doit se faire*
- signal *-- un bit ou un vecteur de bits*
- valeur *-- la valeur qui sera affectée au signal*

Patterns

- La simulation du système consiste à appliquer des valeurs sur les signaux d'entrées à des dates précises et observer son comportement.
  - Une valeur appliquée sur un signal d'entrée à une date précise définit une transaction.
  - L'ensemble de ces transactions peut être placé dans l'échéancier avant le début de la simulation.
  - Ou ajouter au fur et à mesure des besoins

Principe du simulateur

L'algorithme de simulation se déroule en deux phases.

La phase Update

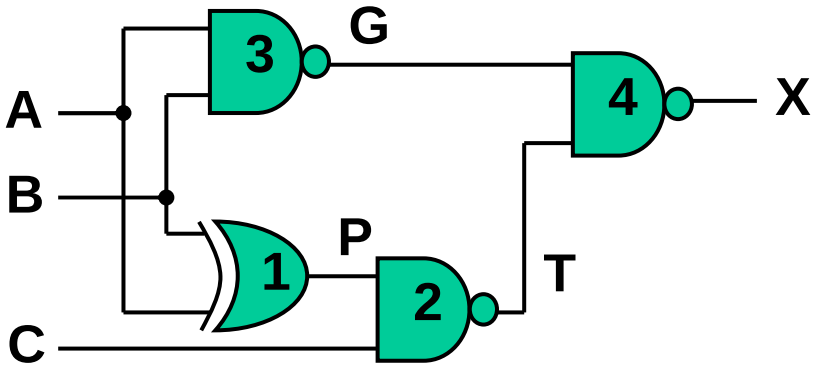
- ➔ Avancer le temps courant jusqu'à la prochaine transaction.
- ➔ Extraire la liste des transactions à la date courante, mettre à jour les signaux et déterminer la liste des événements.
- ➔ Déterminer la liste de tous les processus ayant eu un événement sur une entrée de sa liste de sensibilité.

La phase Execute

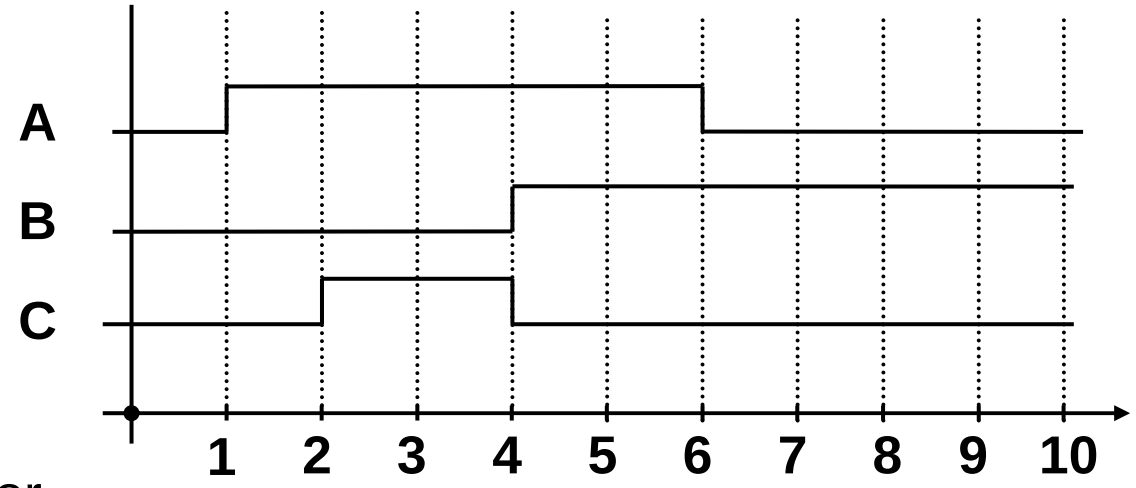
- ➔ Exécuter tous les processus ayant vu un événement.
- ➔ Introduire les nouvelles transactions dans l'échéancier.

Exemple simple

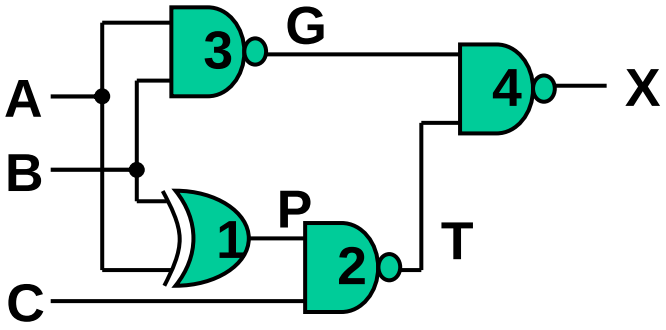
- Hypothèses simplificatrices:
  - tous les processus n'ont qu'une seule sortie
  - le délai de calcul d'un processus ne dépend pas des entrées
  - chaque signal n'a qu'une seule source
  - tous les signaux d'entrées sont initialisés à 0



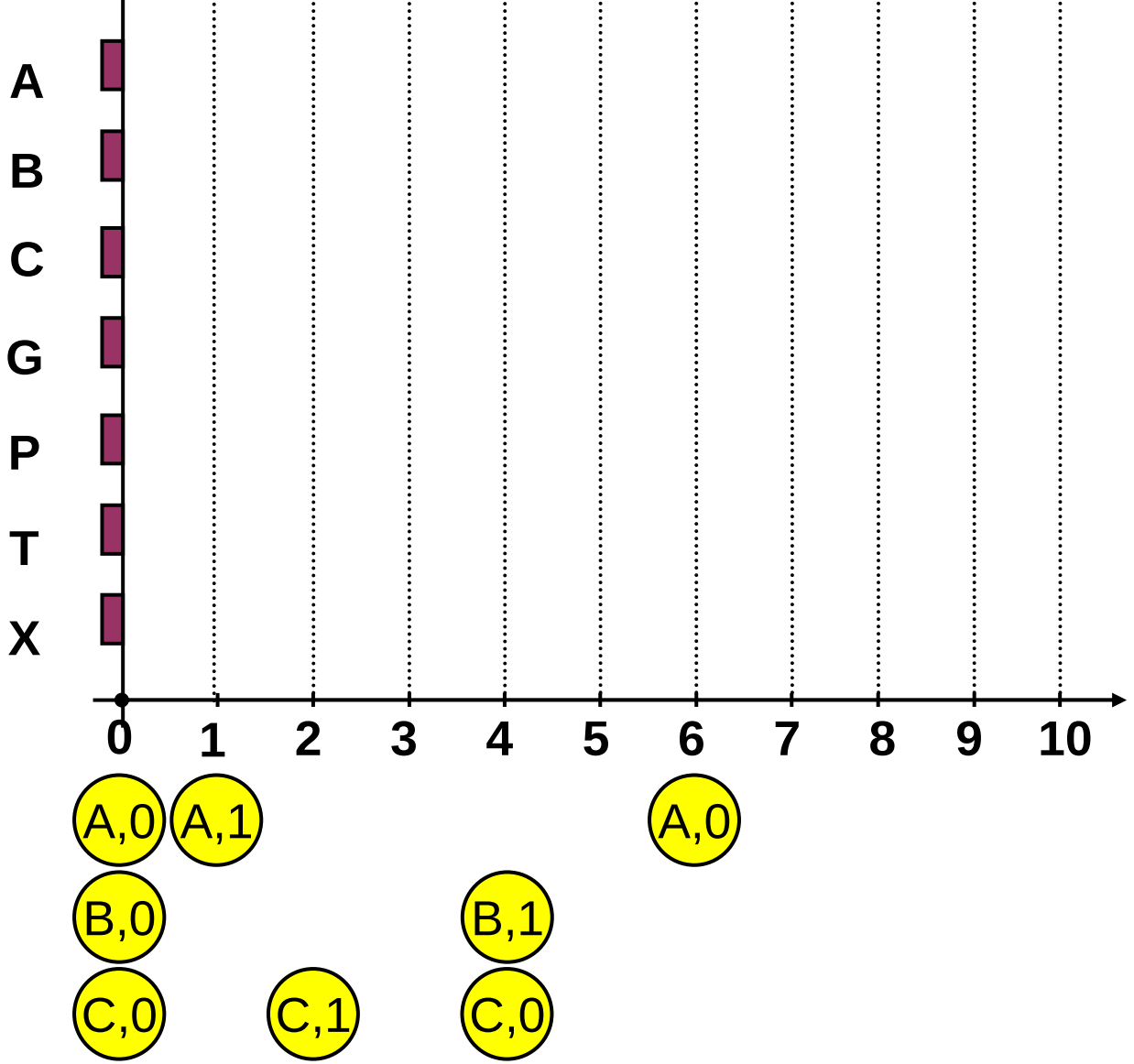
délais : 1ns pour le nand
2ns pour le xor



