Outils de synthèse

syf
boom / boog / loon (+ lax)
proof

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

Plan du cours

- Rappel des objectifs de la synthèse
- Rappel du flot de conception Alliance
- pour chaque outil
    1. principe de fonctionnement
    2. la ligne de commande et les options

Objectif de la synthèse

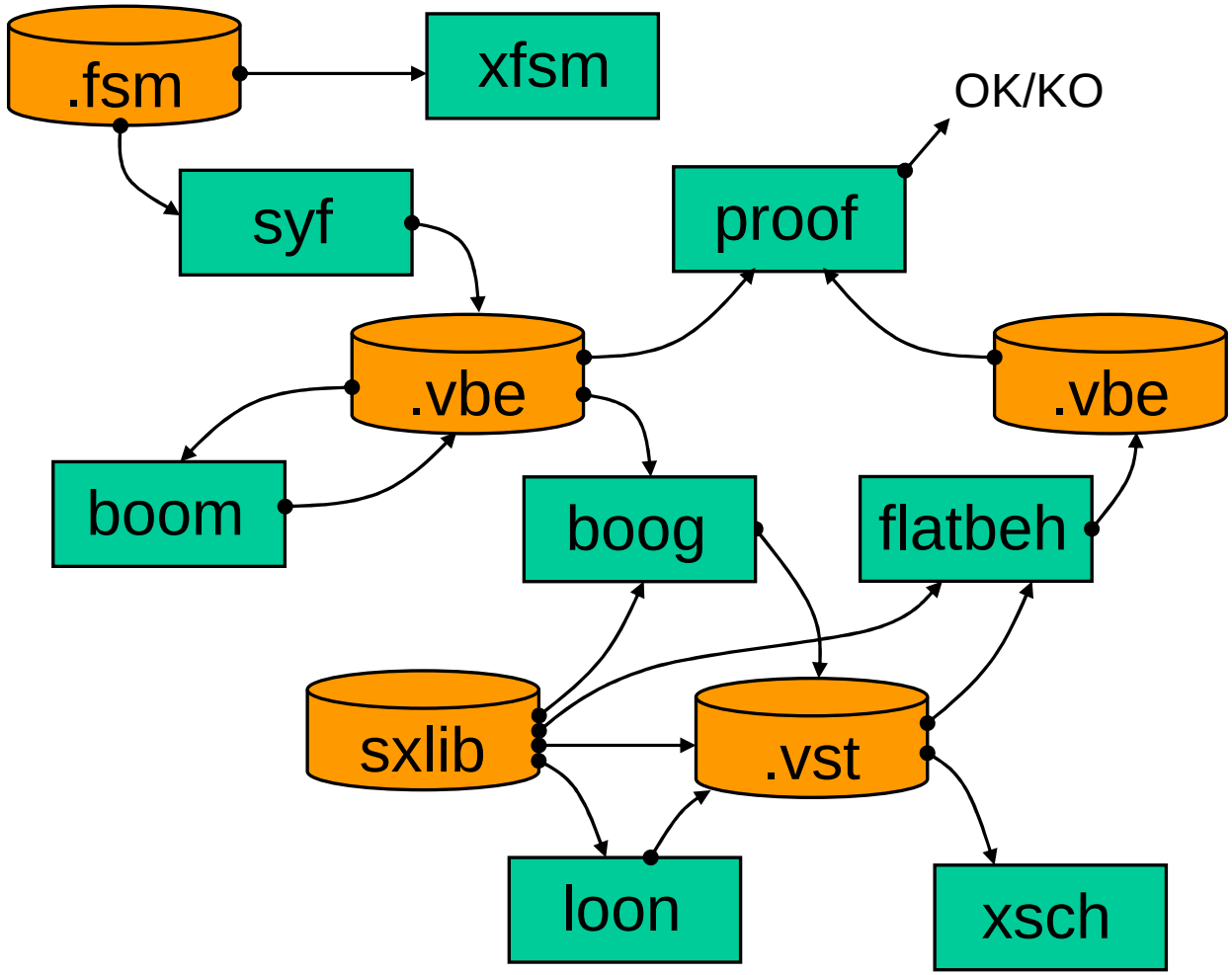
Transformer la description d'un comportement en netlist

- Le comportement est au niveau RTL (vbe)
- La netlist est un réseau de portes précaractérisées (vst)

Deux méthodes

- Automatique : le réseau de portes est obtenu par un algorithme de projection structurelle.
  - ➔ Ensemble de programmes BOOM, BOOG, ….
- Manuelle : le réseau de portes est obtenu à la main et décrit par un schéma, ou dans un langage spécifique ou encore par un programme de génération.
  - ➔ STRATUS : API spécifique / Python
  - ➔ GENLIB : API spécifique / C

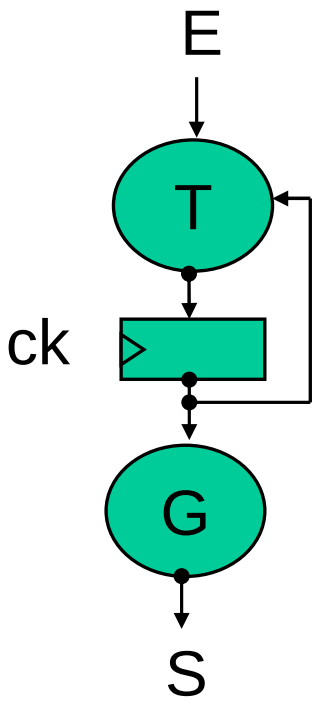
Flot des outils de synthèse Alliance



syf	synthèse fsm
boom	optimisation comportementale
boog	projection structurelle
loon	optimisation électrique
flatbeh	mise à plat
xfsm	visualisation du schéma
xfsm	visualisation du graphe fsm

syf

syf prend une machine d'états finis décrite en vhdl, choisit un codage et produit un modèle au format vbe.

