

PROGRAMMER LES ASICS EN LANGAGE VHDL



(UNE INTRODUCTION AU LANGAGE VHDL)

C. Dupaty
Professeur de génie électrique
dupaty@unimeca.univ-mrs.fr

Table des matières

1.	VHDL et logique combinatoire.....	2
1.1.	Premier programme en VHDL	2
2.	Prise en mains d'OrCAD Express V9.0	2
2.1.	Création VHDL	2
2.2.	Simulation fonctionnelle	2
2.3.	Simulation temporelle.....	2
2.4.	Création du composant deuxversquatre	2
3.	Éléments du langage	2
3.1.	Signaux, constantes, variables	2
3.2.	Les types d'objets	2
3.3.	Les opérateurs	2
3.4.	Les attributs.....	2
4.	La description séquentielle	2
5.	VHDL et logique séquentielle	2
5.1.	Bascules et verrou	2
5.2.	les compteurs.....	2
6.	Les machines d'état.....	2
6.1.	Machine d'état à sorties synchrones (Moore).....	2
6.2.	Registres à décalage	2
6.3.	UART	2
7.	Présentation de MAX+II et du KIT UP1 d' ALTERA	2
7.1.	La série MAX 7000.....	2
7.2.	Structure interne du MAX7000.....	2
8.	Exercices sur MAX+II	2
8.1.	Prise en mains de l'outil UP1 associé à MAX+PLUSII.	2
8.1.1.	Assignation et compilation	2
8.1.2.	Programmation du circuit cible.....	2
8.2.	Clignoteur 1S	2
8.3.	Comptage	2
8.4.	Création d'un projet autour de fonctions.....	2
8.5.	Feux tricolores	2
8.6.	Utilisation de TIMER : gestionnaire de parking	2
8.6.1.	Macro_Fonctions.....	2
8.7.	La série FLEX 10K.....	2
9.	Bibliographie	2

Objectifs du stage :

- Découvrir les concepts et les éléments de syntaxe de base de la programmation de PLD en VHDL
- Mettre en œuvre des PAL et des PLD (types 16L8 et 22V10) à l'aide de l'outil Express d'OrCAD et d'un programmeur de PAL
- Mettre en œuvre un CPLD (MAX7000) à l'aide de l'outil UP1 d'ALTERA

1^{er} jour : VHDL : découverte du langage, descriptions combinatoires, process, TP sur OrCAD Express, programmation d'un PAL 16L8
2^{ème} jour : VHDL : descriptions séquentielles, machines d'état TP sur OrCAD Express, programmation d'un PLD 22V10
3^{ème} jour : Développement autour du CPLD MAX7000 d'ALTERA sur KIT UP1, exemples d'applications.