Simulation

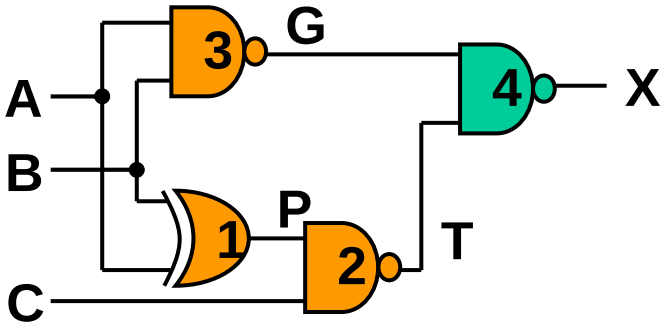


délais : 1ns pour le nand
2ns pour le xor



date courante

Simulation

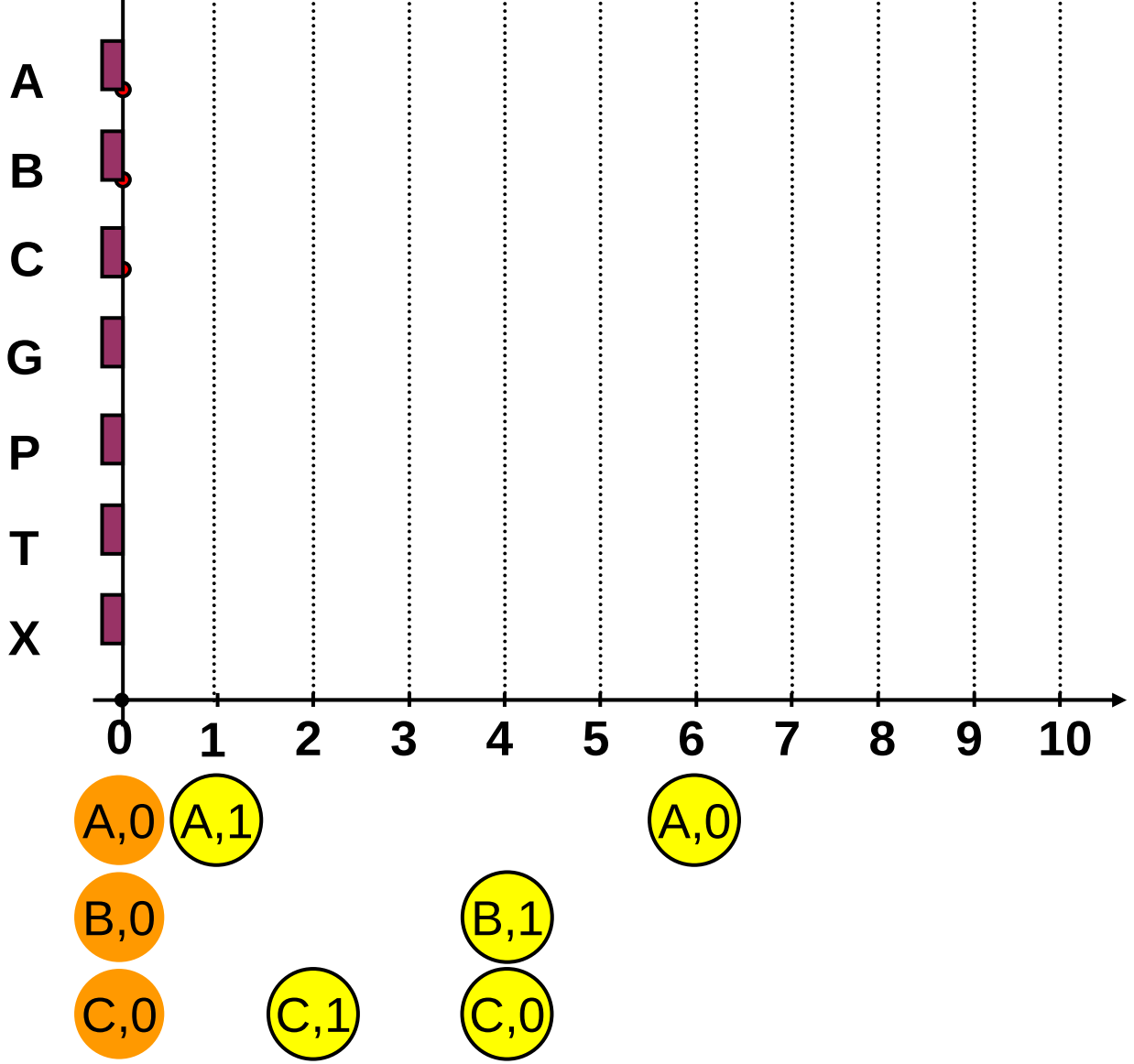


délais : 1ns pour le nand
2ns pour le xor

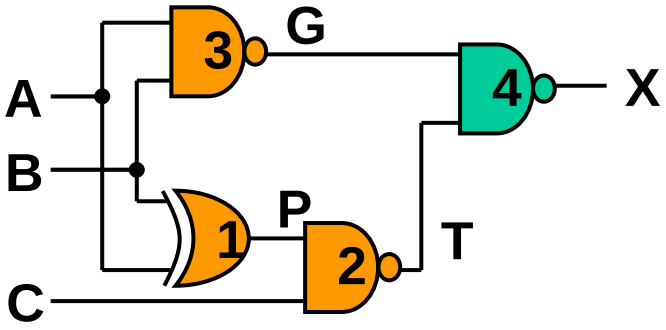
UPDATE

date courante

0



Simulation

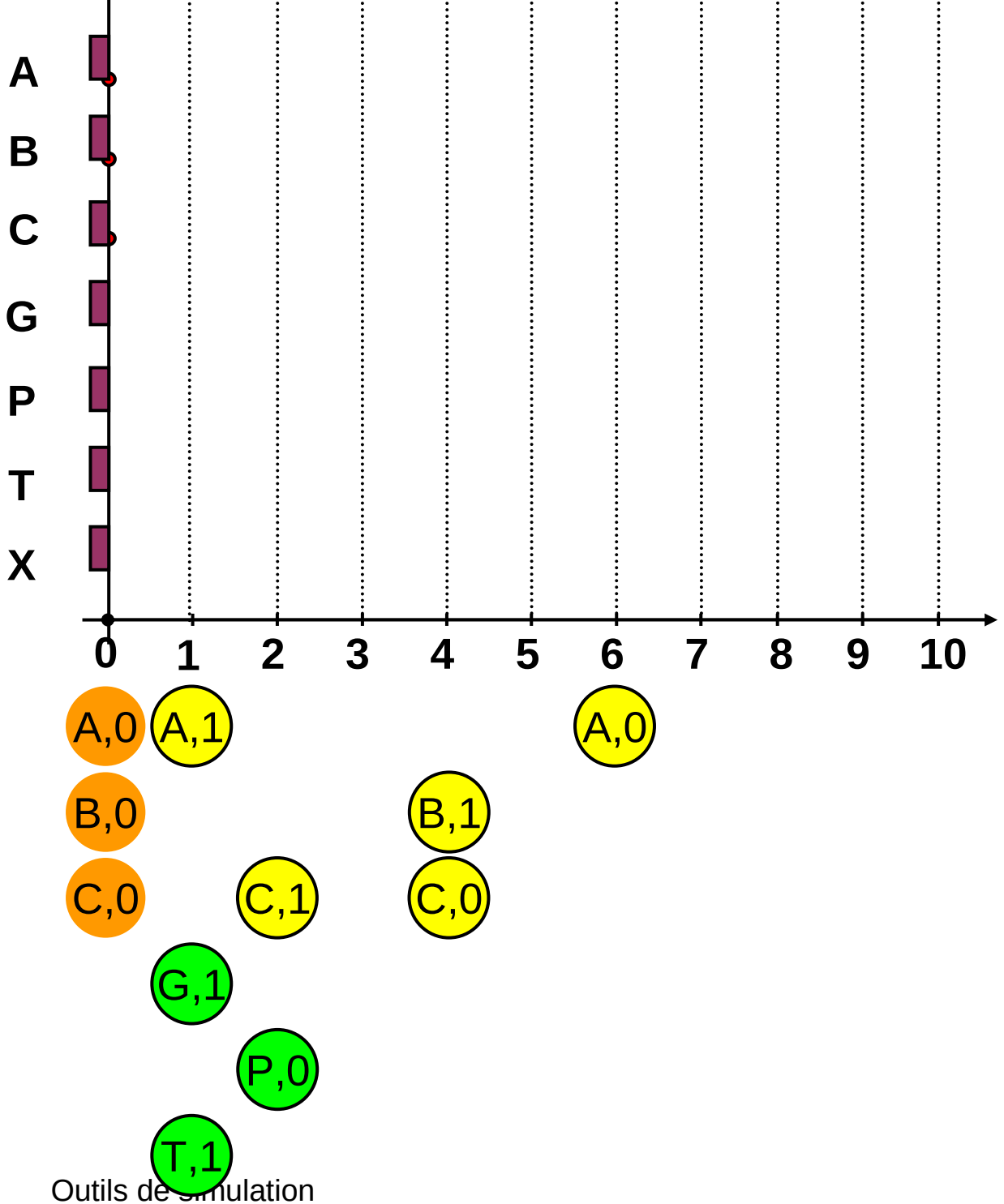


délais : 1ns pour le nand
2ns pour le xor

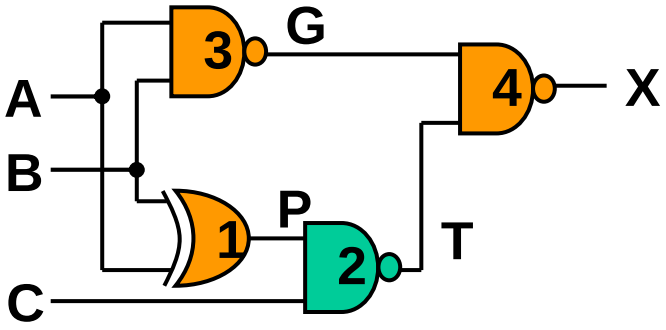
EXECUTE

date courante

0



Simulation

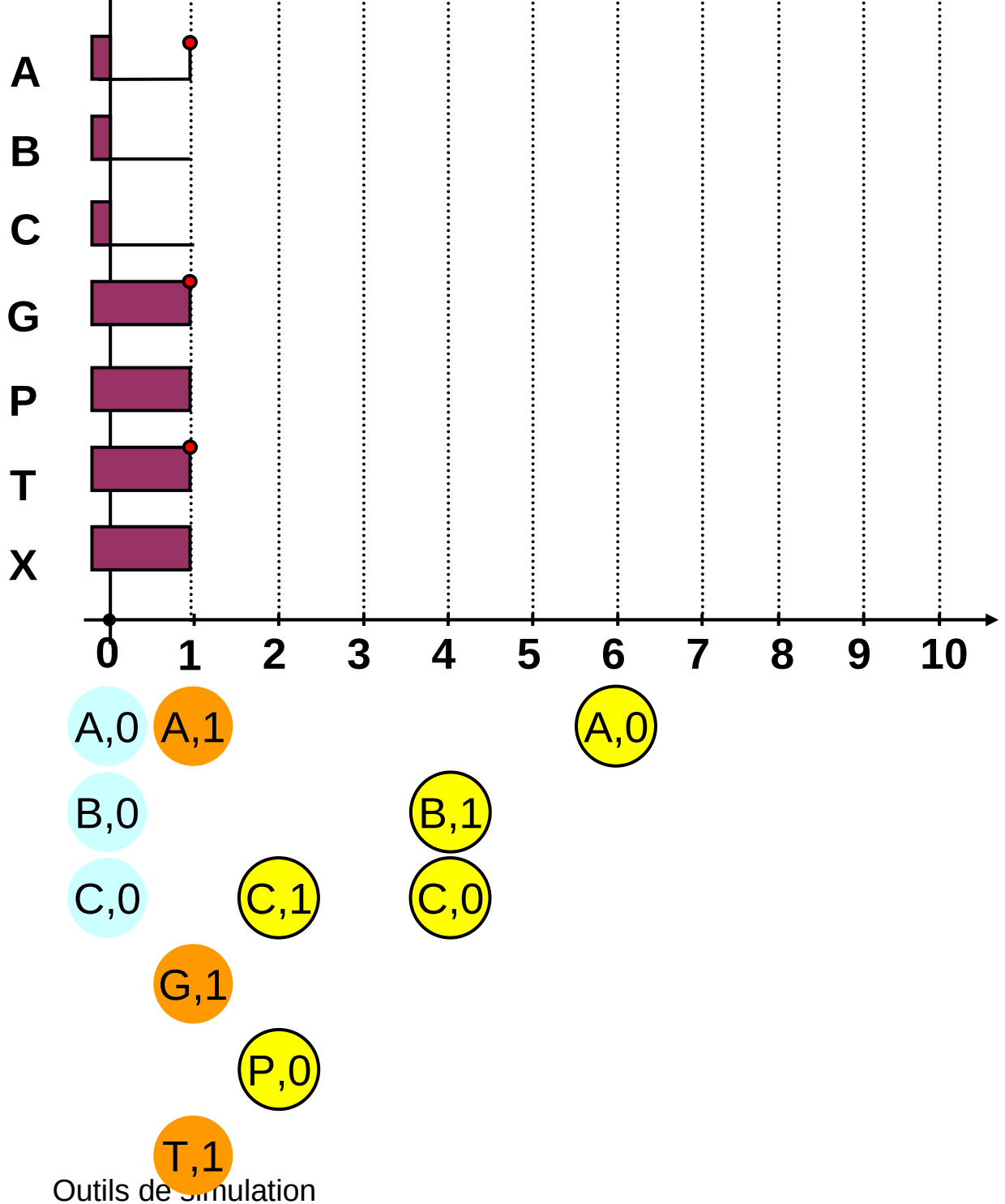


délais : 1ns pour le nand
2ns pour le xor

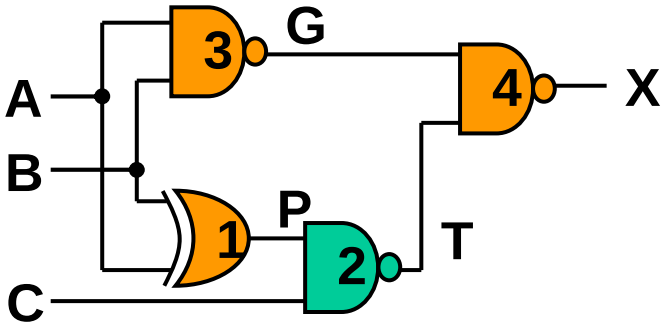
UPDATE

date courante

1



Simulation

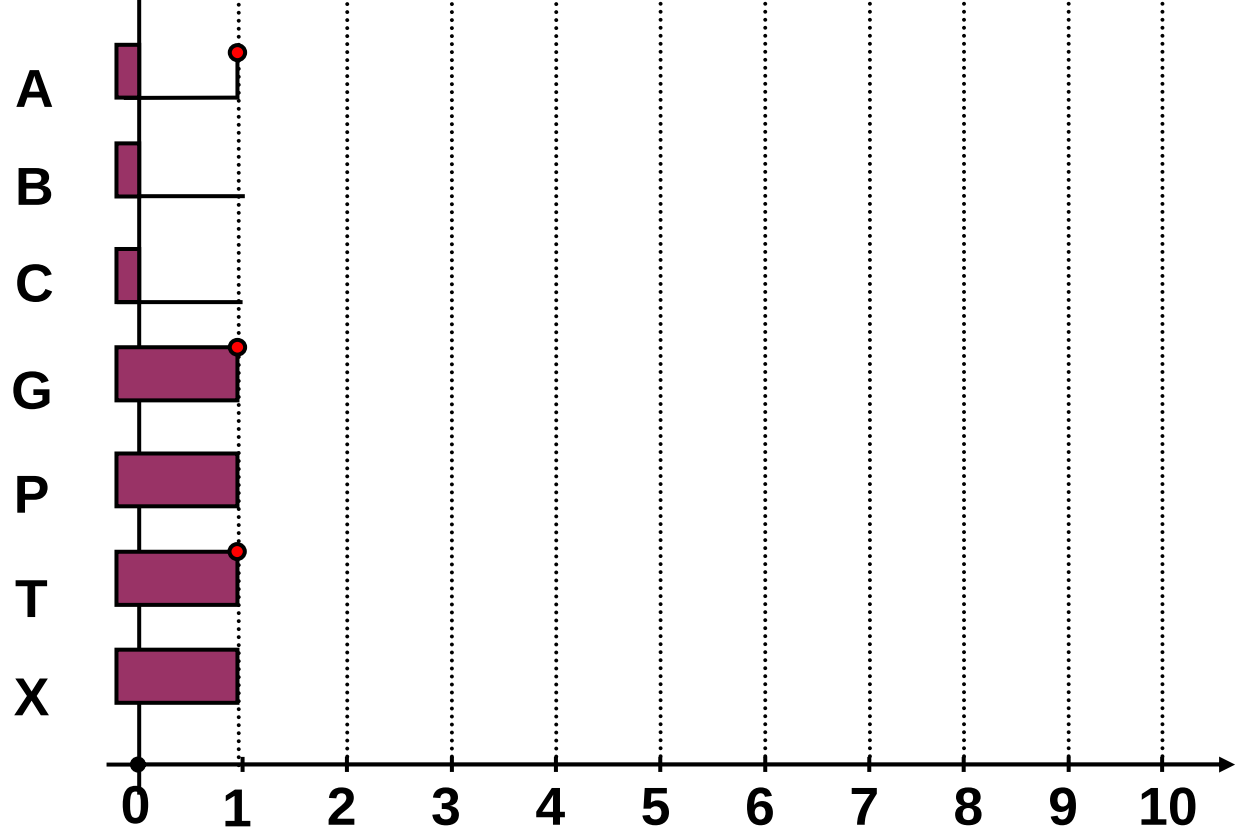


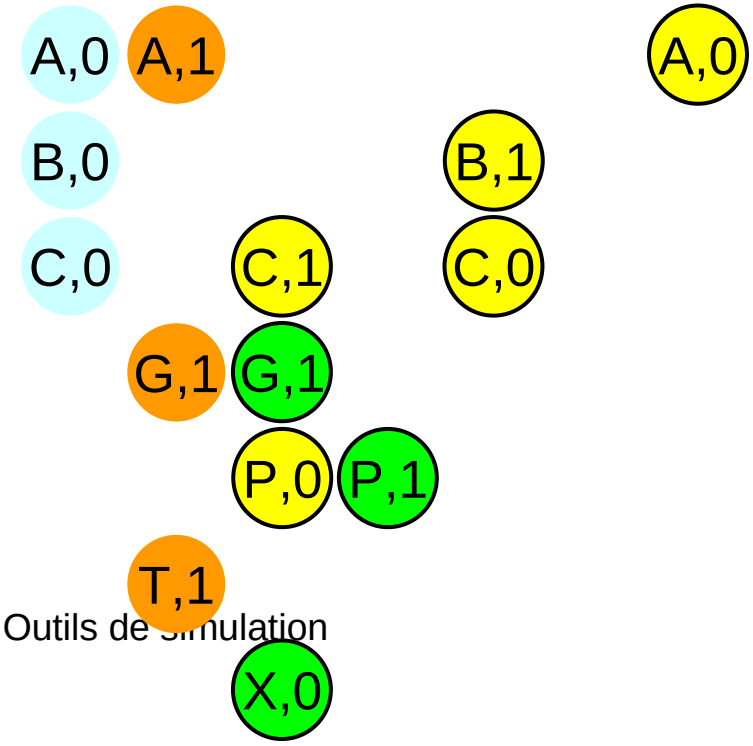
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

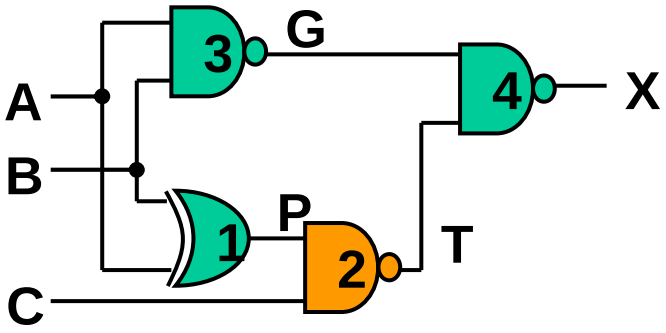
date courante

1





Simulation

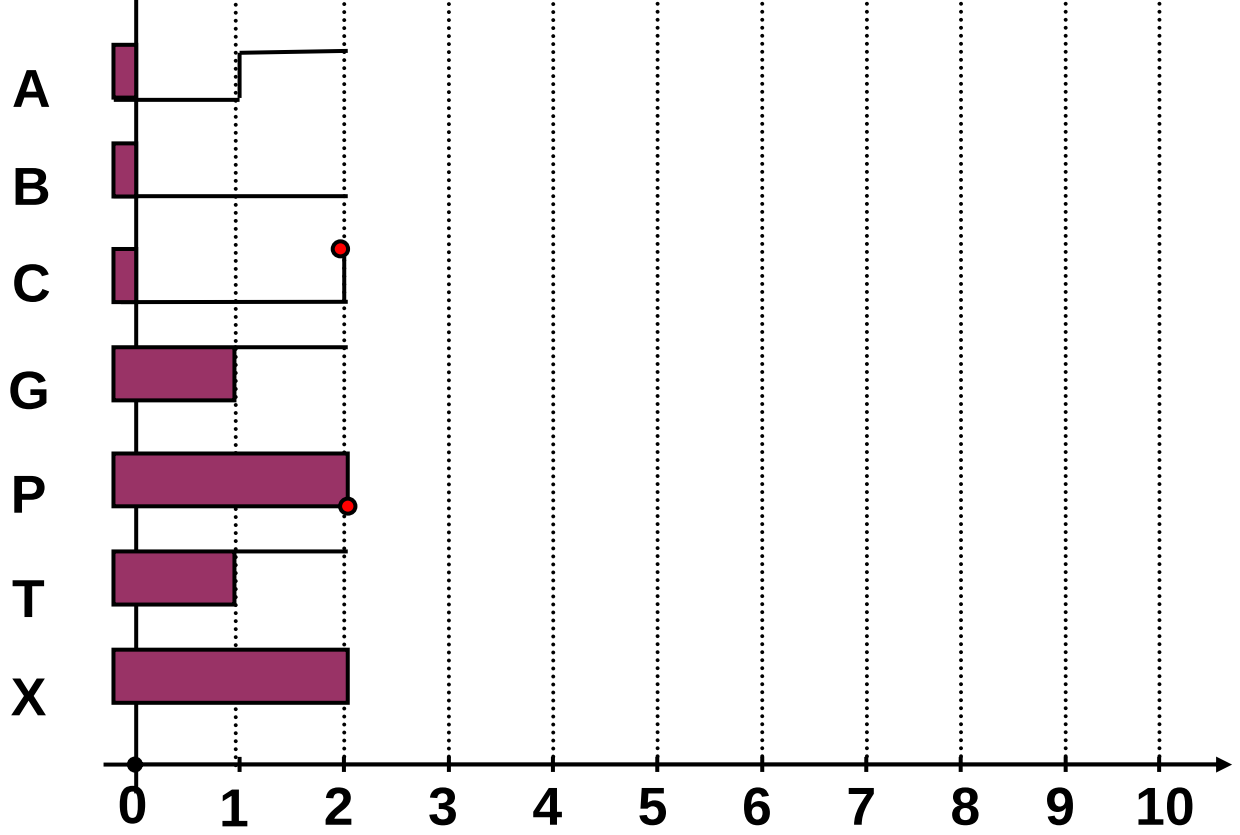


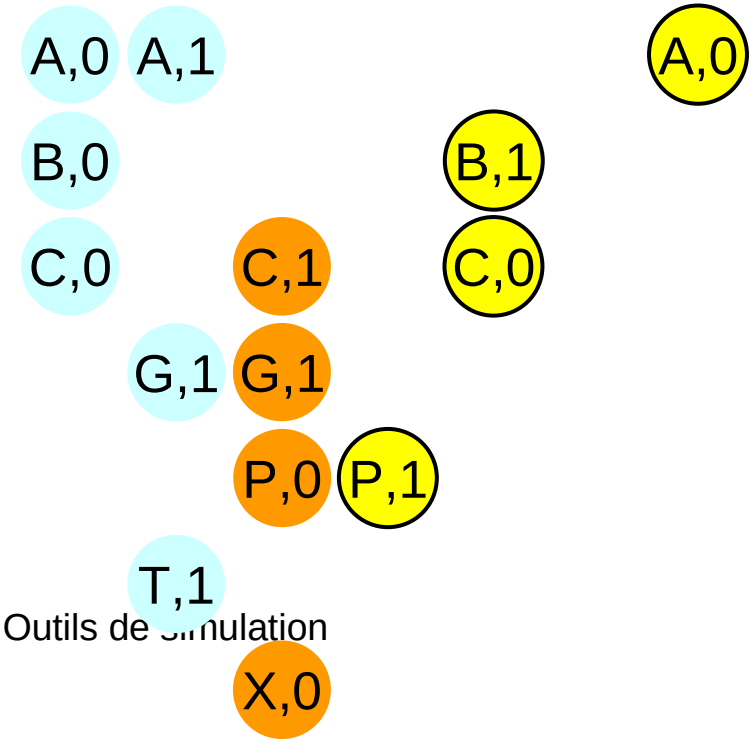
délais : 1ns pour le nand
2ns pour le xor