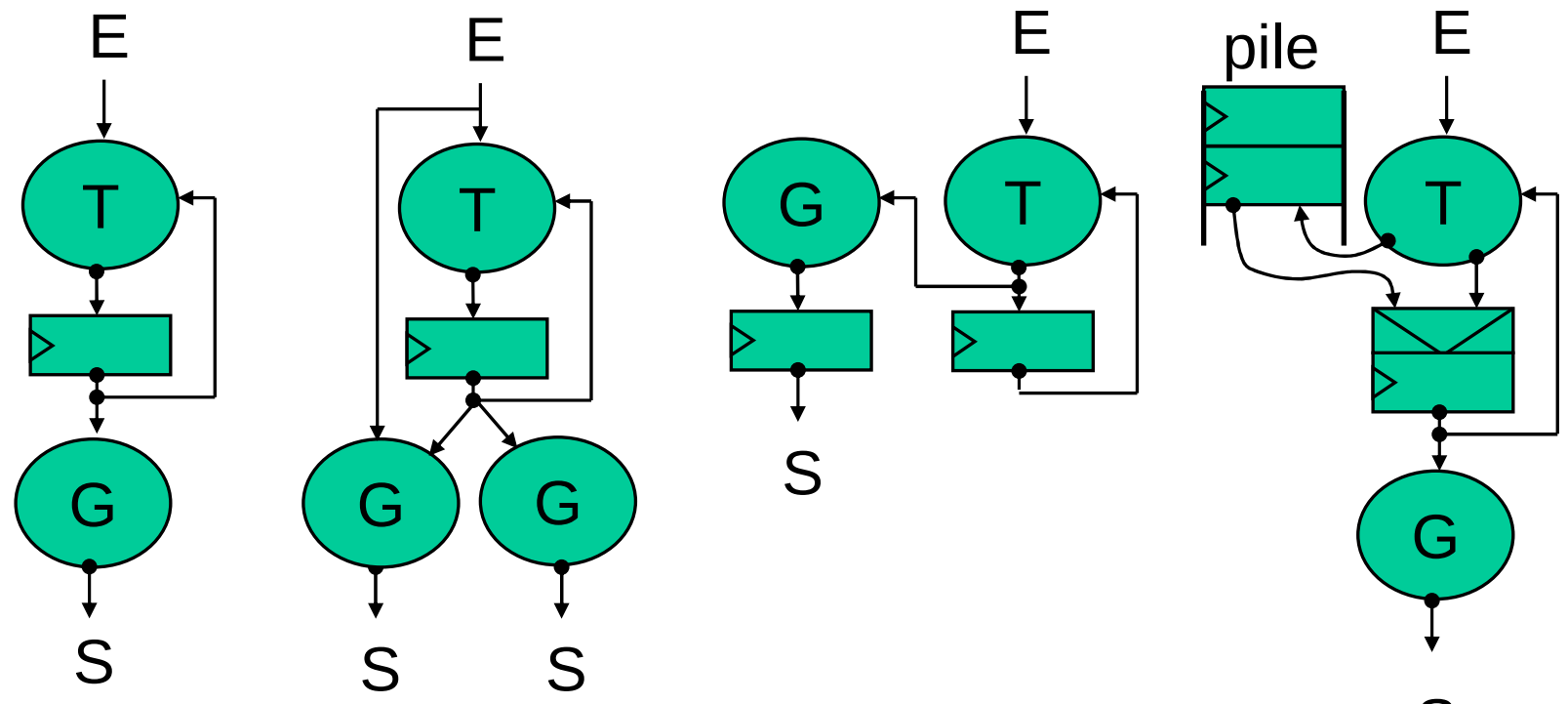


principe de fonctionnement

- le comportement est décrit en vhdl
  - un process ***Theta*** définissant définissant l'état futur en fonction de l'état courant et des entrées
  - un process ***Gamma*** définissant les sorties en fonction de l'état courant
- **syf** choisit un codage (plusieurs algos)
- **syf** détermine les expressions de tous les bits du registre d'état et tous les bits de sorties

syf

Quelques topologies d'automates.