Dessins PAL © AMD 1988

VHDL

Very high speed integrated circuits Hardware Description Language

Le VHDL est similaire à un langage de programmation

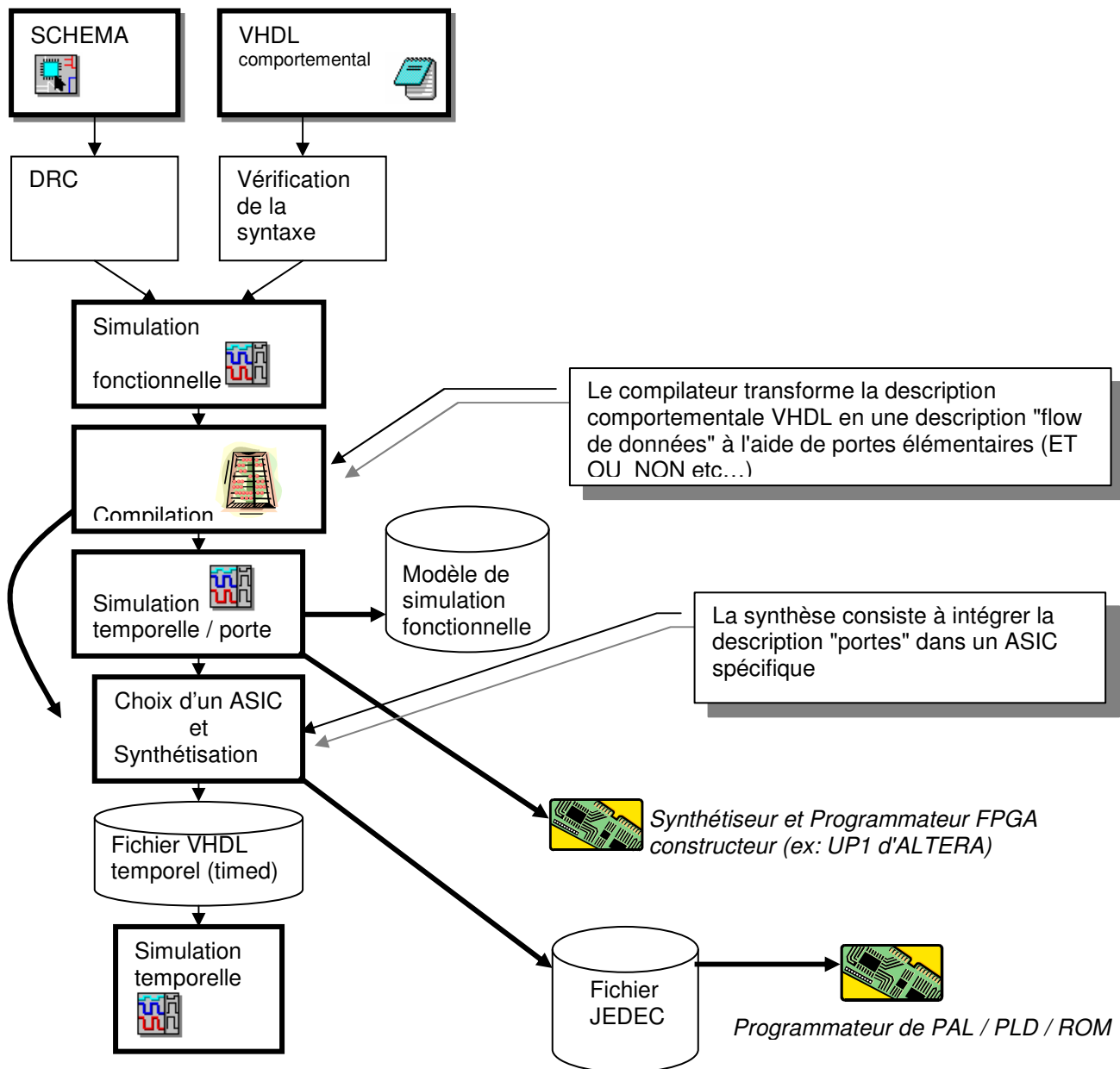
Le langage VHDL ne permet pas de générer des d'instructions séquentielles destinées par exemple à un micro-processeur.

Le langage VHDL permet de câbler entre elles des fonctions logiques intégrées dans un ASIC

Il existe deux standards VHDL 87 e 93 normalisés IEEE correspondant à la naissance du langage et à une évolution majeure (principalement sur les types et les bibliothèques)

VHDL permet le développement d'ASIC et la création de modèles de simulation

Seule une partie du langage est synthétisable dans les ASIC



A lire : Les ASICS, cours sur les PAL/PLD de JP Piau

1. VHDL et logique combinatoire

1.1. Premier programme en VHDL

Description par flot de données

-- VHDL Decodeur 2 vers 4

Un commentaire commence par --

**Library ieee;
Use ieee.std_logic_1164.all;**

Déclaration des librairies de fonctions utilisées

**ENTITY dec24 is
 PORT (
 S : OUT STD_LOGIC_VECTOR (3 downto 0);
 E : IN STD_LOGIC_VECTOR (1 downto 0));
 END dec24;**

Le nom des signaux qui rentrent et sortent du composant
Ici deux vecteurs de bit (en fait des bus) un de deux et un de trois bits
E contient les bits E(0) et E(1)