UPDATE

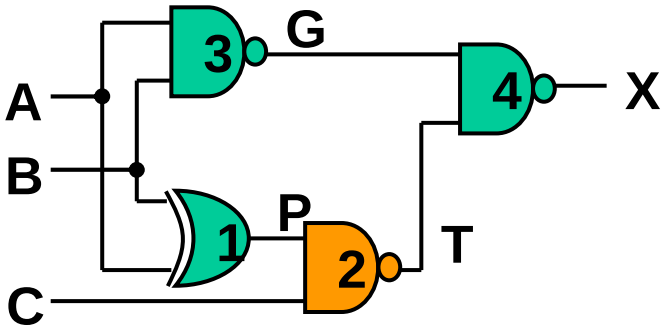
date courante

2





Simulation

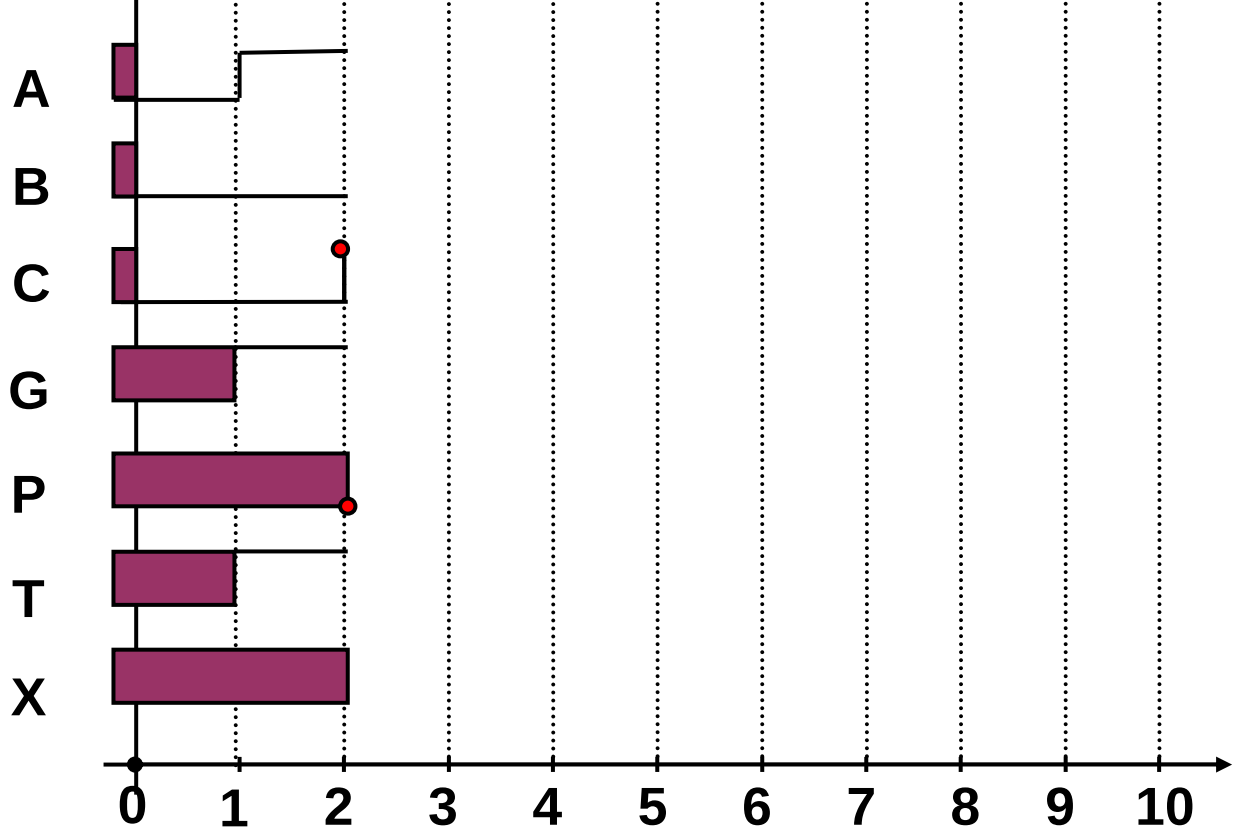


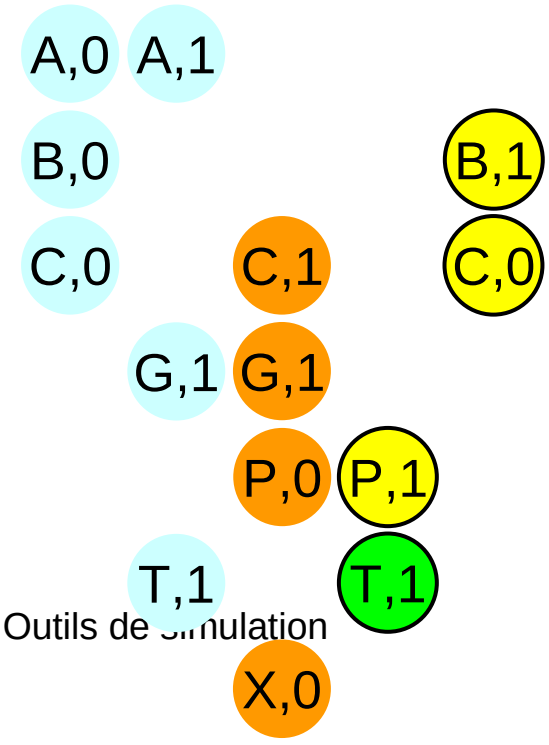
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

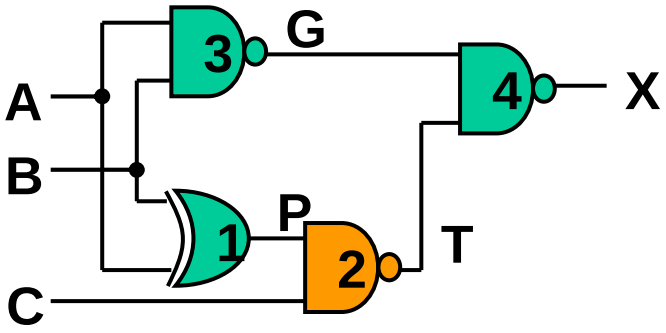
date courante

2





Simulation

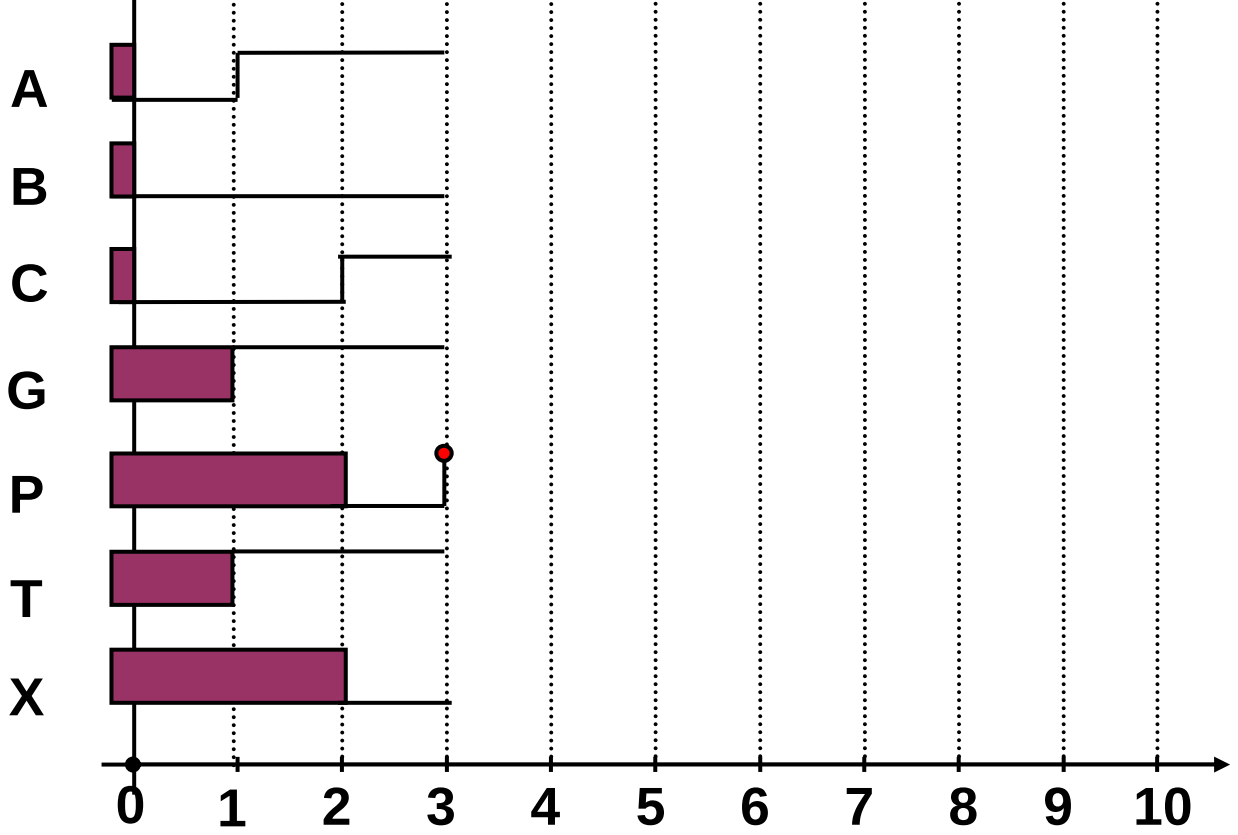


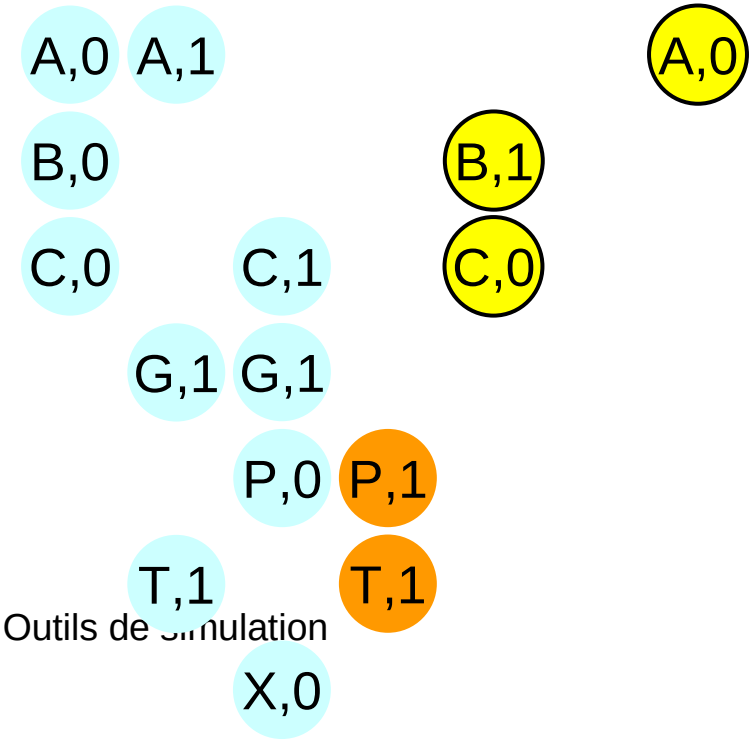
délais : 1ns pour le nand
2ns pour le xor

UPDATE

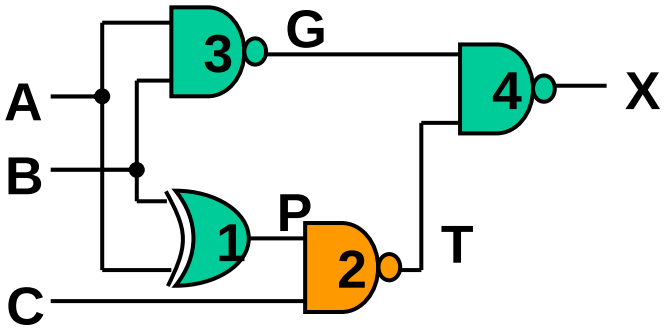
date courante

3





Simulation

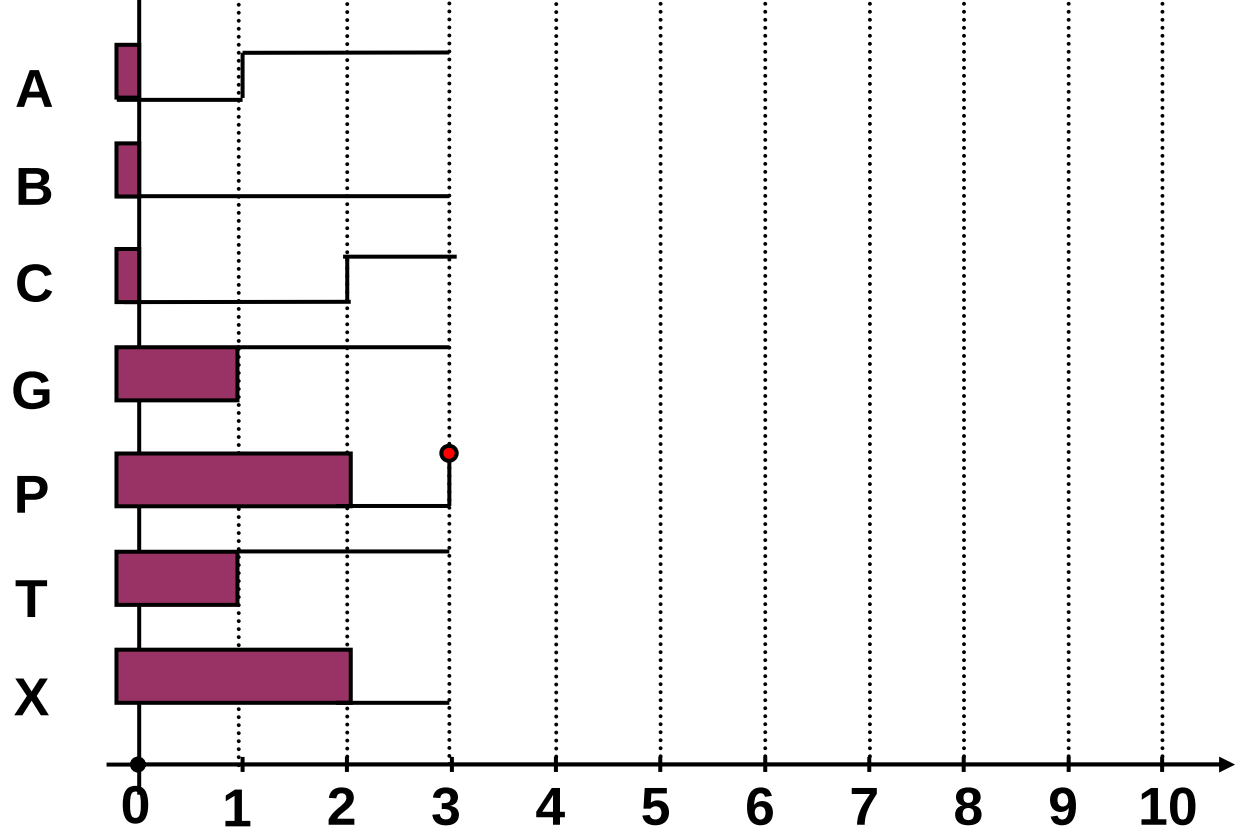


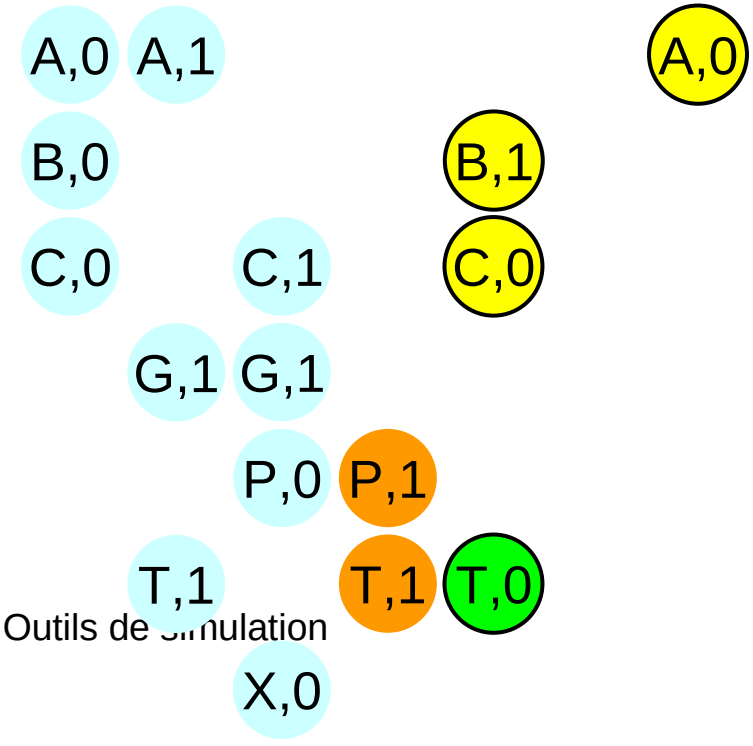
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