Moore

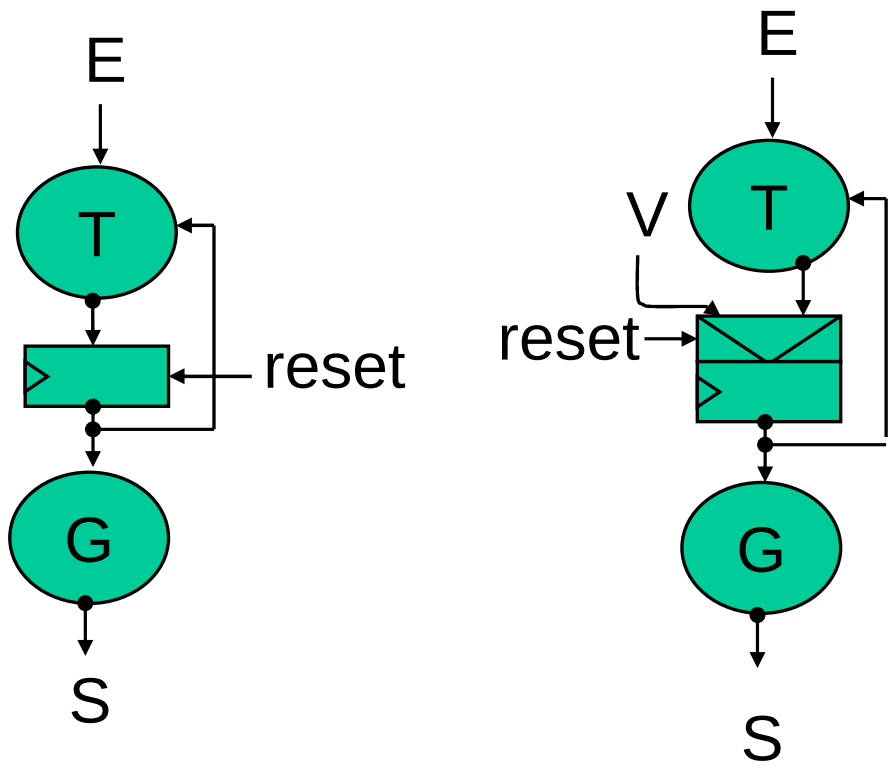
Mealy

Moore
sorties précoces

Moore
avec pile

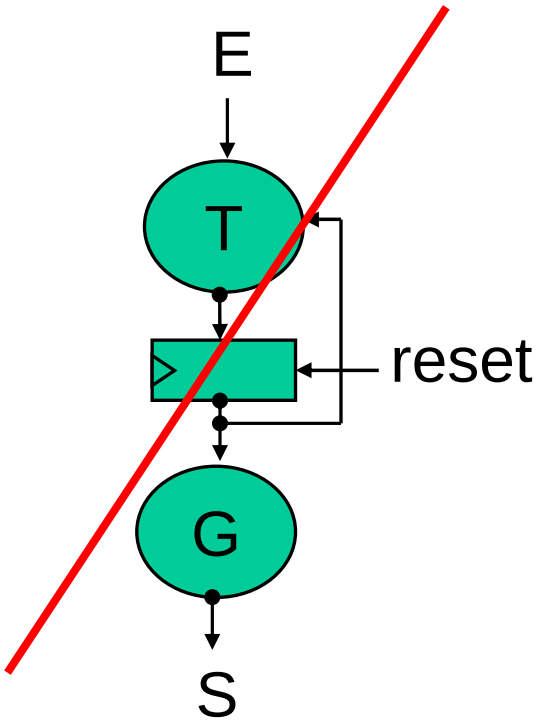
syf

L'initialisation peut être ou ne pas être synchronisée

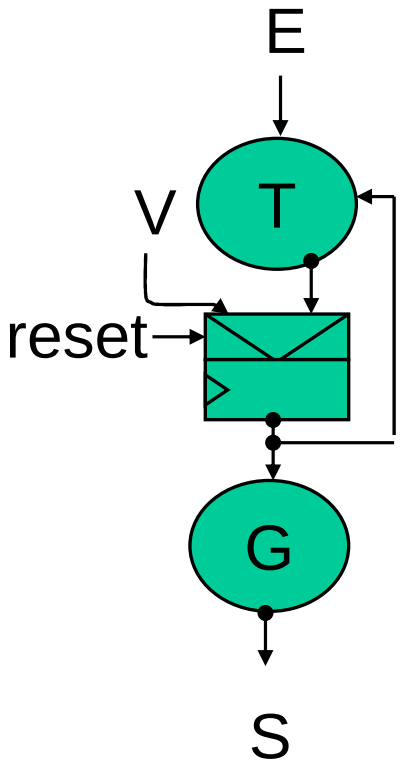


syf

L'initialisation peut être ou ne pas être synchronisée



impossible avec sxlib



syf

Codage d'un automate d'un FSM

syf -j|a|m|o|u|r [-CDEOPRSTV] input [output] [2]

paramètres:

-j -a -m : trois algos de codage (asp, jedi, mustang)
-o : codage one-hot
-u : codage utilisateur dans le fichier input.enc
liste des couples: noms_d'état code_hexa
-r : codage random

-C : vérifie la complétude et l'orthogonalité
-E : sauve l'encodage (syntaxe de –u)
-P : ajoute un scanpath
-R : utilise une ROM et un micro séquenceur
-V : verbose mode

environnement

MBK_WORK_LIB : répertoire de sortie

boom

boom prend un modèle comportemental vbe
et produit un modèle équivalent simplifié.

S <= (A and B) or B;

S <= B;

S <= (A and C) or B;
C <= (not A or B);

S <= B;

- **boom** peut supprimer des signaux auxiliaires,
  mais ne supprime pas de registres.
- **boom** supprime toutes les structures de type *with select*
  ou *when else,* le code produit est difficile à lire.
- **boom** n'est pas déterministe !

boom

Optimisation Booléenne

boom [-VTOAP] [-l num] [-d num] [-i num] [-a num]
[-sjbgpwtmorn] input [output]

paramètres

-V	: verbose
-A	: procède à des optimisations locales en conservant la majorité des signaux internes.
-P	: lit le fichier input.boom contenant - les signaux à conserver dans le fichier produit - les expressions à conserver
-l val	: effort de 0 (faible) à 3 (fort)
-d val	: type d'optimisation de 0 (délai) à 100 (surface)
-i val	*: nb d'itérations pour l'algorithme*
-a val	*: amplitude pendant le réordonnancement des bdd*
-sjbgpwtmorn	*: algorithme choisi*

environnement

MBK_WORK_LIB	répertoire de sortie

boog

boog prend une description comportementale
et produit une netlist de cellules précaractérisées.

- **boog** ne sait traiter que des expressions produites par **boom.**
- **boog** n'utilise que des portes à une sortie et de faible sortance.
- **boog** connaît les caractéristiques des portes
  grâce à des génériques dans les vbe des portes
- La projection d'un et à 8 entrées
  s <= a and b and c and d and e and f and g and h
  8 and à 2 entrées
  s <= a and (b and (c and (d and (e and (f and (g and h))))))
  2 nand à 4 entrées + 1 nor à 2 entrées
  s <= not(not (a and b and c and d) or not (e and f and g and h))
- **boog** pourrait se contenter de :
  - nand 2, nor 2, basculeD, inverseur, xor
- **boog** n'est pas déterministe !

exemple de cellule cible

```
ENTITY na2_x1 IS
GENERIC (
  CONSTANT area             : NATURAL := 1000;
  CONSTANT cin_i0           : NATURAL := 11;
  CONSTANT cin_i1           : NATURAL := 11;
  CONSTANT rdown_i0_nq      : NATURAL := 2850;
  CONSTANT rdown_i1_nq      : NATURAL := 2850;
  CONSTANT rup_i0_nq        : NATURAL := 3720;
  CONSTANT rup_i1_nq        : NATURAL := 3720;
  CONSTANT tphl_i0_nq       : NATURAL := 59;
  CONSTANT tphl_i1_nq       : NATURAL := 111;
  CONSTANT tplh_i1_nq       : NATURAL := 234;
  CONSTANT tplh_i0_nq       : NATURAL := 288;
  CONSTANT transistors      : NATURAL := 4
);
```

```
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  nq          : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END na2_x1;

ARCHITECTURE behaviour_data_flow
                    OF na2_x1 IS

BEGIN
   nq <= not ((i0 and i1)) after 900 ps;
END;
```

boog

Projection structurelle

boog [-hmxold] input output [lax]

paramètres

-h : help
-m val : optimisation de 0 (surface) à 4 (delai)
-x val : génération d'un fichier de coloration des signaux
0 (chemin critique), 1 (dégradé en fonction du délai)

environnement

MBK_CATA_LIB	liste des répertoires contenant les fichiers sources
MBK_TARGET_LIB	répertoire de la bibliothèque cible
MBK_OUT_LO	format de la netlist de sortie
MBK_WORK_LIB	répertoire de sortie

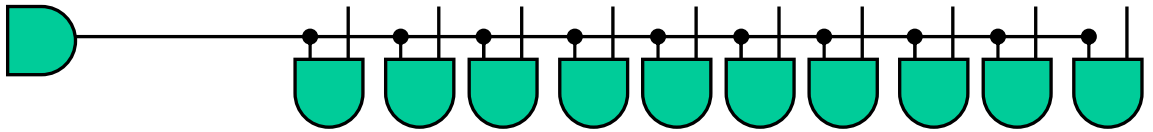
loon

loon prend une netlist de cellules précaractérisées et produit une netlist électriquement améliorée.