**ARCHITECTURE combinatoire OF dec24 IS
 signal S0, S1, S2, S3 : STD_LOGIC ;
 BEGIN**

**S0 <= not(E(0)) and not (E(1)) ;
S1 <= E(0) and not(E(1));
S2 <= not(E(0)) and E(1);
S3 <= E(0) and E(1) ;
S <= S3 & S2 & S1 & S0;**

Description de ce que fait le composant.
Opération whih – select
Le symbole <= représente une connexion physique à un signal, on le prononce « reçoit »
S vaudra 0001 lorsque E vaudra 00
S vaudra 0010 lorsque E vaudra 01
Etc...

END combinatoire;

Remarques :

Les types d'un port sont **IN** **OUT** **INOUT** et **BUFFER** (sortie rebouclée) ils sont accompagnés de **STD_LOGIC** pour les bits ou **STD_LOGIC_VECTOR** pour les bus lors de l'utilisation de la bibliothèque **ieee**.

E et S sont des signaux électriques de type BIT ('0' ou '1') regroupés en BUS.

L'ordre importe dans la déclaration des bits d'un bus

S OUT (3 down to 0) implique que S(3) représente les poids forts

S OUT (0 to 3) implique que S(3) représente les poids faibles

L'écriture **S<= "0001"** indique que S(3) reçoit '0' S(2) reçoit '0' S(1) reçoit '0' et S(0) reçoit '1'

Autres solutions pour le décodeur

Descriptions comportementales

<= when else

```
-- VHDL created by OrCAD Express
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Decode is
  PORT (
    S   : OUT STD_LOGIC_VECTOR (3 downto 0);
    E   : IN STD_LOGIC_VECTOR (1 downto 0));
END Decode;

ARCHITECTURE combinatoire OF Decode IS
BEGIN
  S <= "0001"    WHEN E = "00"
    else "0010"  WHEN E = "01"
    else "0100"  WHEN E = "10"
    else "1000" ;
END combinatoire;
```

TYPE std_ulogic IS ('U', -- Uninitialized 'X', -- Forcing Unknown '0', -- Forcing 0 '1', -- Forcing 1 'Z', -- High Impedance 'W', -- Weak Unknown 'L', -- Weak 0 'H', -- Weak 1 '-' -- Don't care);

Avec WITH - SELECT

```
-- VHDL created by OrCAD Express
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Decode is
  PORT (
    S   : OUT STD_LOGIC_VECTOR (3 downto 0);
    E   : IN STD_LOGIC_VECTOR (1 downto 0));
END Decode;

ARCHITECTURE combinatoire OF Decode IS
signal S0, S1, S2, S3 : STD_LOGIC ;
begin
with E select
  S <= "0001" when "00",
      "0010" when "01",
      "0100" when "10",
      "1000" when "11",
      "----" when others ;
END combinatoire;
```