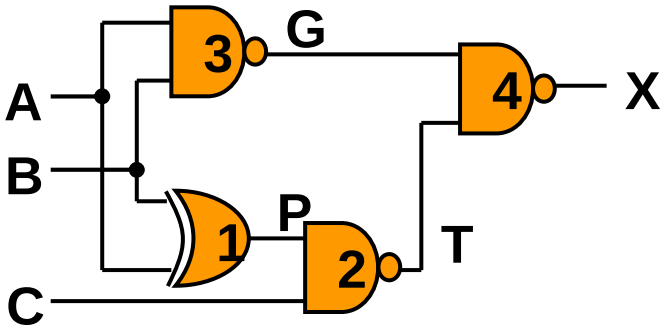
date courante

3





Simulation

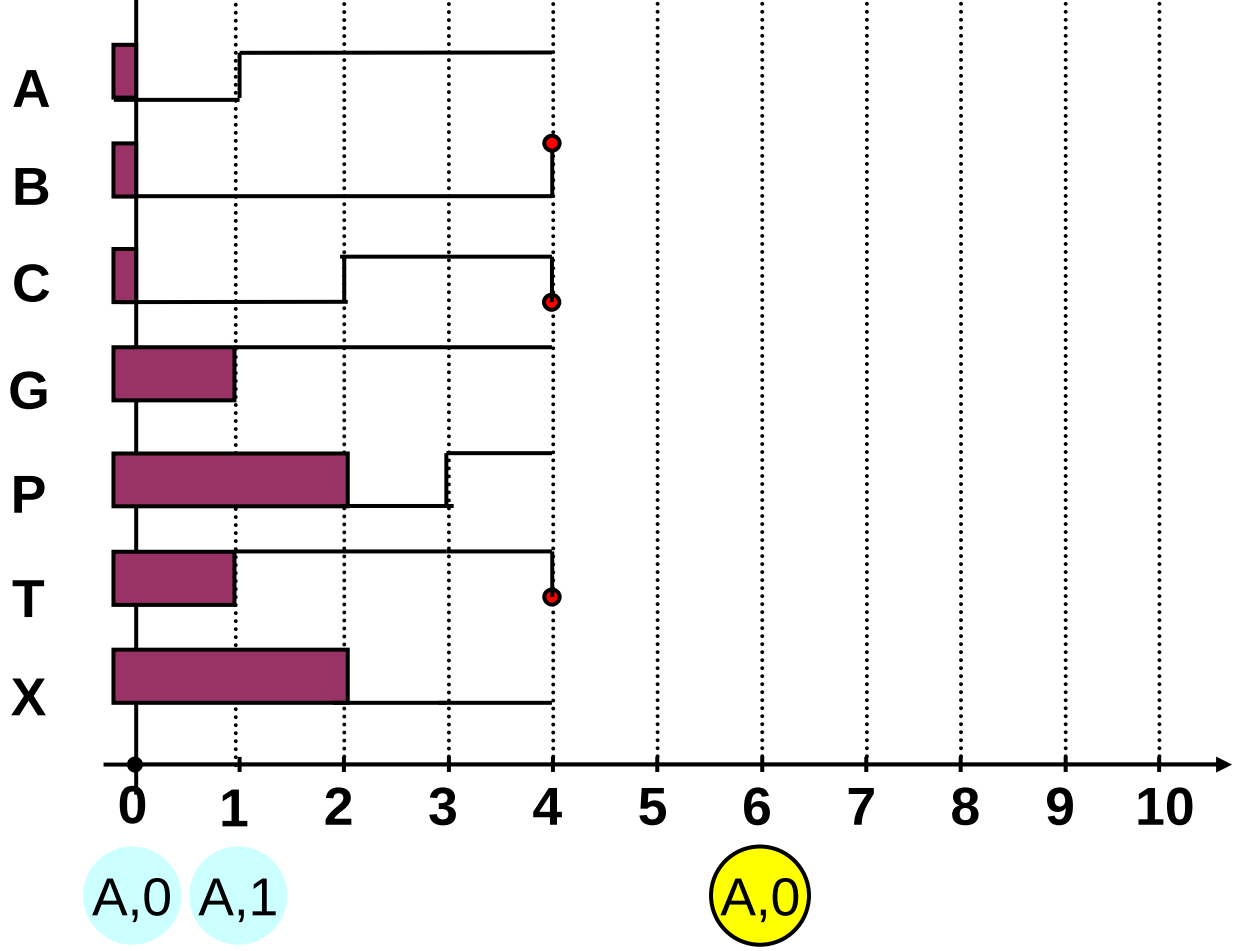


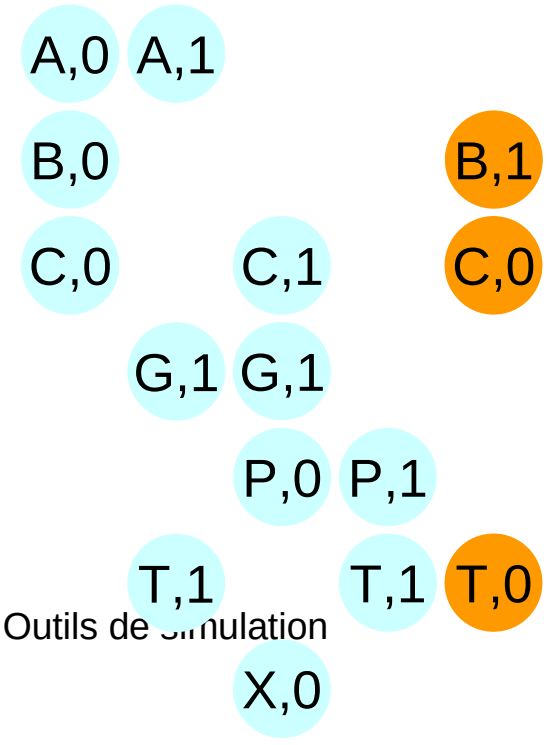
délais : 1ns pour le nand
2ns pour le xor

UPDATE

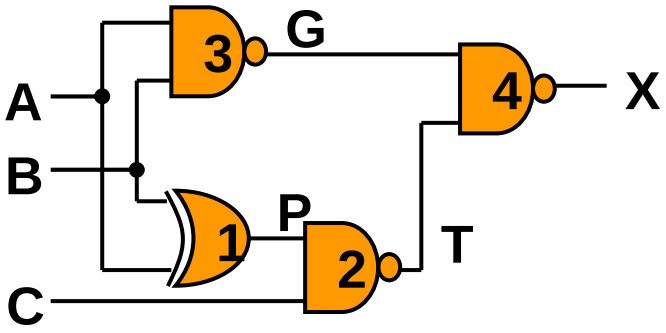
date courante

4





Simulation

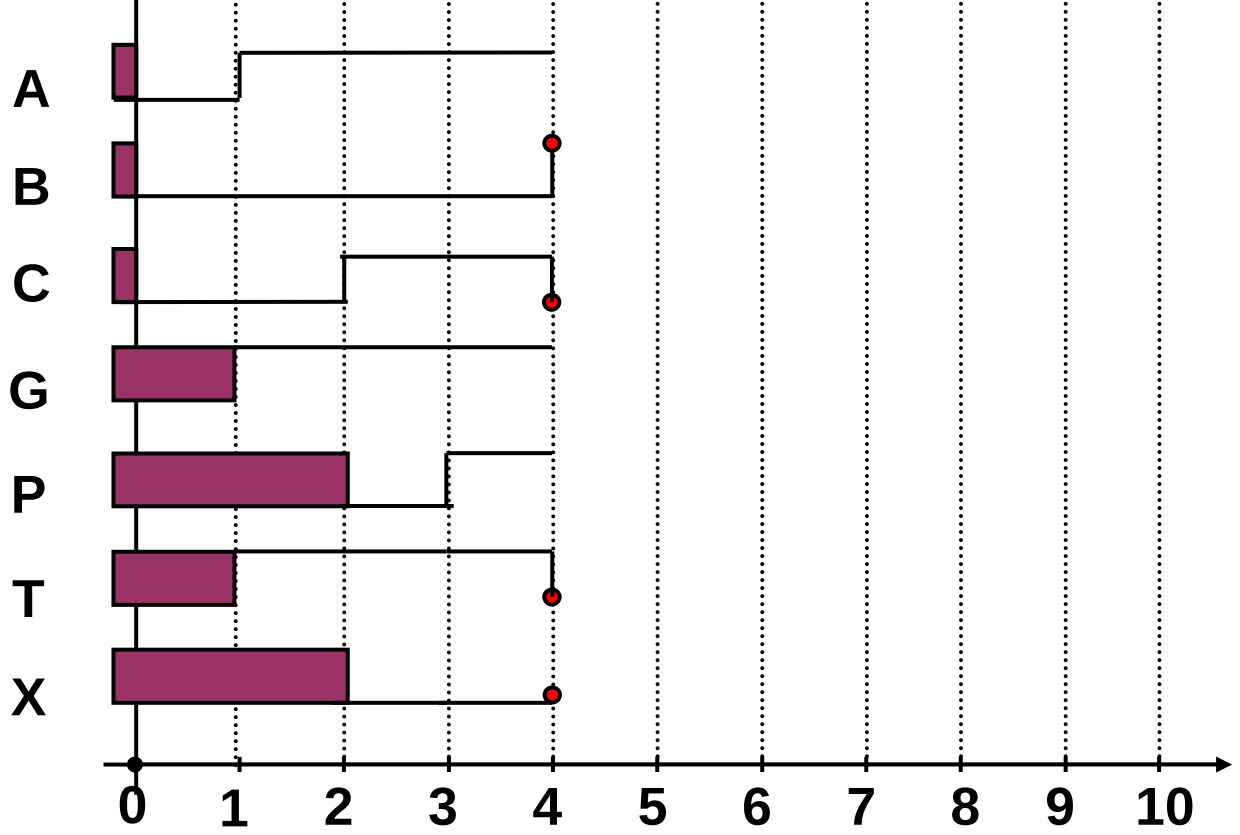


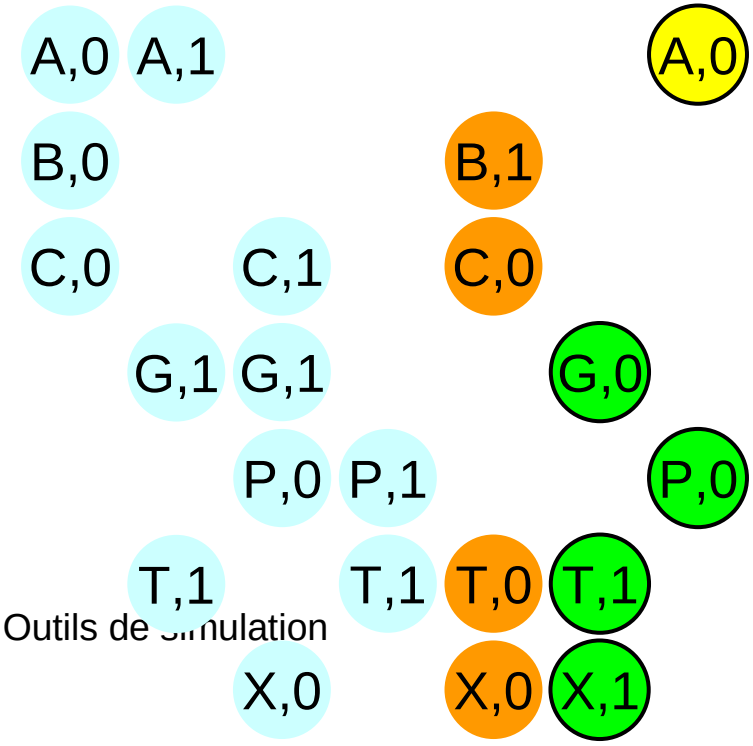
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

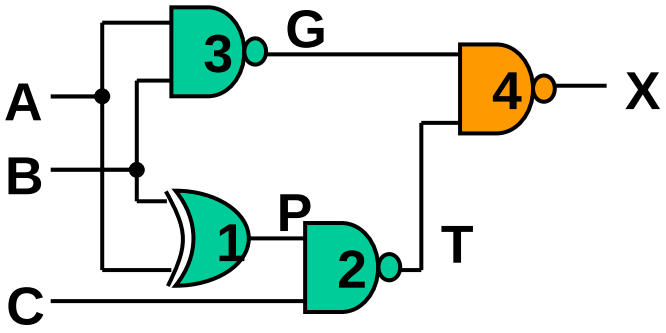
date courante

4





Simulation

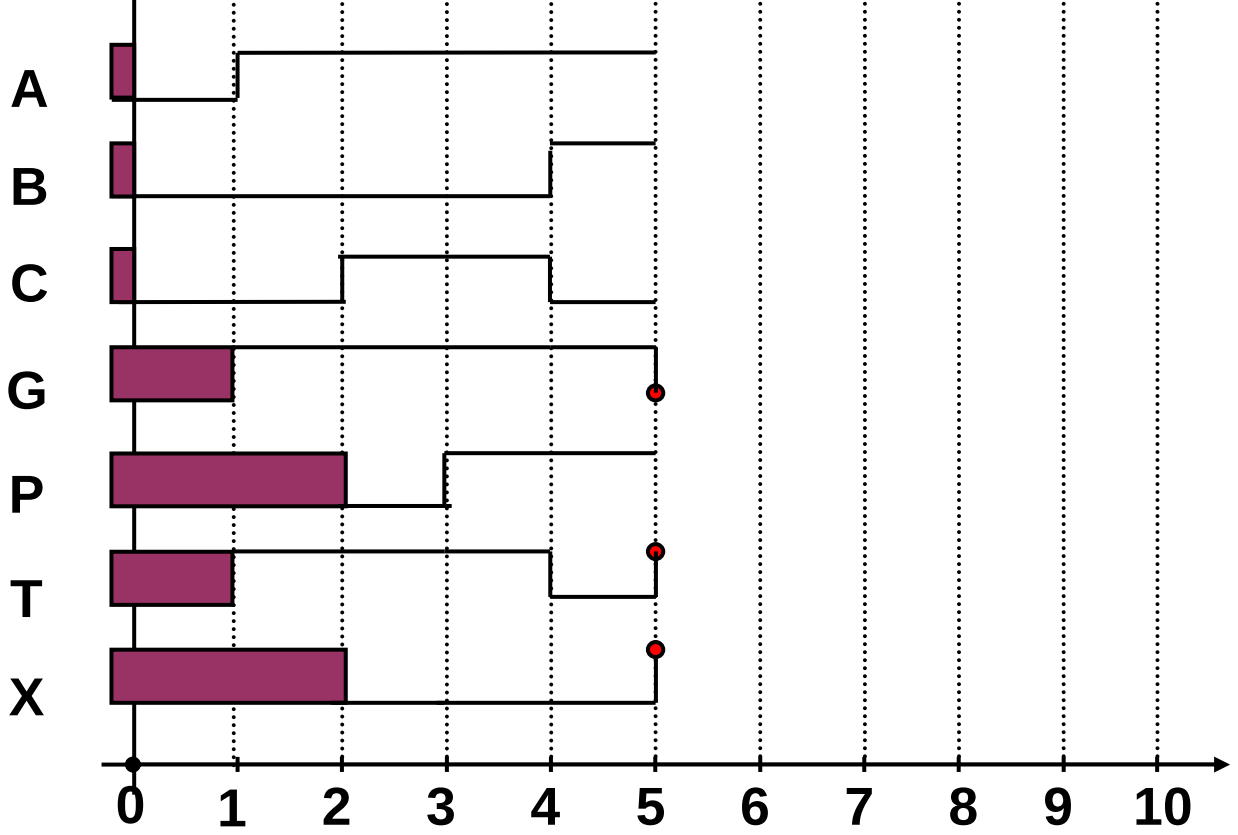


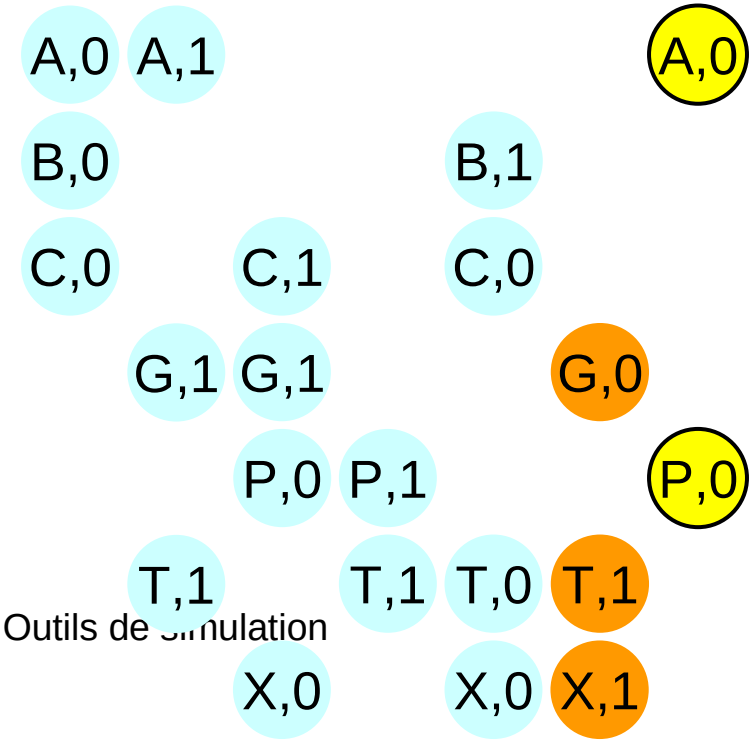
délais : 1ns pour le nand
2ns pour le xor