- **loon** ne s'autorise pas à modifier en profondeur la netlist.
- **loon** essai de n'améliorer que la chaîne critique.

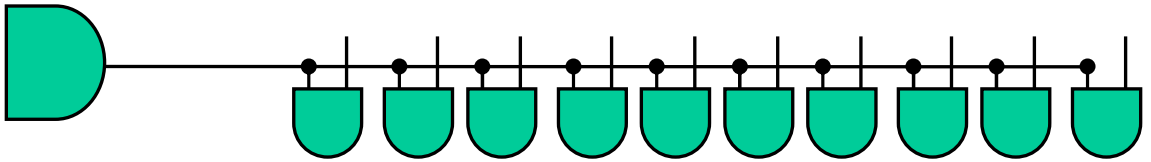
loon

Pour améliorer :

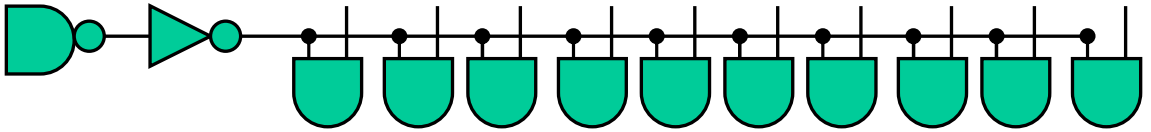


Trois techniques

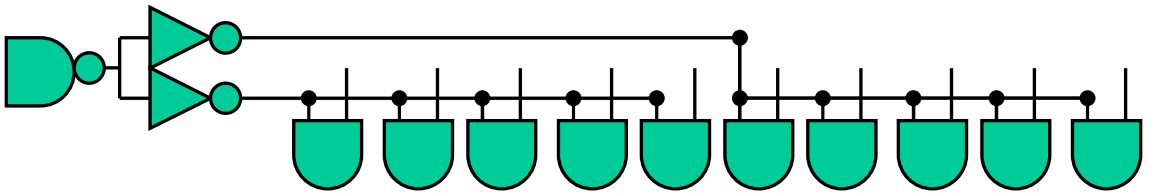
- Usage de cellules plus puissantes



- Insertion de buffers



- duplication de cellules



loon

Optimisation électrique locale

loon [-hmxlo] input_file output_file [lax_file]

paramètres

-h : help
-m val : optimisation de 0 (surface) à 4 (délai)
-x val : génération d'un fichier de coloration des signaux
0 (chemin critique), 1 (dégradé en fonction du délai)

environnement

MBK_CATA_LIB	liste des répertoires contenant les fichiers sources
MBK_TARGET_LIB	répertoire de la bibliothèque cible
MBK_IN_LO	format de la netlist d'entrée
MBK_OUT_LO	format de la netlist de sortie
MBK_WORK_LIB	répertoire de sortie

lax

Fichier de paramétrisation des outils de synthèse.

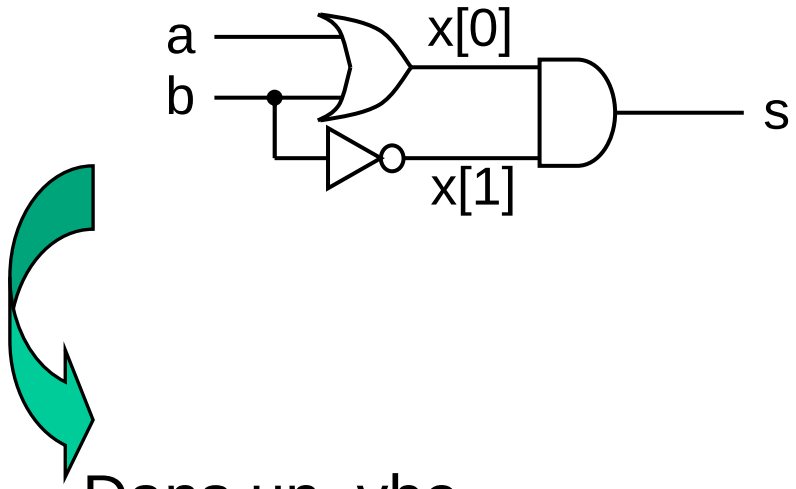
Les informations présentes sont

- le type d'optimisation général (délai ou surface)
- le degré d'optimisation (plus ou moins d'effort)
- le retard en ns de certaines entrées
- les sorties qu'il faut optimiser en priorité
- les signaux auxiliaires à conserver
- les charges capacitives sur les entrées et les sorties (en fF)
- les impédances des entrées (en ohms)
- le nombre de buffers à ajouter sur certaines entrées

flatbeh

flatbeh réalise la mise à plat d'une netlist et produit le modèle comportemental correspondant.

Dans un .vst



Dans un .vbe

```
s <= (not b) and (a or b);
```

*(on aura intérêt à utiliser **boom** pour simplifier le résultat)*

flatbeh

Mise à plat comportementale

flatbeh root_structural_file [output_file]

paramètre

CATAL : fichiers contenant les cellules feuilles

environnement

MBK_CATA_LIB	liste des répertoires contenant les fichiers sources
MBK_IN_LO	format de la netlist d'entrée
MBK_OUT_LO	format de la netlist de sortie
MBK_WORK_LIB	répertoire de sortie

proof

proof compare formellement deux descriptions comportementales et affiche les différences.

- Les deux descriptions doivent avoir
  - les mêmes entrées/sorties
  - les mêmes registres
- Chaque description est représentée par une structure ROBDD (Reduced Oriented Binary Decision Diagram)
- La représentation d'une expression Booléenne par un ROBDD est canonique
  $\Rightarrow$ si les représentations sont superposables, c'est qu'elles sont égales

ROBDD

Étant donnée une fonction Booléenne F (i0, i1, i2, i3, …)

Si on choisit un ordre pour les variables
(par exemple l'ordre alphabétique)