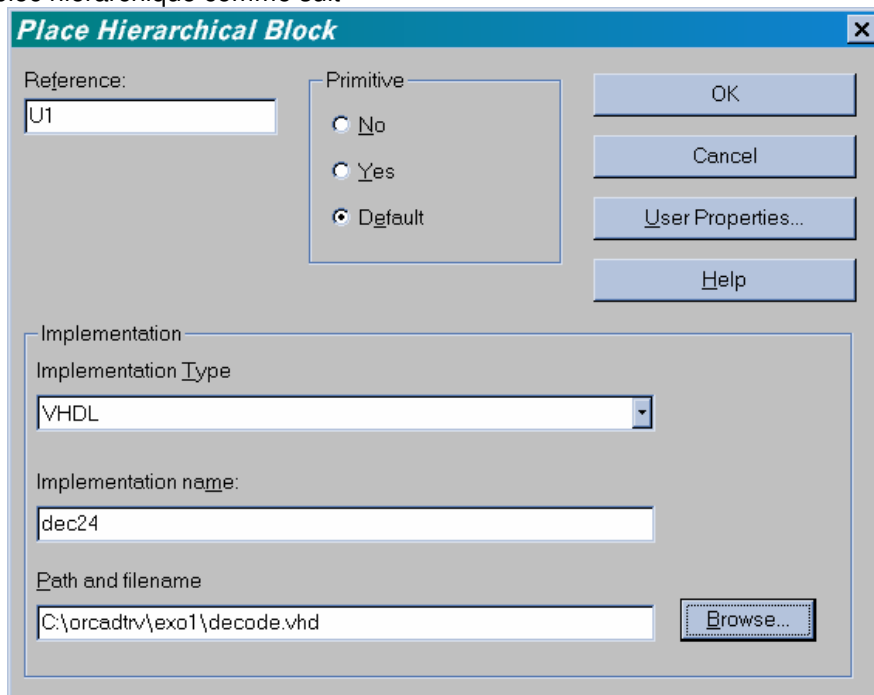
2. Prise en mains d'OrCAD Express V9.0

Cet exercice propose la réalisation d'un simple décodeur 2 vers 4, sa simulation fonctionnelle, sa synthèse dans un PAL 16L8, sa simulation temporelle puis la création d'un nouveau composant

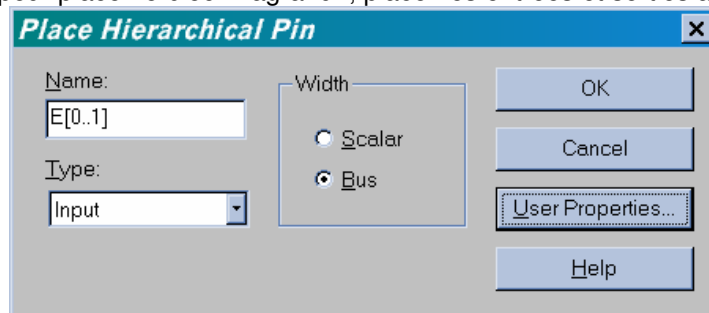


2.1. Création VHDL

- ⇒ Lancer OrCAD Capture
- ⇒ File New Project Programmable Logic Wizard
- ⇒ A l'aide de **Browse**, créer un répertoire c:\orcadtrv\exo1.
- ⇒ Nommer le projet **deuxversquatre** puis OK
- ⇒ Sélectionner **Simple PLD GAL/PAL/PROM**
- Dans capture les barres de menu sont contextuelles (la barre change si l'on passe de la gestion de projet à l'édition de schéma ou de description VHDL)*
- ⇒ File New design Une fenêtre d'édition de schéma s'ouvre, la barre de schéma est sur la droite
- ⇒ Placer un bloc hiérarchique comme suit



sur le schéma pour placer le bloc. L'agrandir, placer les entrées et sorties avec **place pin**