UPDATE

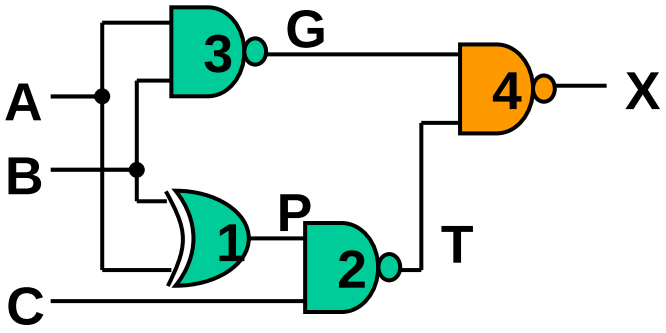
date courante

5





Simulation

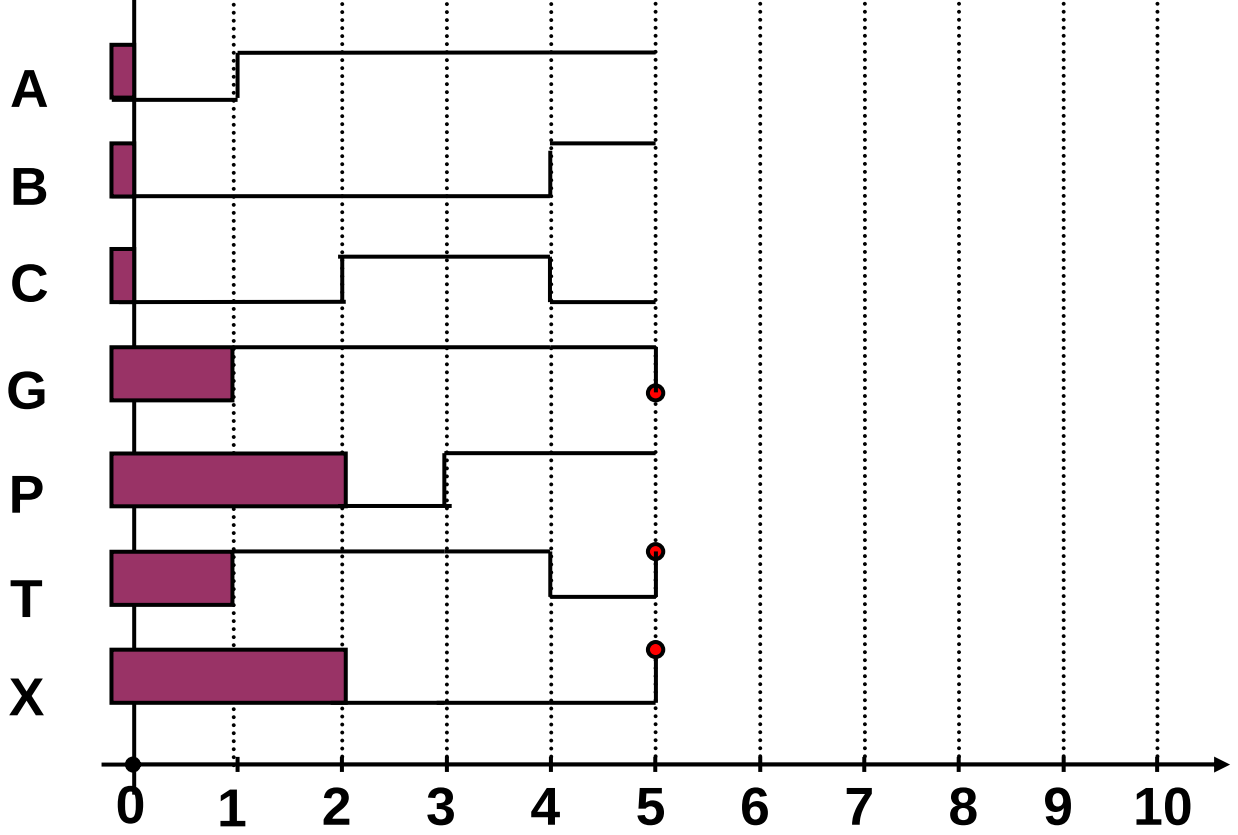


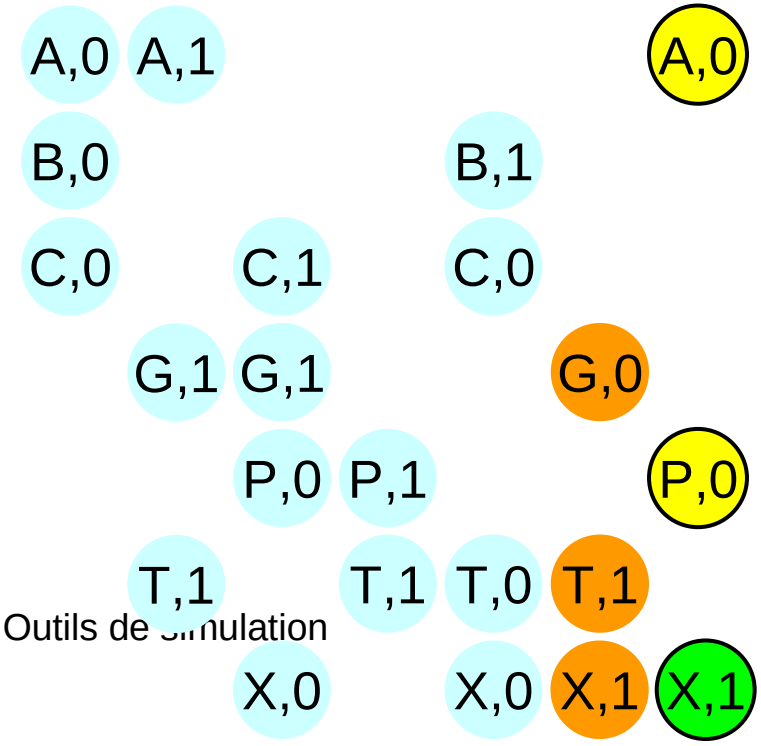
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

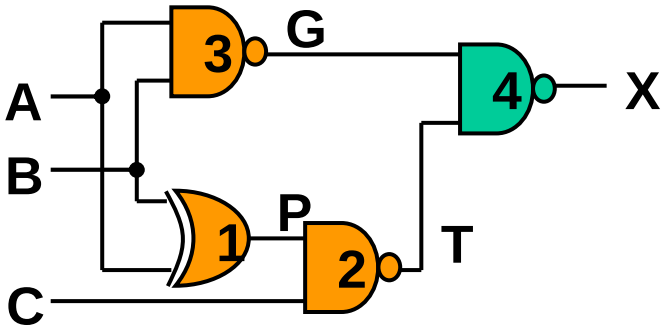
date courante

5





Simulation

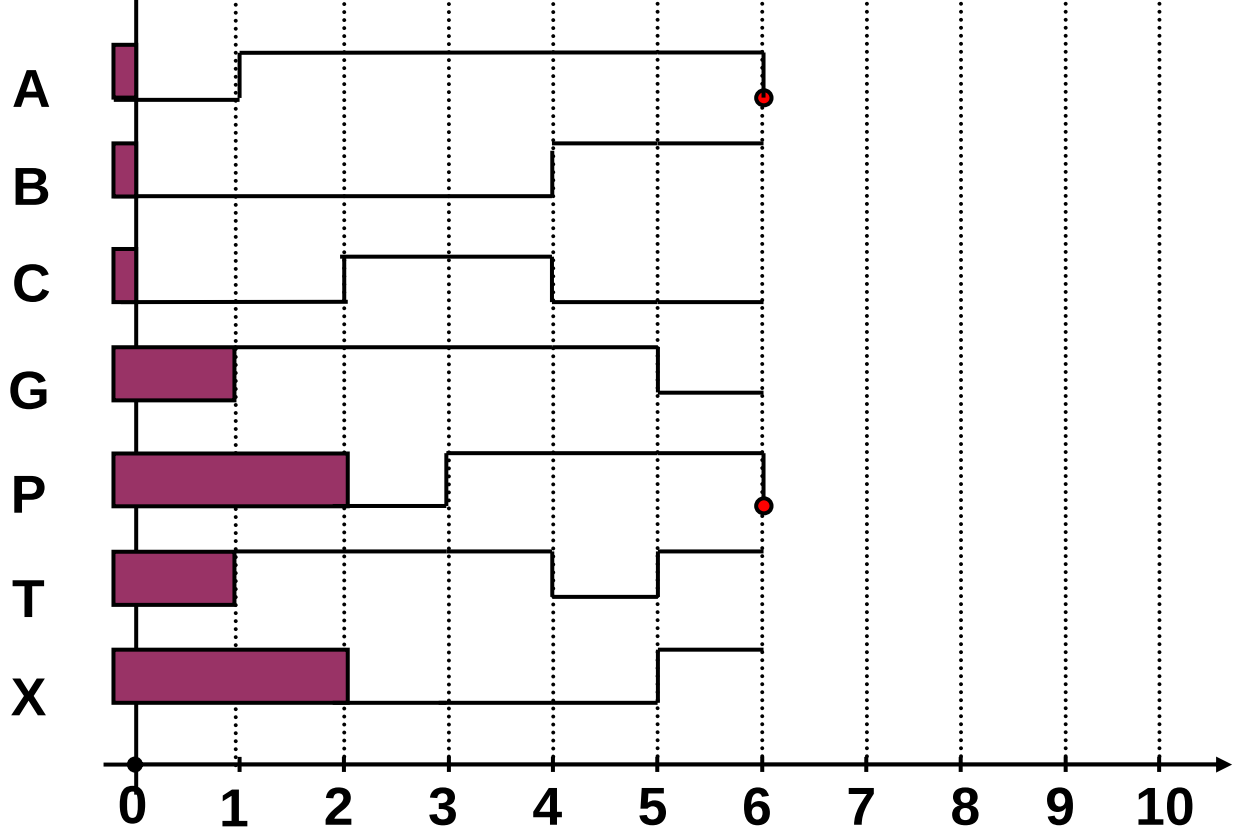


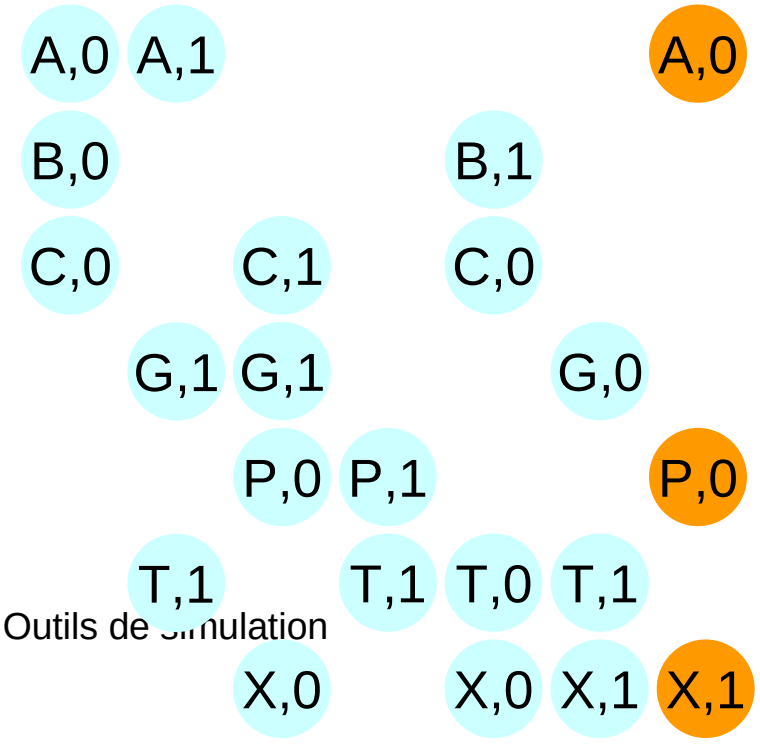
délais : 1ns pour le nand
2ns pour le xor

UPDATE

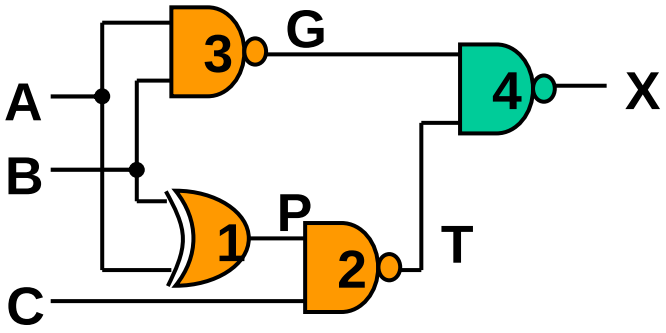
date courante

6





Simulation

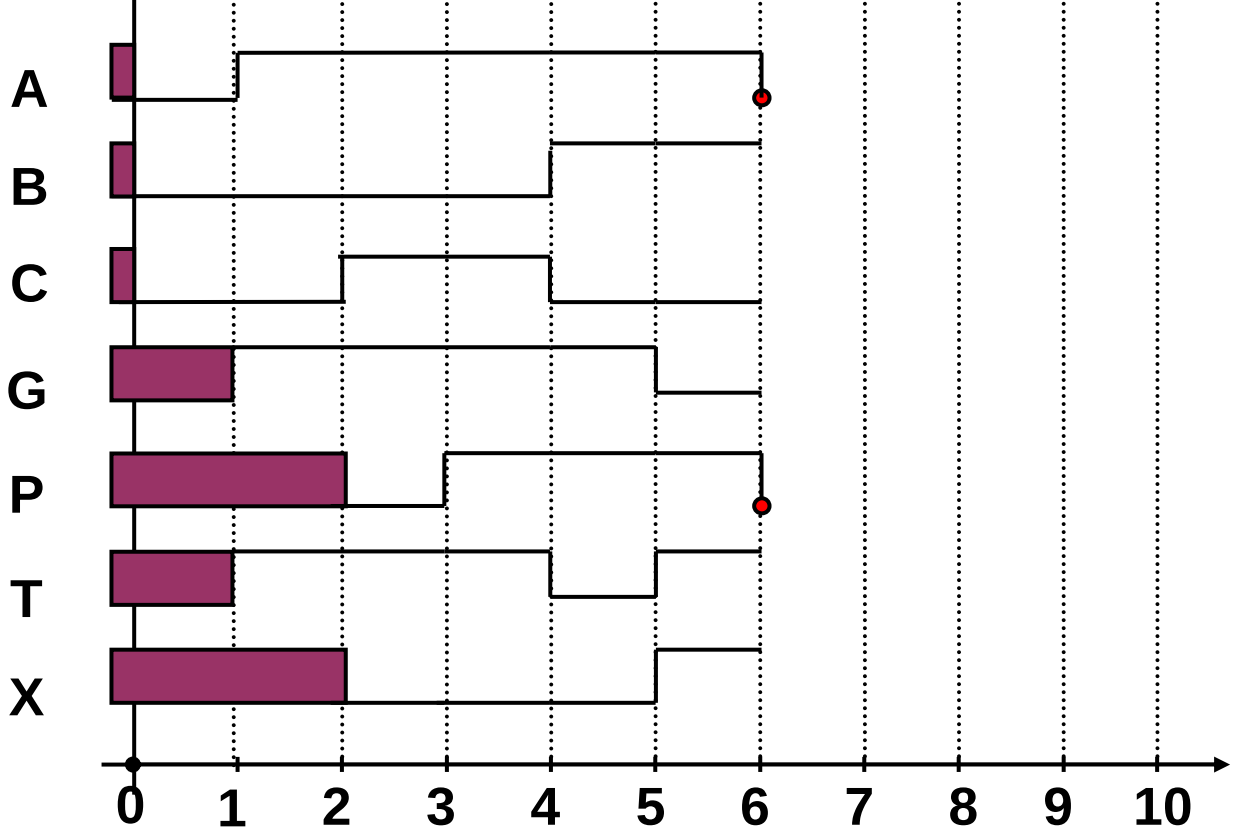


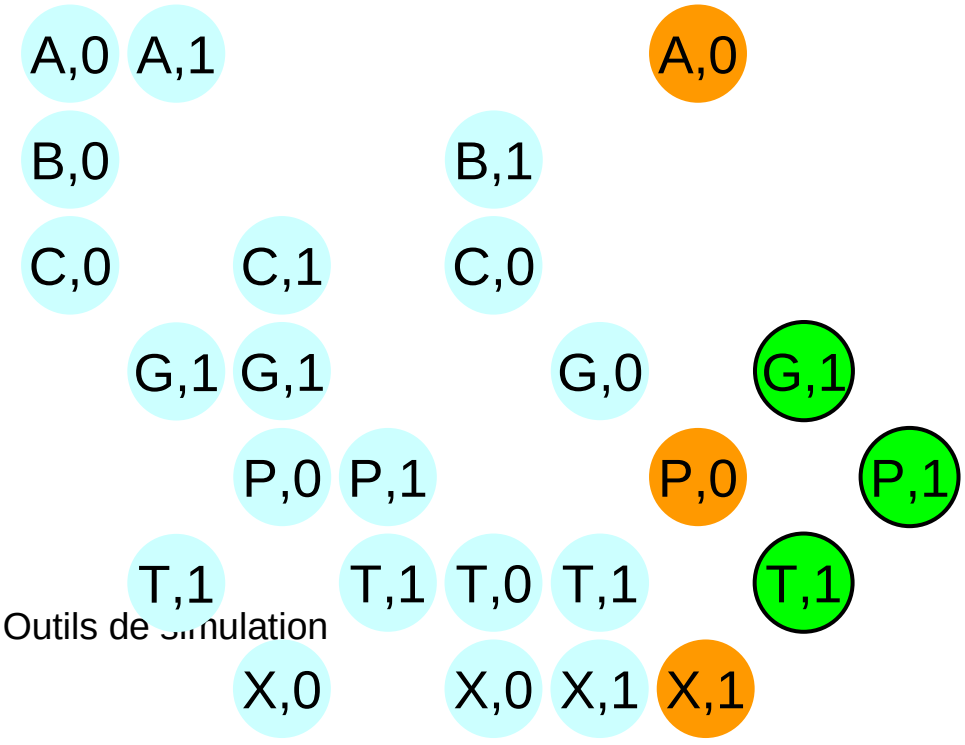
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