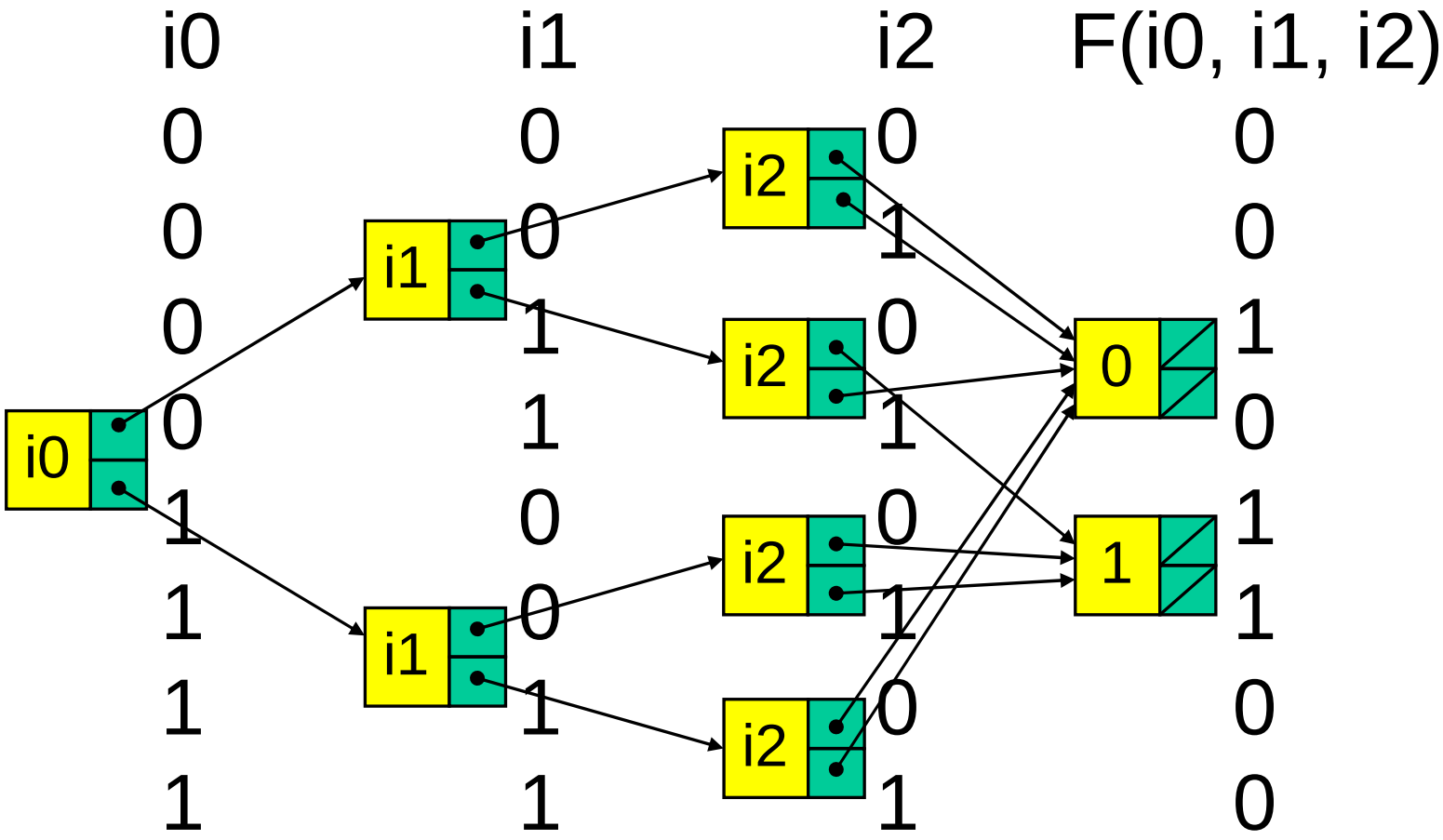
Si on choisit un ordre pour lister les combinaisons de variables
(par exemple l'ordre naturel des entiers)

Alors il n'existe qu'une seule table de vérité représentant F

i0	i1	i2	F(i0, i1, i2)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Un ROBDD peut être vu comme une représentation compacte d'une table de vérité.

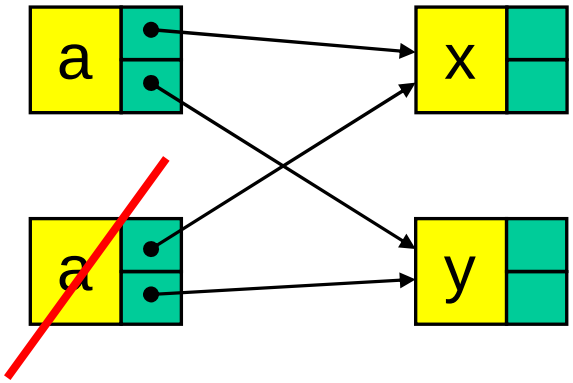
BDD


i0
i1
i2
F(i0, i1, i2)
0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1
0 0 1 1 1 1 0 0
i0
i1
i1
i2
i2
i2
i2
0
1

Réduction

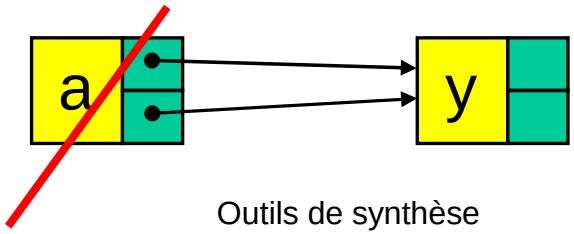
Les nœuds doivent être uniques :

- quand on demande la création d'un nœud,
  on vérifie s'il n'existe pas déjà, si oui on utilise l'existant