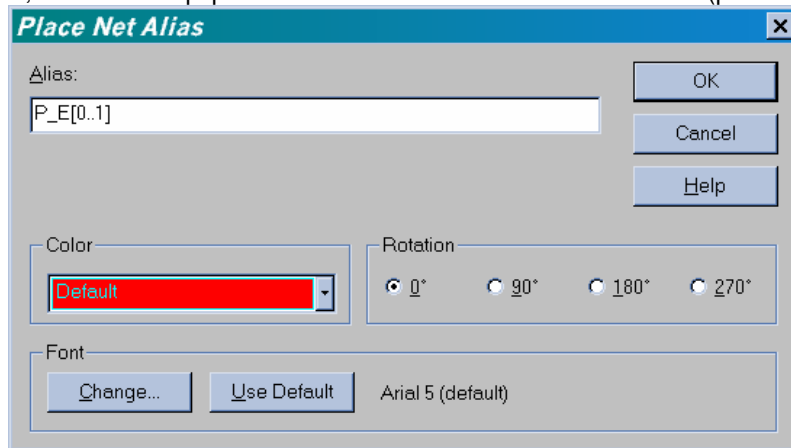


Entrées : nom **E[0..1]**, type **input** et with **bus**

Sorties : nom **S[0..3]** type **output** et with **bus**

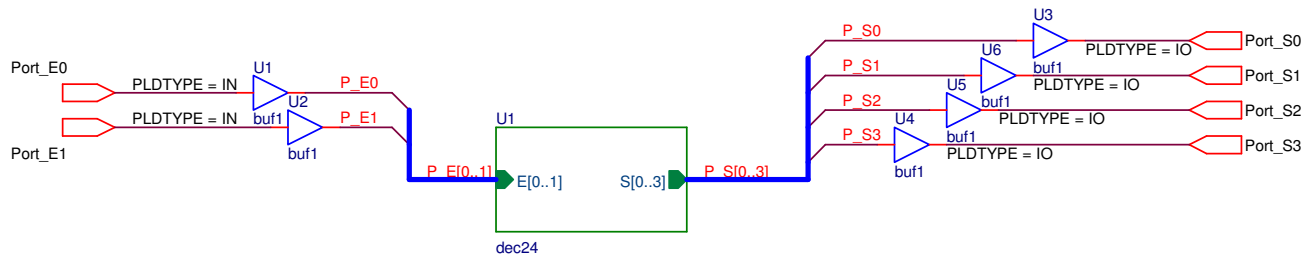
⇒ Ajouter les buffers (buf1), (ils ne sont pas indispensables ici),

- ✎ dans le cas contraire il faut : **Option** **compile setup** **output** **add IO**
- ➔ Ajouter les ports, les bus et équipotentielles et nommer les avec des alias (place net alias)



- sur un net fait apparaître ses propriétés.
- ➔ Pour les deux entrées et les quatre sorties. Filtrer les propriétés par **Express-simple PLD**, puis configurer **PLD TYPE** avec **IN** ou **IO**, activer la visualisation de ces paramètres

schéma obtenu



- ➔ Sélectionner U1 puis à gauche et sélectionner **descend hierarchy**
- ➔ Compléter le fichier VHDL comme suit :

```

-- VHDL Decodeur 2 vers 4
--
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;

ENTITY dec24 is
    PORT (
        S      : OUT STD_LOGIC_VECTOR (3 downto 0);
        E      : IN STD_LOGIC_VECTOR (1 downto 0));
END dec24;

ARCHITECTURE combinatoire OF dec24 IS
signal S0, S1, S2, S3 : STD_LOGIC ;
BEGIN

    S0 <= not(E(0)) and not (E(1)) ;
    S1 <= E(0) and not(E(1));
    S2 <= not(E(0)) and E(1);
    S3 <= E(0) and E(1) ;
    S <= S3 & S2 & S1 & S0 ;

END combinatoire;

```

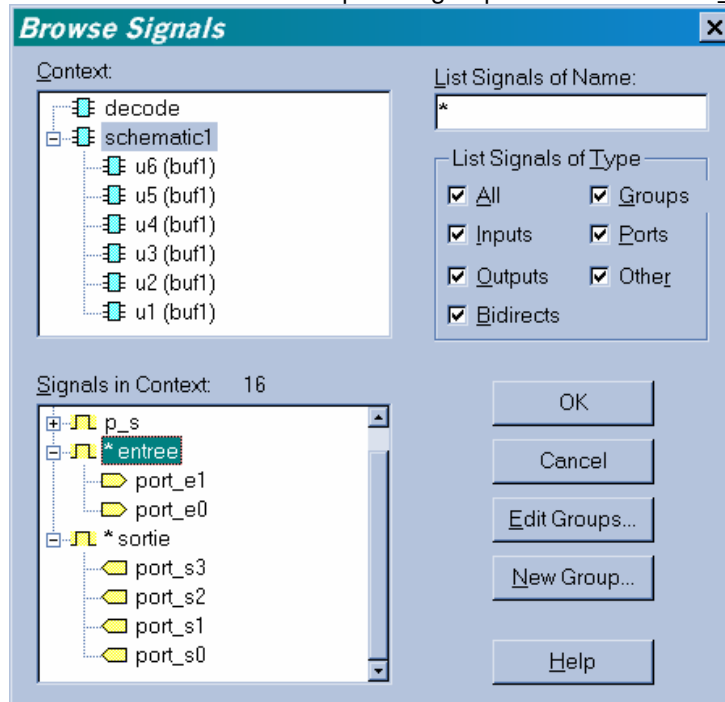
- ➔ Taper **alt-c** pour vérifier la syntaxe VHDL