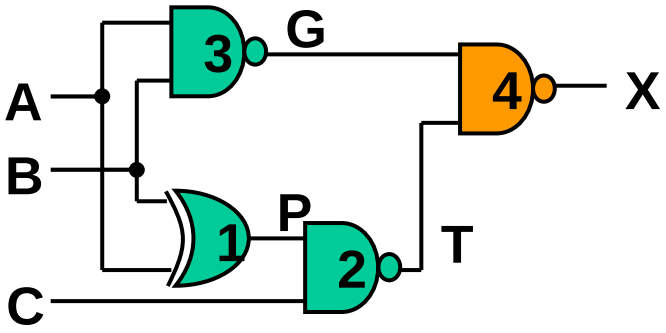
date courante

6





Simulation

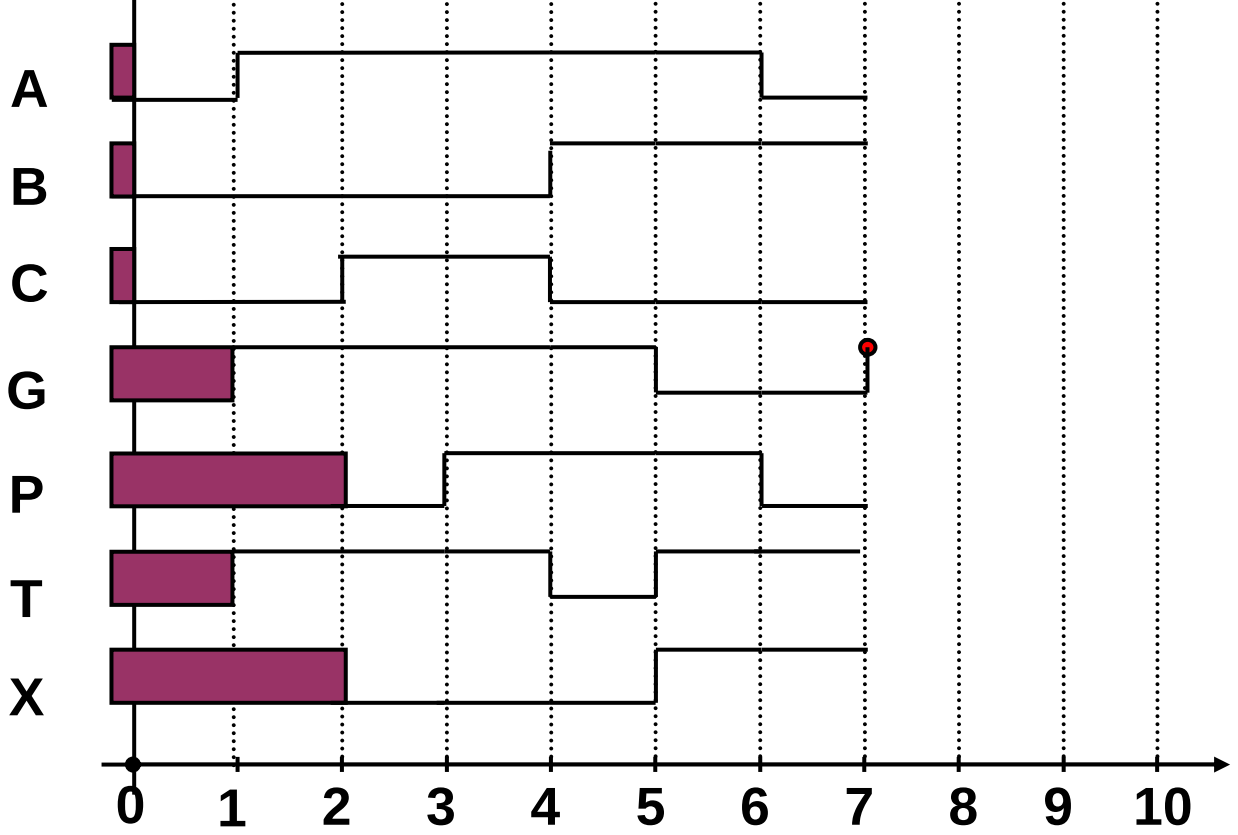


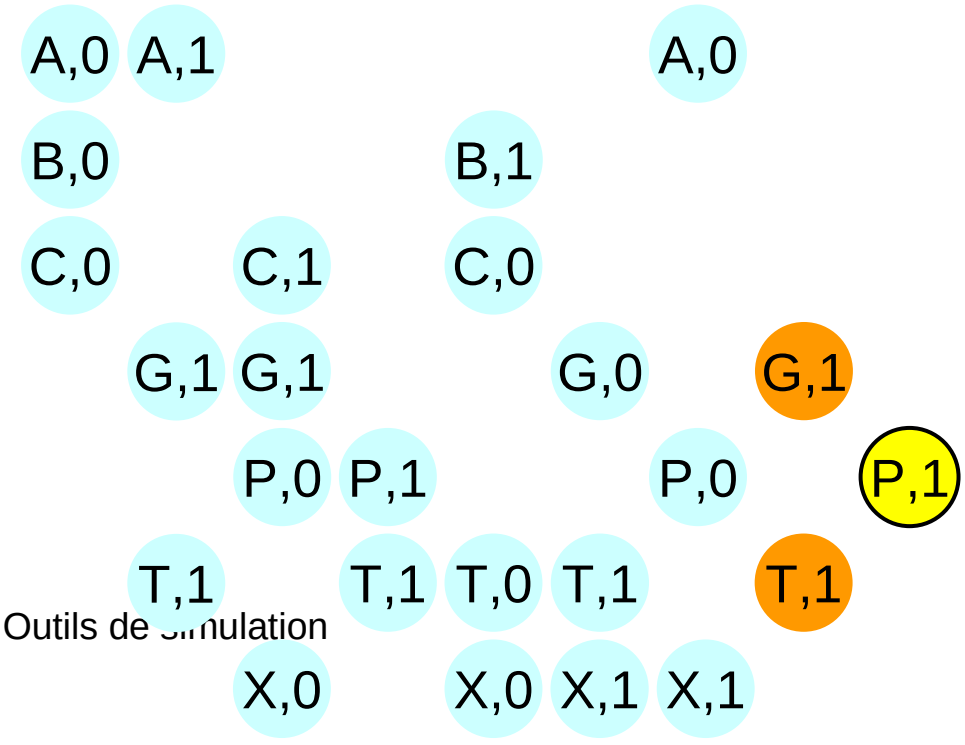
délais : 1ns pour le nand
2ns pour le xor

UPDATE

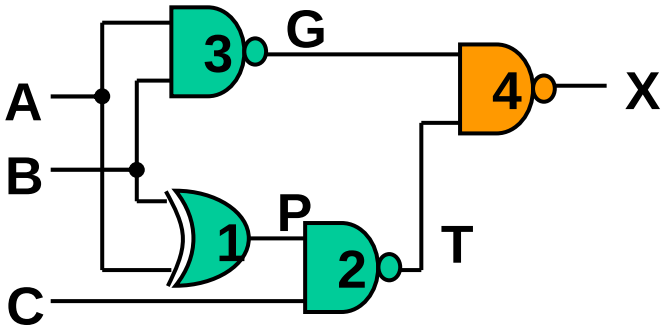
date courante

7





Simulation

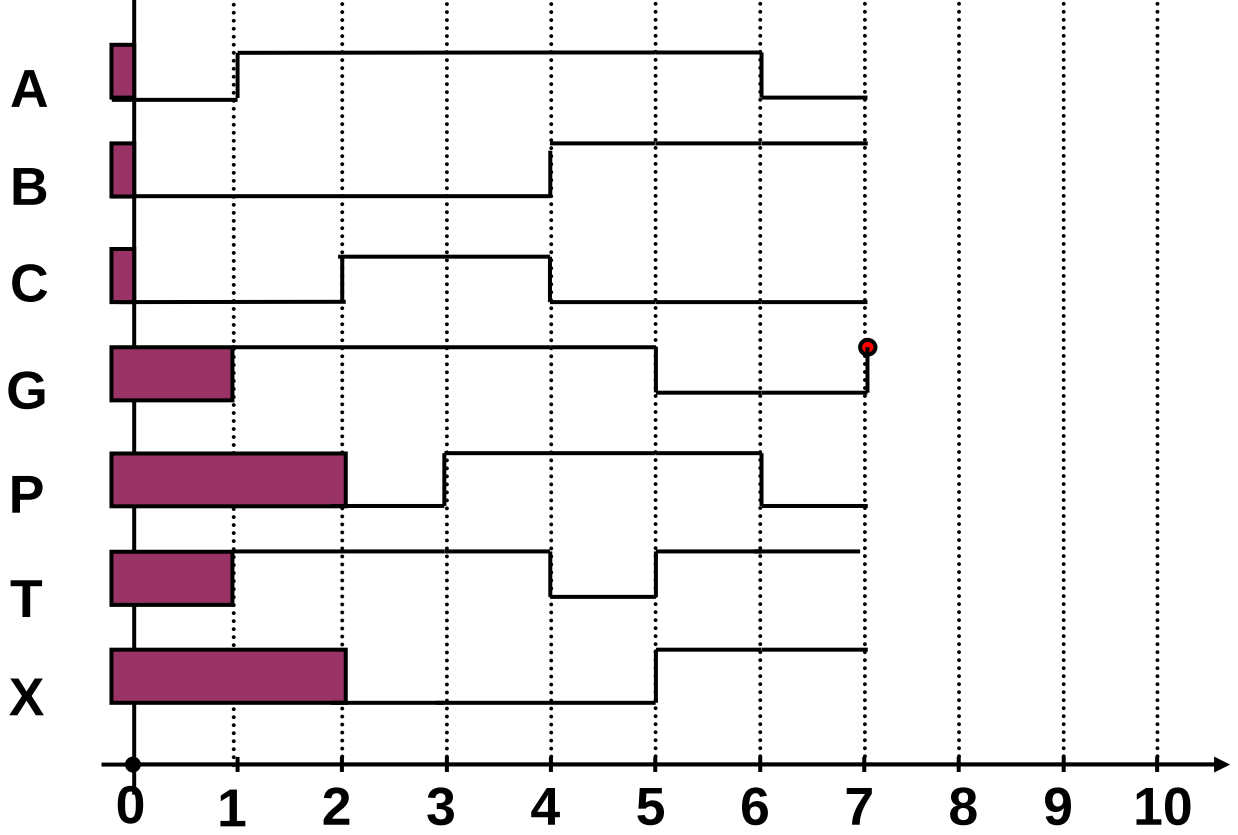


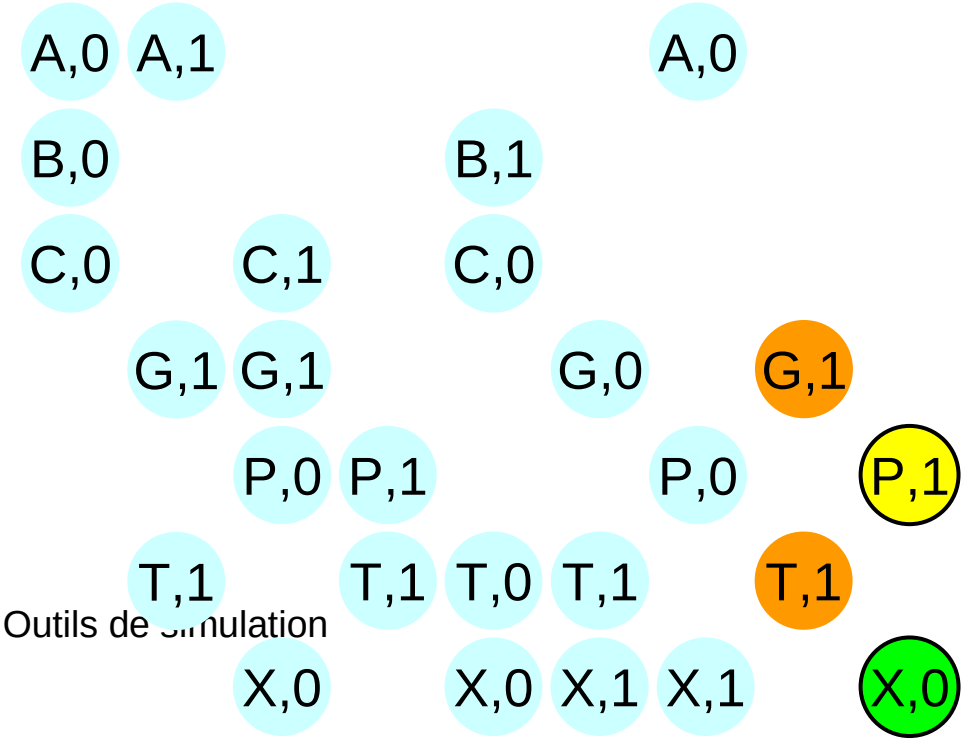
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