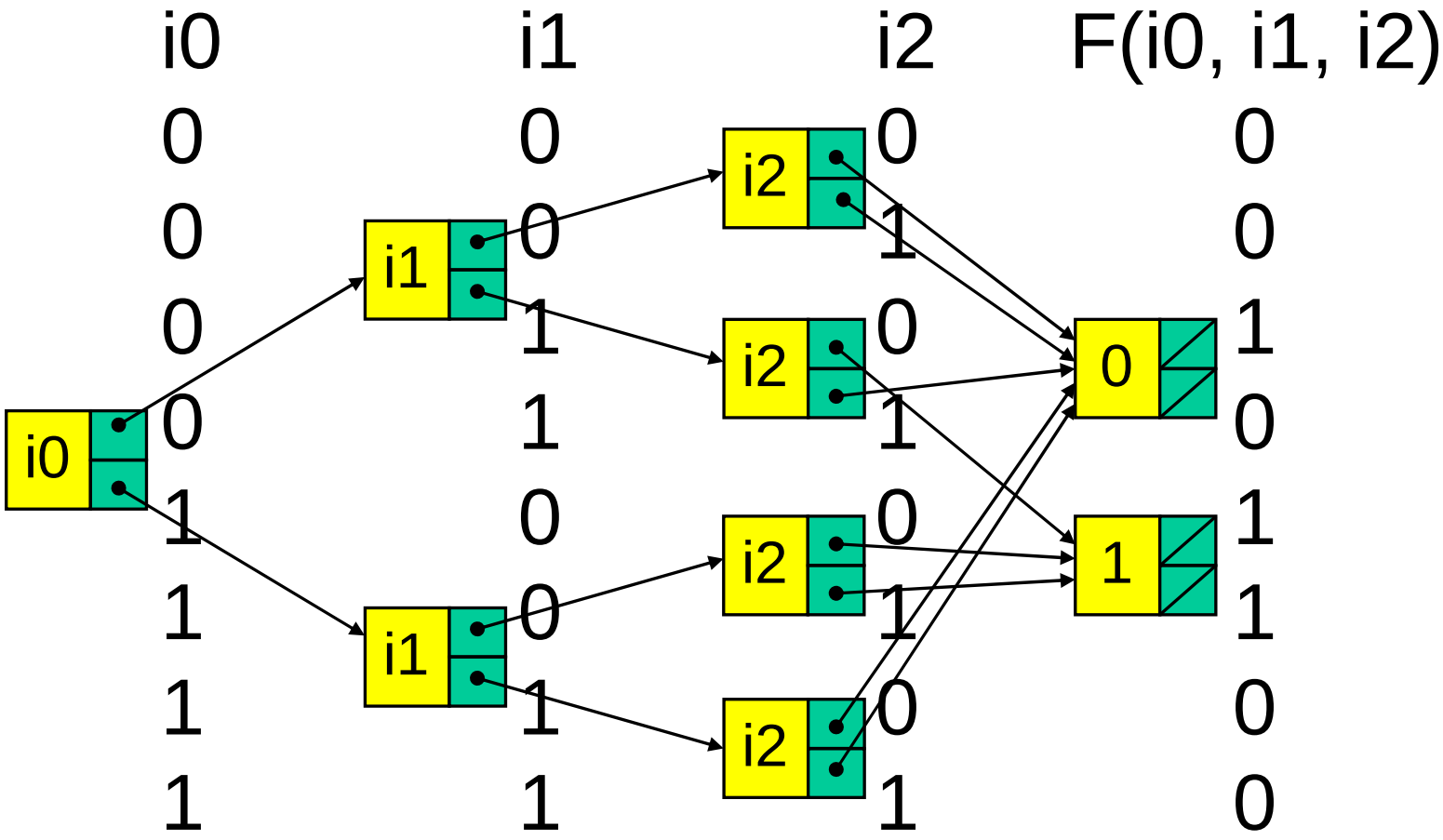


Les nœuds ne doivent pas être redondants :

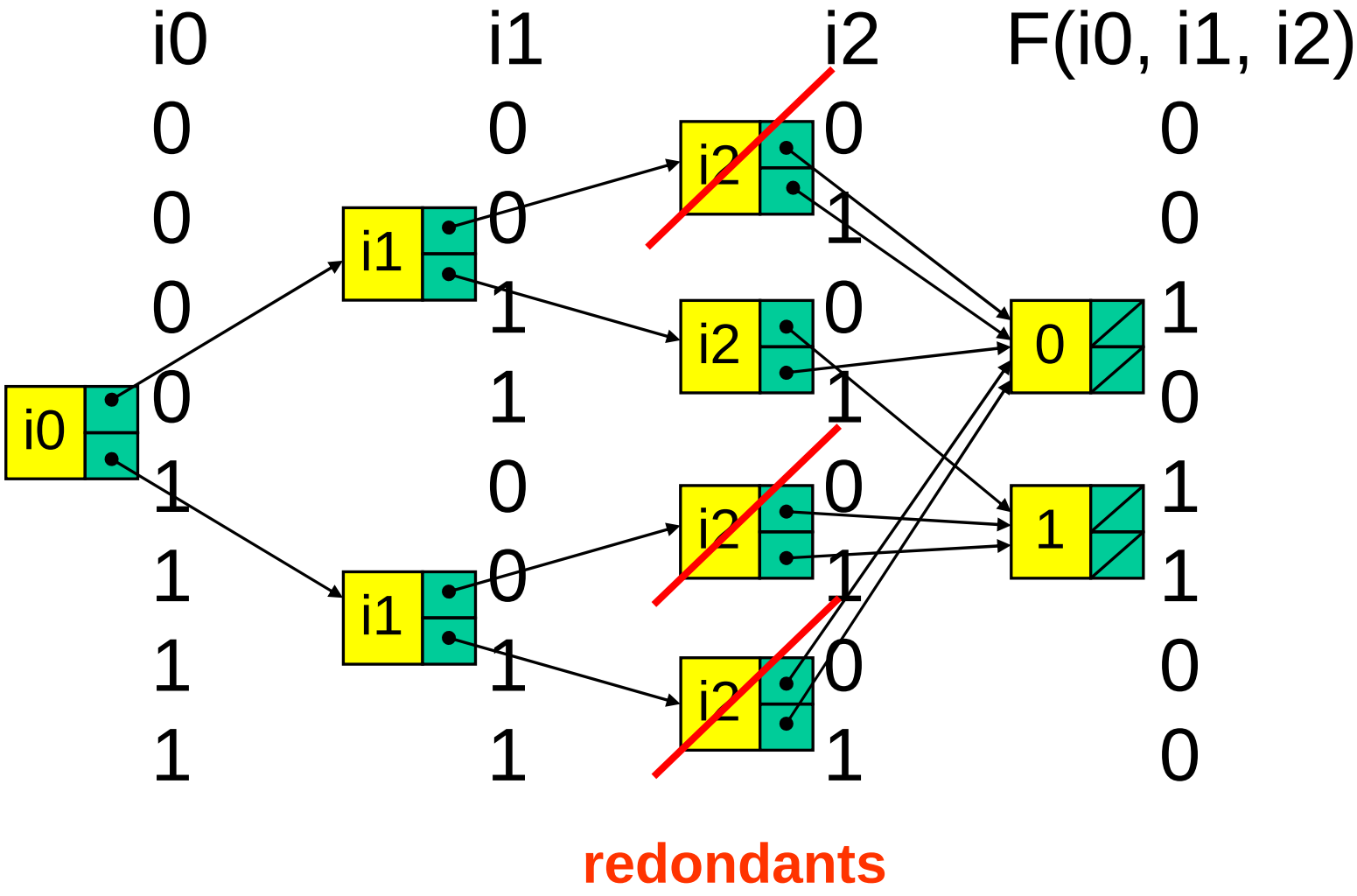
- le nœud doit servir à quelque chose
- la fonction qui passe part un nœud redondant ne dépend
  pas de la variable représentant le noeud



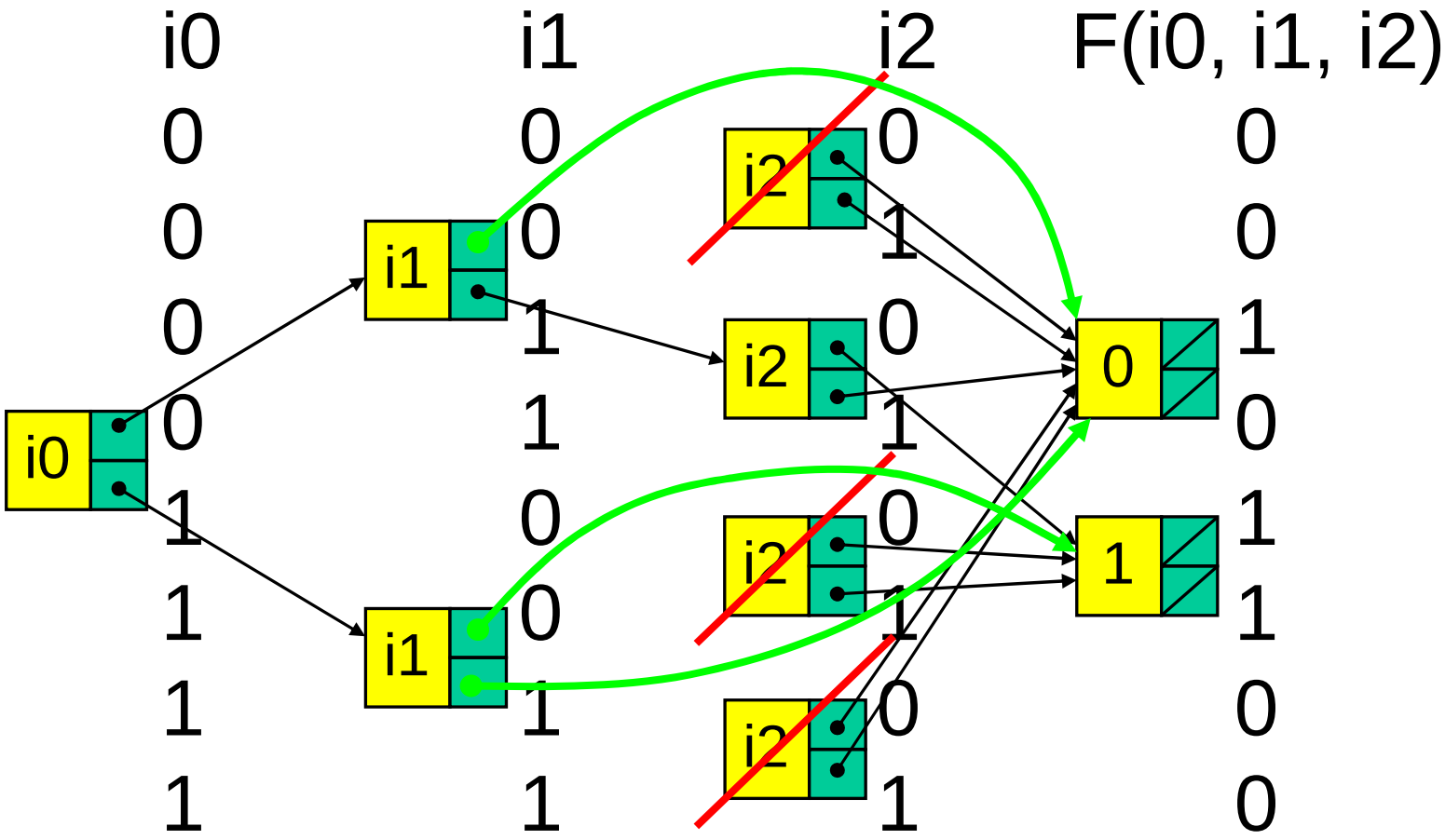
BDD


i0
i1
i2
F(i0, i1, i2)
0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1
0 0 1 1 1 1 0 0
i0
i1
i1
i2
i2
i2
i2
0
1