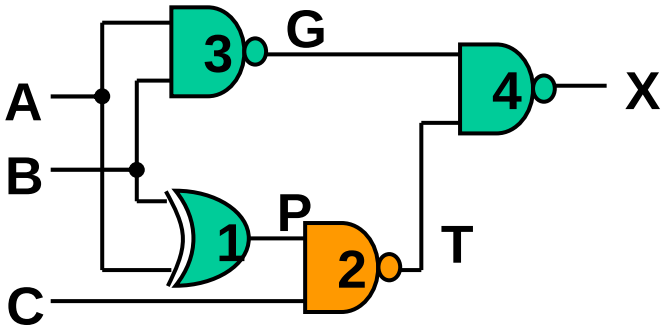
date courante

7





Simulation

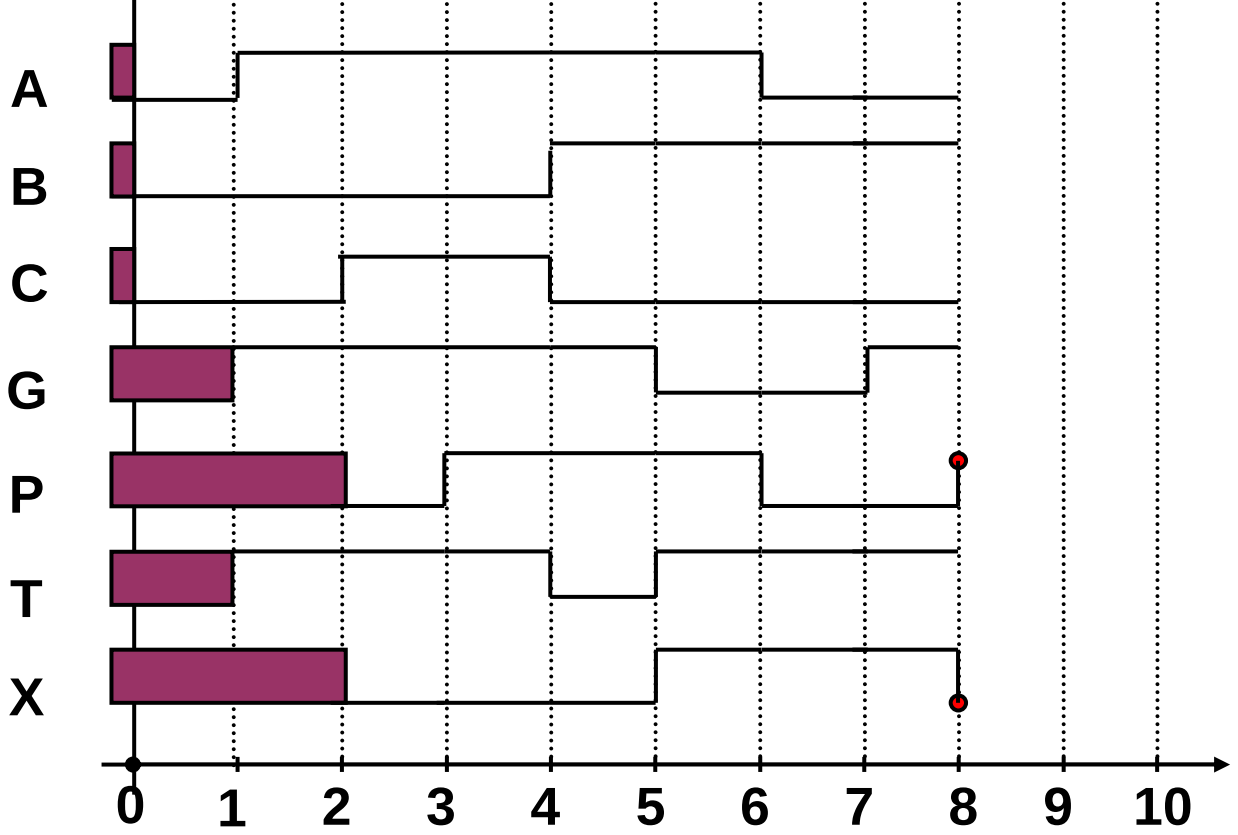


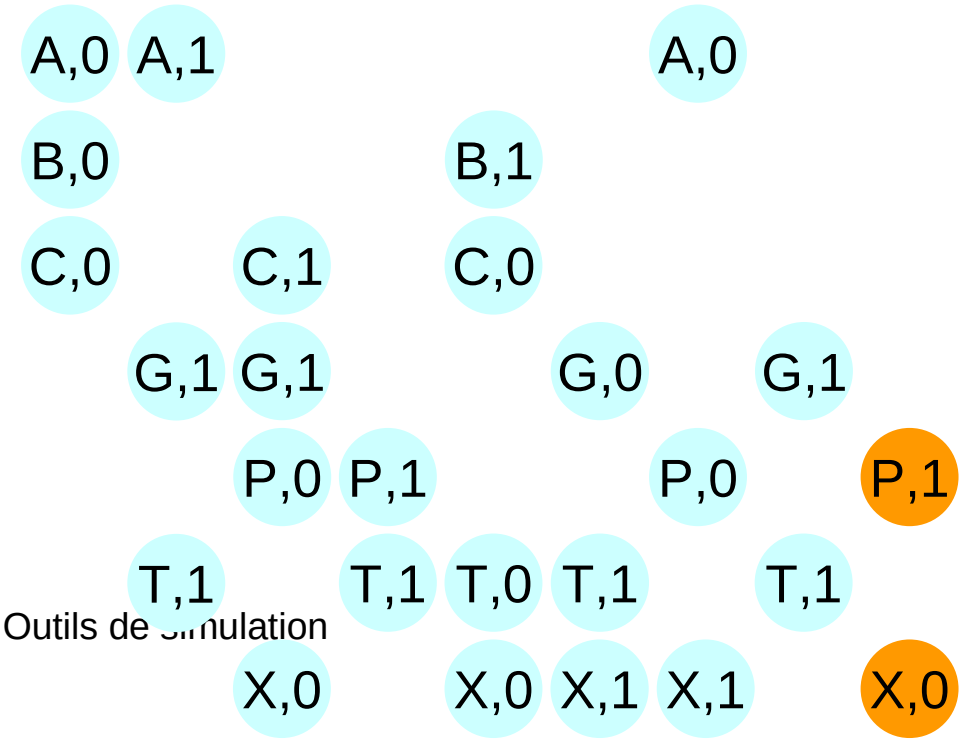
délais : 1ns pour le nand
2ns pour le xor

UPDATE

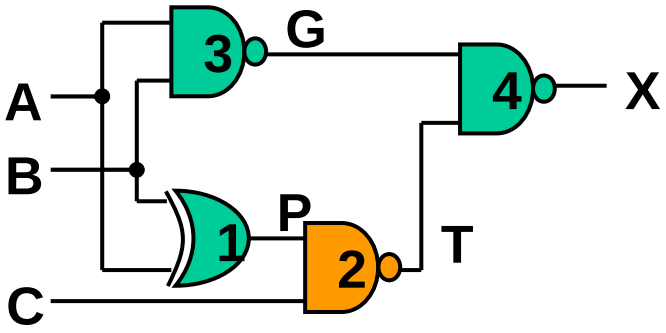
date courante

8





Simulation

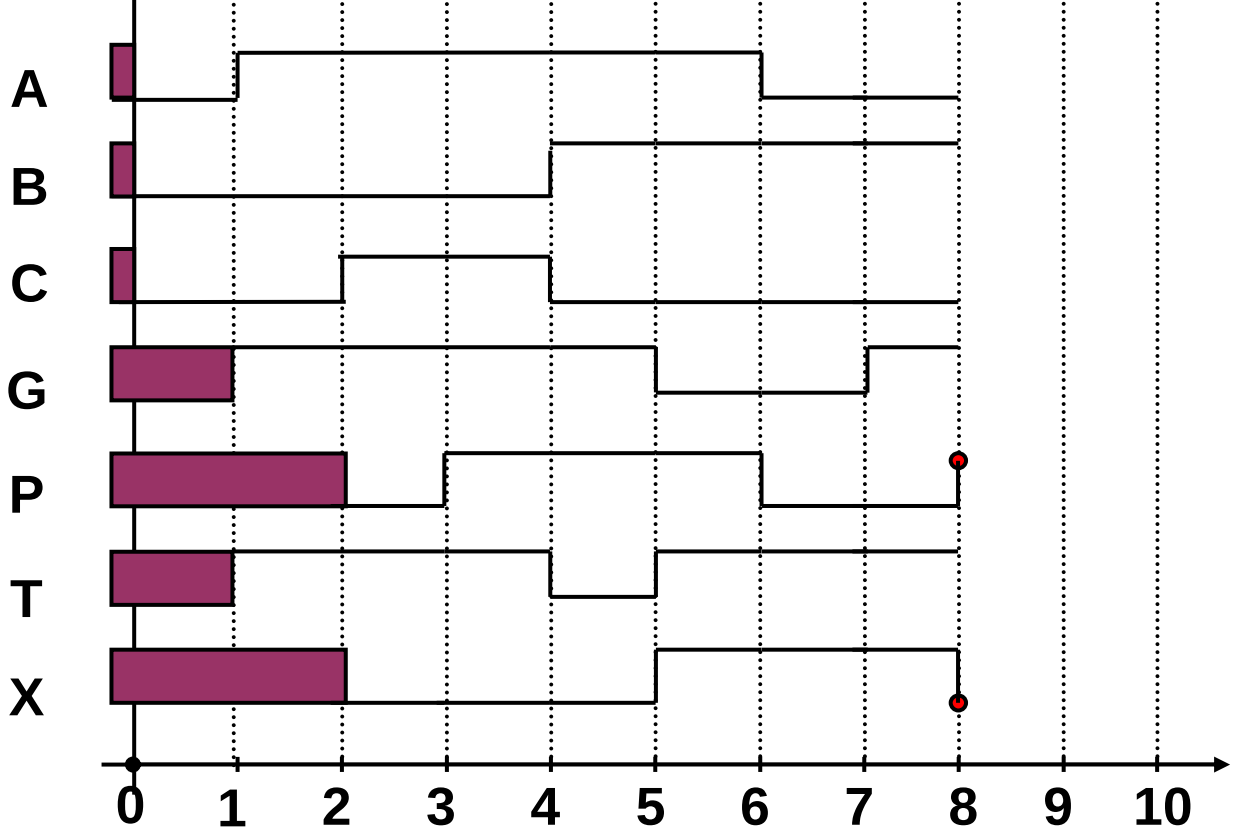


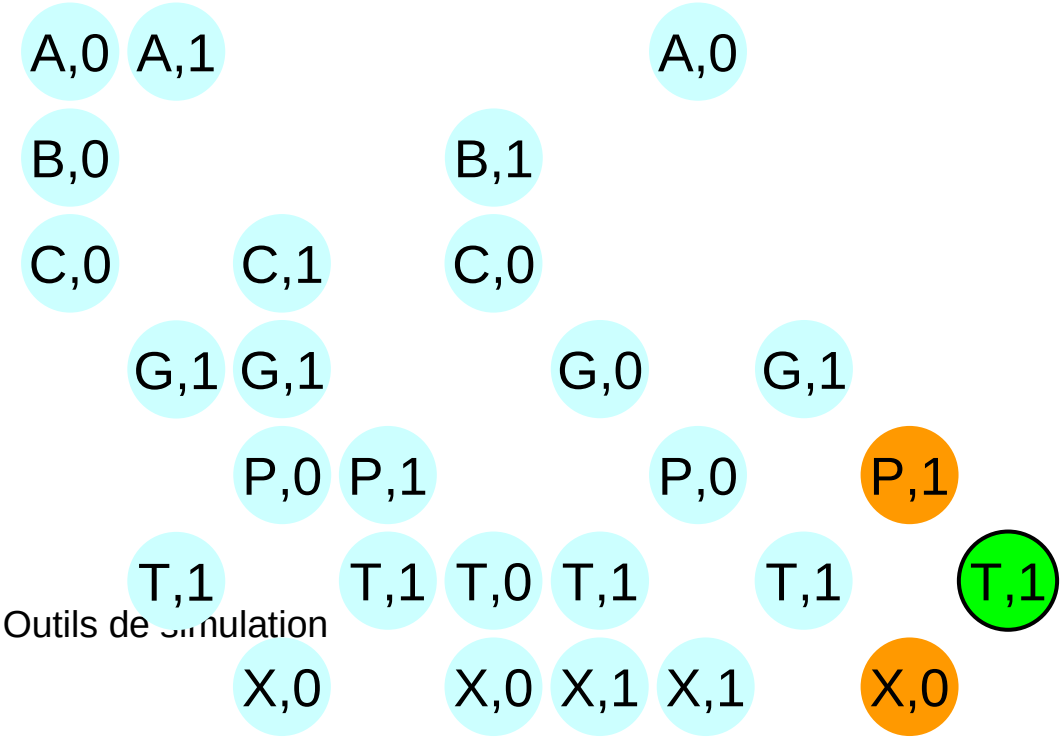
délais : 1ns pour le nand
2ns pour le xor

EXECUTE

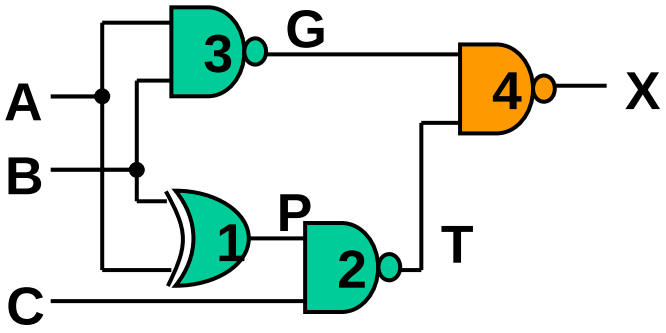
date courante

8





Simulation

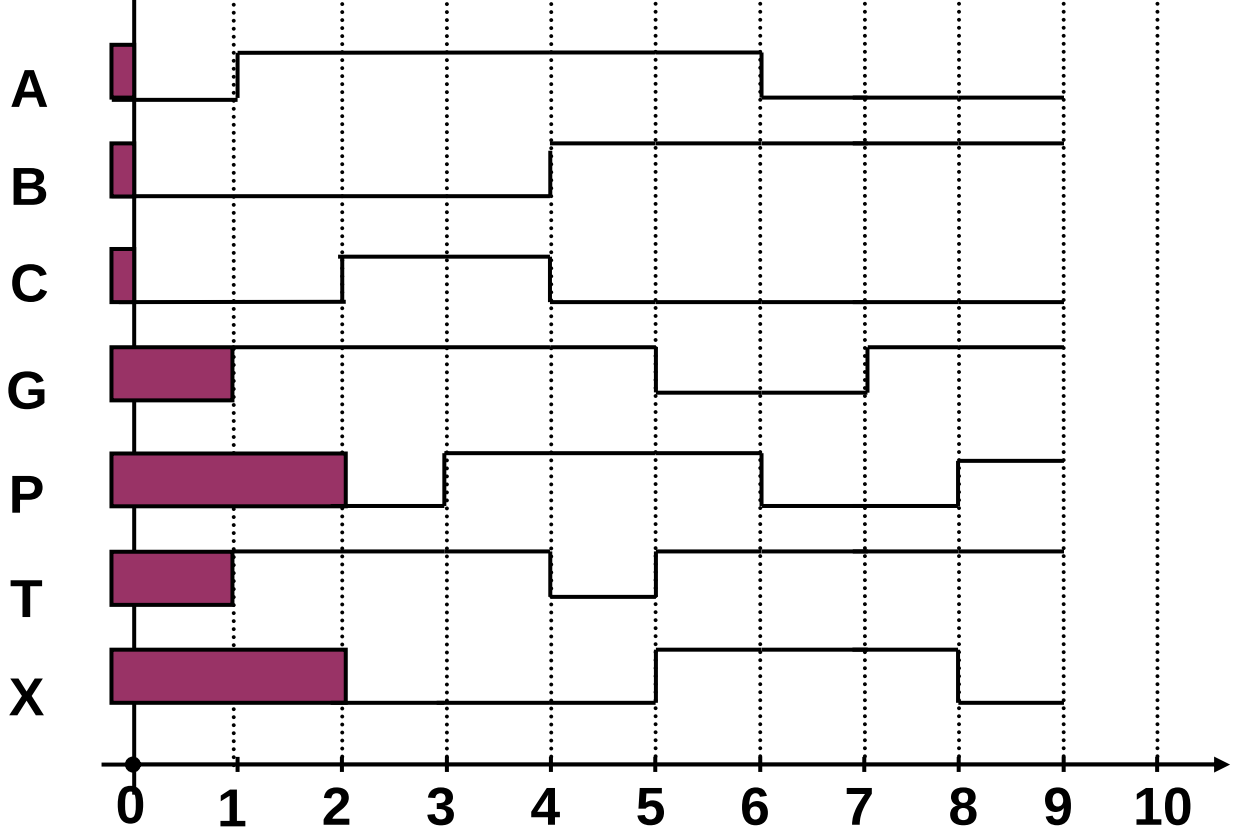


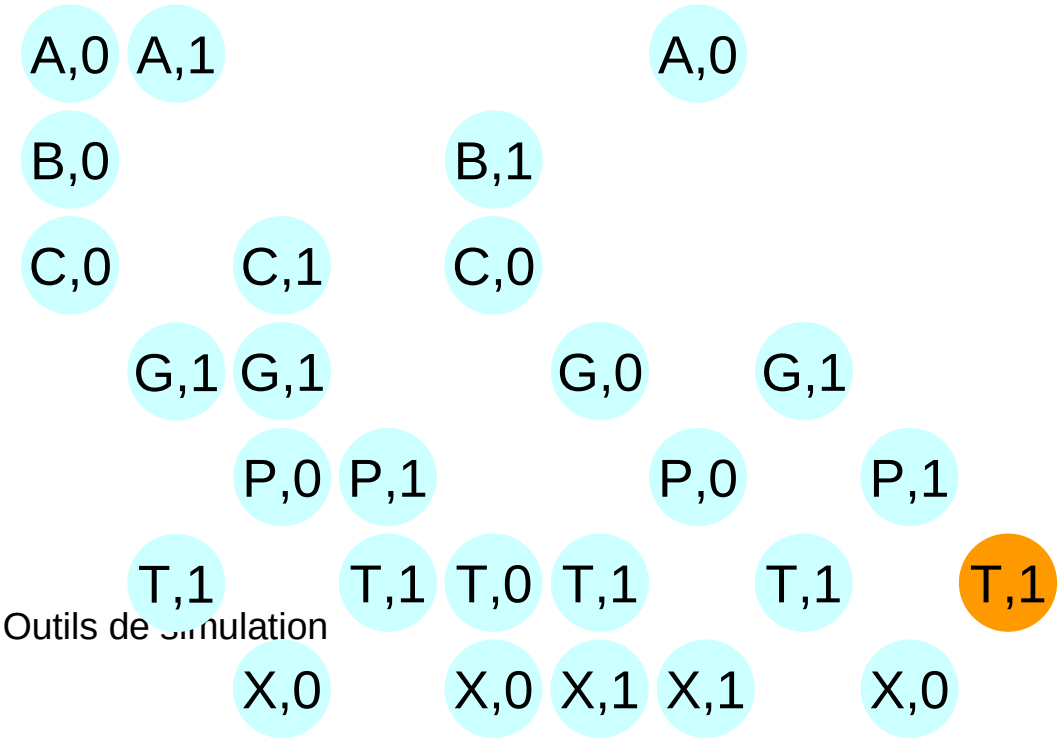
délais : 1ns pour le nand
2ns pour le xor