BDD

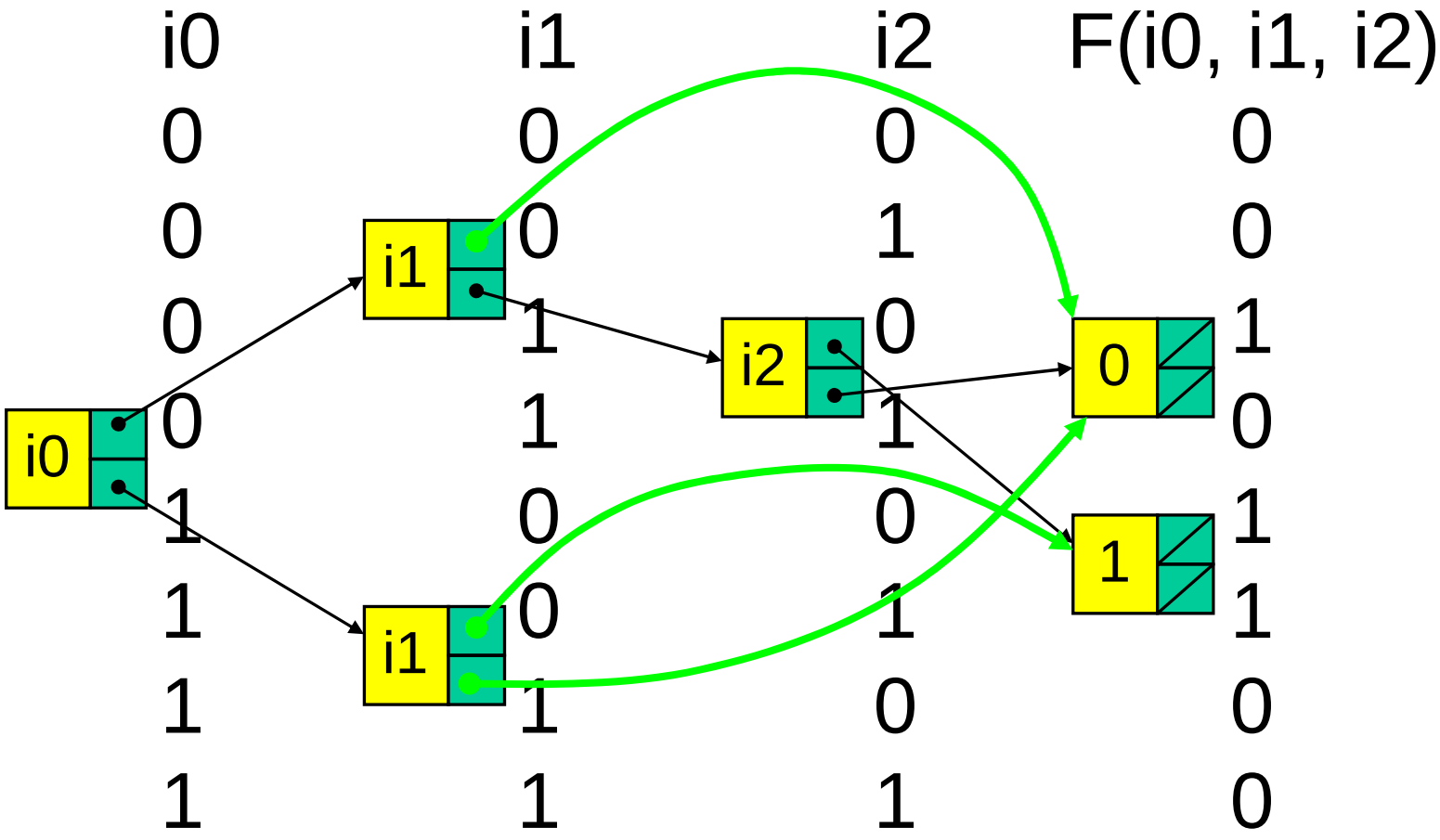

i0
i1
i2
F(i0, i1, i2)
0 0 0 0 0 0 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1
0 0 1 1 0 0 1 1
i0
i1
i1
i2
i2
i2
i2
0
1
redondants

BDD



redondants

ROBDD



En pratique, la méthode de construction des ROBDD ne passe pas par une table de vérité que l'on réduit !

Sinon on explose la mémoire !

ROBDD

Avantages

- La propriété de canonicité va permettre à proof de comparer très simplement deux expressions Boolèenes.
- Le parcours d'un ROBDD est très simple et peut permettre au simulateur d'accélérer ses évaluations.

MAIS

- Un mauvais choix pour l'ordre des variables peut entrainer une explosion combinatoire du nombre de nœuds nécessaires à la représentation d'une fonction

  $\Rightarrow$

  Pour une fonction à n variables on peut avoir besoin de 2^n nœuds.

proof

Comparateur formel

proof [-a] [-d] file1 file2

paramètres

-a : conserve tous les signaux auxiliaires présents dans les deux descriptions

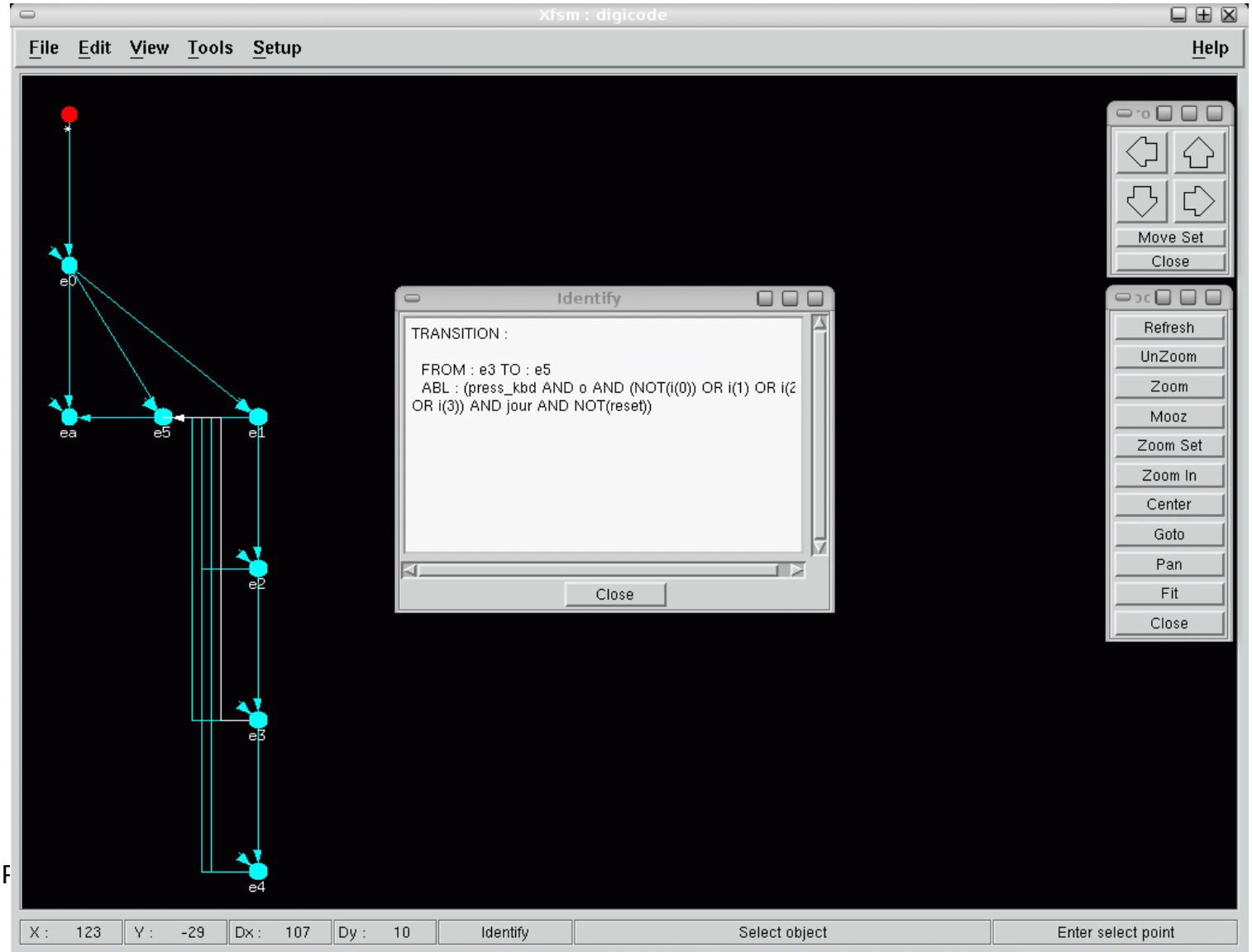
-d : donne des détails sur les différences

environnement

MBK_CATA_LIB	liste des répertoires contenant les fichiers sources
MBK_WORK_LIB	répertoire de sortie

xfsm

xfsm représente le graphe d'états d'un automate.



UF

xsch

xsch représente une netlist par un schéma