UPDATE

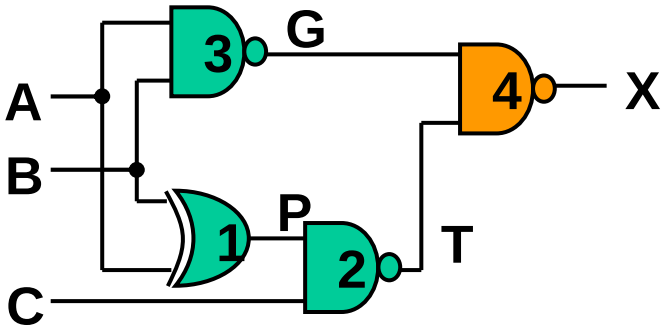
date courante

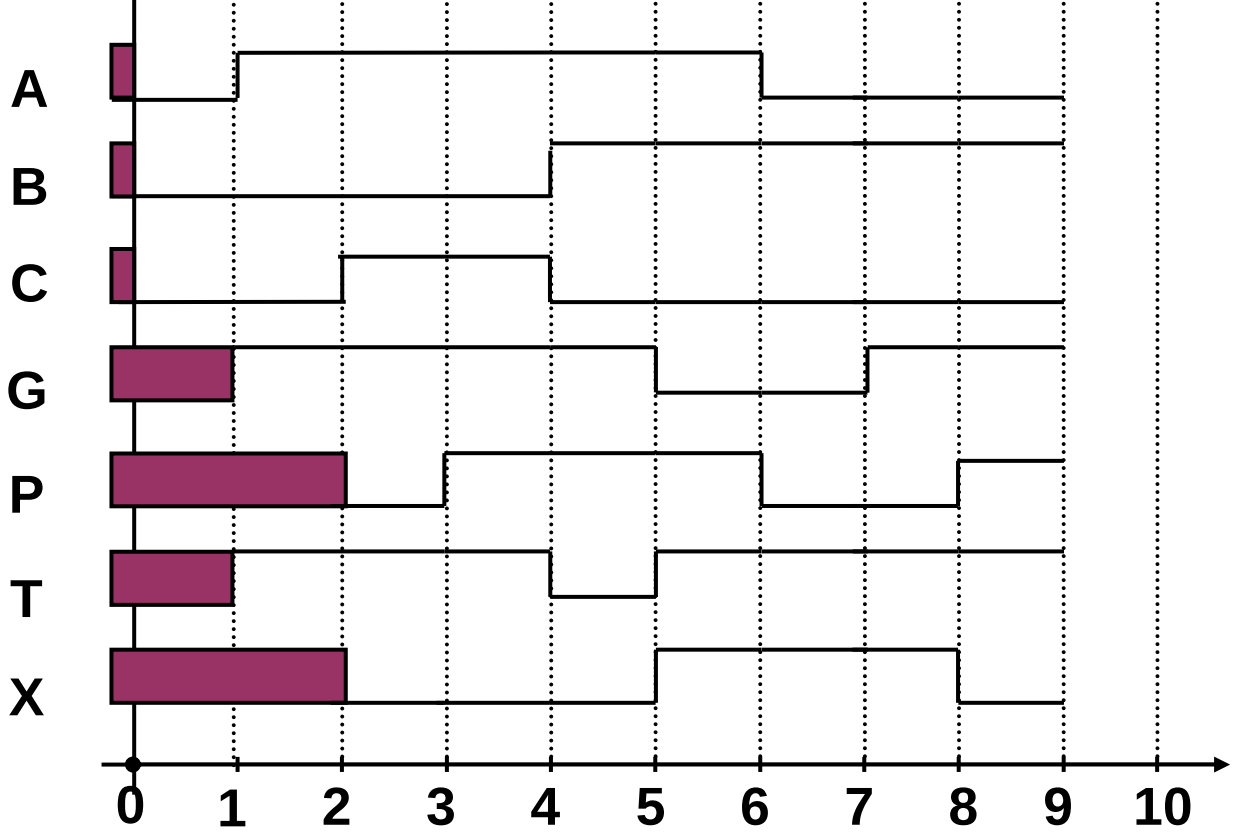
9





Simulation



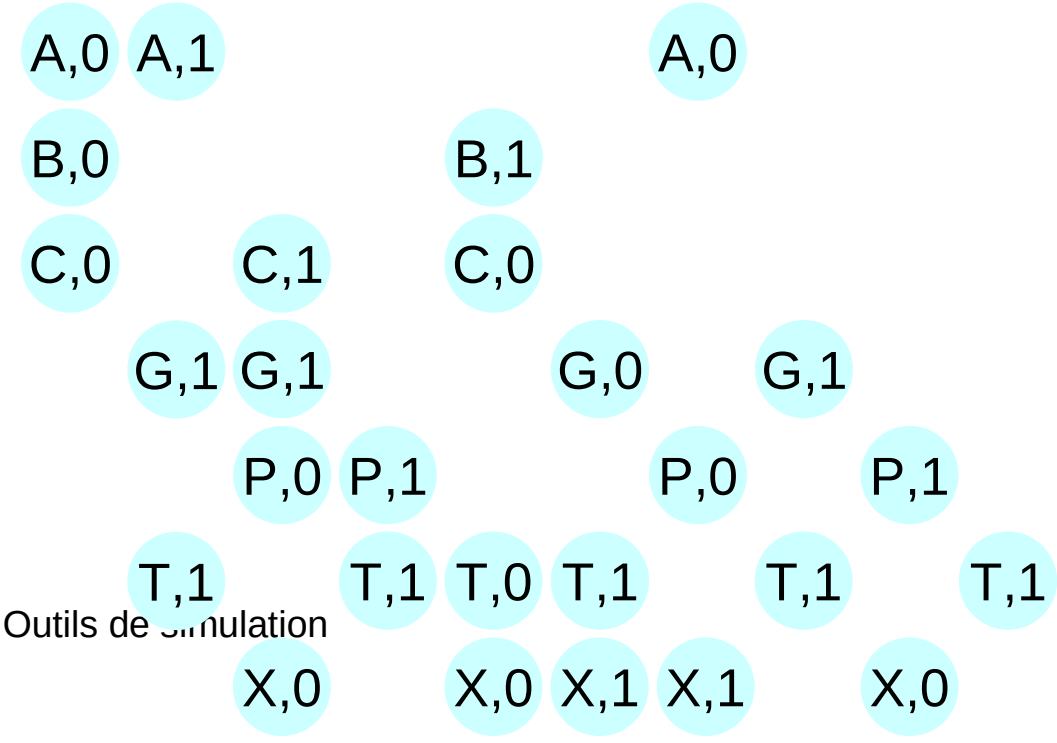


délais : 1ns pour le nand
2ns pour le xor

EXECUTE

date courante

9



Retour sur l'ajout d'une transaction

- L'ajout d'une nouvelle transaction peut conduire à l'effacement de transaction introduite avant
  - au cas où il existerait déjà une transaction sur le même signal à une date postérieure.

    Dans ce cas, les transactions postérieures ne sont plus valables. Ceci ne peut survenir que lorsqu'il existe plusieurs délais pour un même processus dépendant des entrées présentant un événement.

  - au cas où il existerait déjà une transaction sur le même signal à une date antérieure, mais très proche qui ne va pas dans le même sens et que l'on simule l'inertie du système réel.

    Dans la réalité, les changements d'état de signaux ne sont pas instantanés.

Fonction de résolution

- Un signal peut avoir plusieurs émetteurs à la condition que chacun puisse être déconnecté.

- La valeur d'un signal à émetteur multiple est calculée par une fonction de résolution qui dépend du type du signal et du nombre d'émetteurs connectés au moment de l'évaluation.

Pour ASIMUT :

- il existe trois types : **reg_bit**, **wor_bit bus**, **mux_bit bus**
- la valeur des signaux peut être à 0, 1, U (inconnu).

émetteurs / type	0	1	>1
reg_bit	mémoire	émetteur	**erreur**
wor_bit	1	émetteur	émetteur ou **erreur**
mux_bit	1	émetteur	**erreur**

Simulation sans délai

- Il faut respecter le **principe de causalité**
  ➔ On introduit la notion de **delta-délai** (delta cycle)
- Un delta-délai définit une durée infime, mais non nulle
- La durée entre deux "vrais" patterns correspond à une infinité de delta-délais.
- Le principe de la simulation reste inchangé :
  - l'échéancier est rempli avec les "vrais" patterns de simulation
  - La première date courante de la phase update correspond à un "vrai" pattern, contenant une liste de transactions qui en produisent de nouvelles un delta-délai plus tard lors la phase exécute. Ce délai est toujours strictement inférieur à la date du prochain pattern.
  - Lorsqu'il n'y a plus de transactions produites à l'échelle du delta-délai, alors la phase update passe naturellement au prochain "vrai" pattern.

Plan du cours

- Aperçu de la chaine de compilation
- Principe de la simulation à événements discrets
  - Modélisation
  - Exemple de simulation d'un circuit combinatoire
  - Extension aux bus et aux registres
  - simulation sans délai
- Le format pat
- Le simulateur ASIMUT
- Le générateur de patterns GENPAT
- Le visualisateur de patterns XPAT

format PAT

- Le format de fichier .pat utilisé par Asimut permet de décrire les patterns appliqués et les résultats attendus.

- Le format .pat contient:
  - L'interface du circuit
  - La séquence des patterns
  - Les actions sur le simulateur

- Documentation : **`man 5 pat`**

Exemple de .pat

```
in       A (0 to 15) X;
in       B (0 to 15) X;
in       Cin;
out      Cout;
signal   S (0 to 15) X;
register Accu.A (0 to 15) X;

begin

<   0 ns > pattern_0 : F0F0 0A0A 1 ?0 ?FAFA ?6DE7;
< +10 ns > pattern_1 : 0F0F F6F0 0  +  **** ?54FC;

end;
```

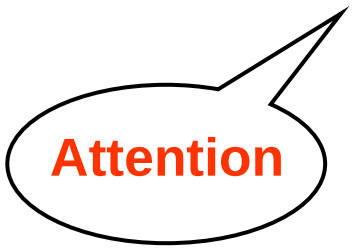
Interface .pat

mode ***nom de l'entrée-sortie*** ***format*** **[*spy*] *;* [*;*]**

- in
- out
- inout
- signal
- register

- nom
- group (nom1, nom2, …)
- chemi-nom pour les signaux ou registres internes
  - vbe : root.nom
  - vst : nom

- B
- X
- O
- *rien*



instance.nom

Exemple

```
in        A (0 to 15) X;
in        B (0 to 15) X;
in        Cin ;;
out       Cout;
signal    S (0 to 15) X;
register  Accu.A (0 to 15) X;
```

Les noms de vecteurs de signaux ont le format VHDL

Séquence de patterns

begin

[< date >] [étiquette] : valeurs_des_signaux ; [;]

…

end ;

- **date** :: [+] *nombre_entier unité*
  - unité :: ps | ns | us | ms
  - [+] :: délai depuis le précédent pattern
- **étiquette** :: sert à s'y retrouver dans la séquence de patterns
- **valeurs_des_signaux**::
  entrées = 0 | 1 | + | - | 00110 | 0256 | DEAD
  sorties prédites = ?0 | ?1 | ?+ | ?- | ?00110 | ?0256 |
  sorties non prédites = * | ****
- **[;]** :: permet d'ajouter une ligne blanche dans le fi

Actions sur le simulateur

- Initialisation d'un registre
  - il est possible d'initialiser à tout moment un registre à une valeur quelconque, au format vhdl :
    registre <= value;
- sauvegarde de l'état des signaux
  - placé entre deux patterns :
    save;
- commentaires du fichier
  - précédés par -- Ils sont simplement ignorés
  - précédés par # ils sont recopiés intacts dans le fichier de sortie.

Plan du cours

- Aperçu de la chaine de compilation
- Principe de la simulation à événements discrets
  - Modélisation
  - Exemple de simulation d'un circuit combinatoire
  - Extension aux bus et aux registres
  - simulation sans délai
- Le format pat
- Le simulateur ASIMUT
- Le générateur de patterns GENPAT
- Le visualisateur de patterns XPAT

Le simulateur ASIMUT

Documentation : **`man asimut`**

asimut [options] [root_file] [pattern_file] [result_file]

options essentielles

-b si le fichier à simuler est uniquement comportemental (.vbe)
-c asimut fait seulement une lecture du modèle
-i val initialise tous les signaux à val (0 ou 1)
-zd force la simulation sans délai

autres options

-i file initialise tous les signaux à partir du fichier **file** (sauvé par la commande save)
-inspect inst_name
produit un fichier de pattern avec les signaux à l'interface de inst_name.
-core file
produit un fichier .cor avec l'état de tous les signaux dès la première erreur.

variables d'environnement

- MBK_CATA_LIB répertoires contenant les descriptions et les patterns
- MBK_WORK_LIB répertoire de travail avec les descriptions et les patterns et où sont écrits les fichiers produits
- MBK_CATAL_NAME nom du fichier catalogue (placé dans MBK_WORK_LIB)
- MBK_IN_LO extension (type) des fichiers netlist (al ou vst)
- VH_MAXERR nombre maximum d'erreurs autorisées avant l'arrêt de la simulation

fichier CATAL

- Le fichier CATAL permet d'indiquer quelles sont les feuilles de l'arborescence dans le cas de la simulation d'une netlist.
- Les noms présents dans le fichier ont un modèle comportemental.
- format

```
bloc1 C
bloc2 C
bloc3 C
```

Plan du cours

- Aperçu de la chaine de compilation
- Principe de la simulation à événements discrets
  - Modélisation
  - Exemple de simulation d'un circuit combinatoire
  - Extension aux bus et aux registres
  - simulation sans délai
- Le format pat
- Le simulateur ASIMUT
- Le générateur de patterns GENPAT
- Le visualisateur de patterns XPAT

Langage GENPAT

- Documentation : **`man genpat`**
- Le but est d'exprimer dans un langage procédural, les transactions sur les signaux.
- C'est une bibliothèque de fonctions C:
  - pour définir l'interface
  - et les transactions
  - pour générer un fichier de patterns au format .pat
- Le langage C permet l'écriture de boucles et de fonctions
- Un script permet de lancer le compilateur C puis l'exécution du programme :
  `> genpat [-v] [-k] file`

API Genpat (essentielle)

- DEF_GENPAT("nom")

  definit le nom de fichier dans lequel les patterns seront placés.

- SAV_GENPAT()

  sauve le fichier sur disque.

- DECLAR("ident",":nb_space","format",mode,"size","option")

  déclare le nom du signal

ident	nom du signal
nb_space	nombre d'espaces entre les colonnes,
format	format d'affichage (B,O,X),
type	IN, OUT, INOUT, SIGNAL, REGISTER
size	rien, X to Y, Y downto X
option	rien, S

- AFFECT("pattern_date", "ident", "value")

  définit une transaction sur le signal

pattern_date	"nombre" date absolue de la transaction, ou "+nombre" date relative au dernier AFFECT() ou INIT()
ident	nom du signal
value	*, nombre dans la base définie par format, mélange possible

API Genpat (complément)

- INIT ("pattern_date", "ident", "value")

  identique à AFFECT mais pour initialiser un registre

- SETTUNIT("unit")

  définit l'unité de temps utilisée pour les dates : "ps", "ns", "ms", "us".

- LABEL("ident")

  affecte une étiquette pour le pattern courant (donc celui qui vient d'être affecté)

- ARRAY ("ident",…,":nb_spa","fmt",mode,"option","ident_group", 0)
  - déclare un groupe de signaux (aggrégat) de même type.

    ident, … noms des signaux composant le groupe (MSB en premier)
    nb_spa nombre d'espaces entre les colonnes,
    fmt format d'affichage (B,O,X),
    type IN, OUT, INOUT, SIGNAL, REGISTER
    option rien, S
    ident_group nom du groupe

API Genpat

fonction transformant un entier en chaine de caractères

fichier vecteurs.pat

déclaration de l'interface et des signaux internes

boucles imbriquées produisant 256 patterns

```
#include <stdio.h>
#include "genpat.h"

char *inttostr(int entier)   {
  char * str = malloc (32);
  sprintf (str, "%d",entier);
  return(str);
}

main ()  {
   int i, j, cur_vect = 0;

   DEF_GENPAT("vecteurs");

   DECLAR ("a", ":2", "X", IN, "0  to 3");
   DECLAR ("b", ":2", "X", IN, "0  to 3");
   DECLAR ("s", ":2", "X", OUT, "0 to 3");

   for (i=0; i<16; i++)
      for (j=0; j<16; j++) {
            AFFECT (inttostr(cur_vect), "a", inttostr(i));
            AFFECT (inttostr(cur_vect), "b", inttostr(j));
           cur_vect++;
      }

   SAV_GENPAT ();
}
```

Plan du cours

- Aperçu de la chaine de compilation
- Principe de la simulation à événements discrets
  - Modélisation
  - Exemple de simulation d'un circuit combinatoire
  - Extension aux bus et aux registres
  - simulation sans délai
- Le format pat
- Le simulateur ASIMUT
- Le générateur de patterns GENPAT
- Le visualisateur de patterns XPAT

Le visualisateur de patterns XPAT

XPAT est l'un des nombreux outils de visualisation

- avantage : fonctionnement simple et intuitif.
- inconvenient : peu de fonctions

`xpat [-l file]`