2.2. Simulation fonctionnelle

➔ Sélectionner la fenêtre projet puis lancer **Simulate** en mode **in design**

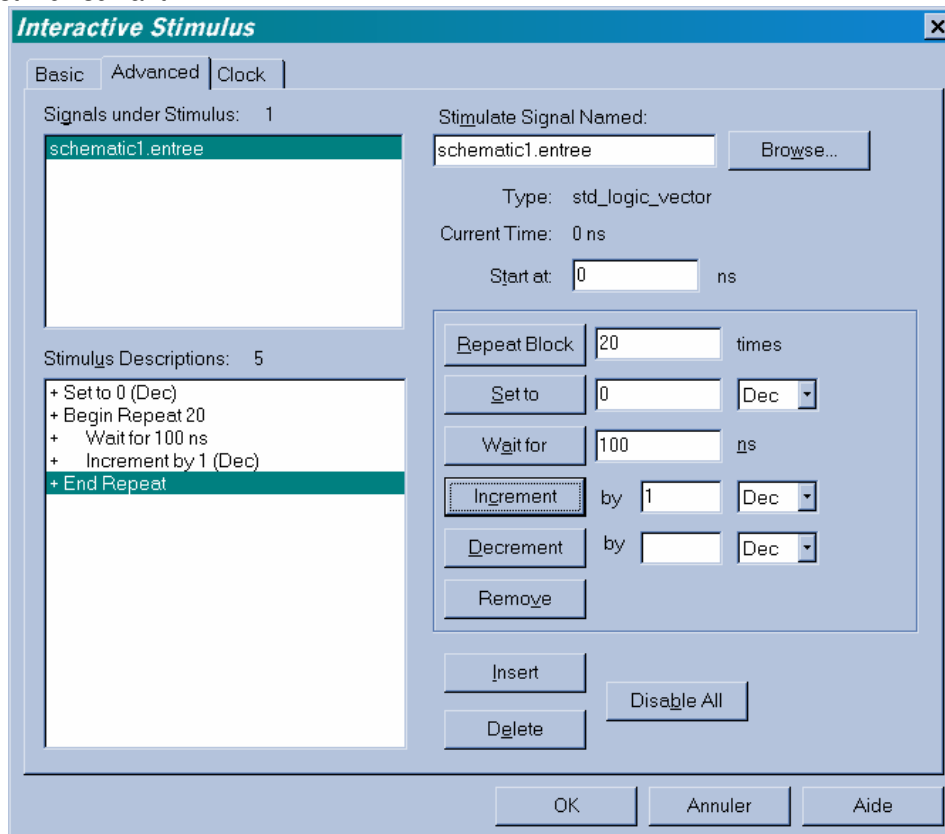
📁 **Stimulus** 📁 **New interactive**

📁 **Browse**, créer deux groupes de signaux (entrées et sorties). Sélectionner **Port_e0** et **Port_e1** puis **New Group**, le nommer **entree**. Faire de même pour le groupe **sortie** avec **Port_s0** à **Port_s3**.



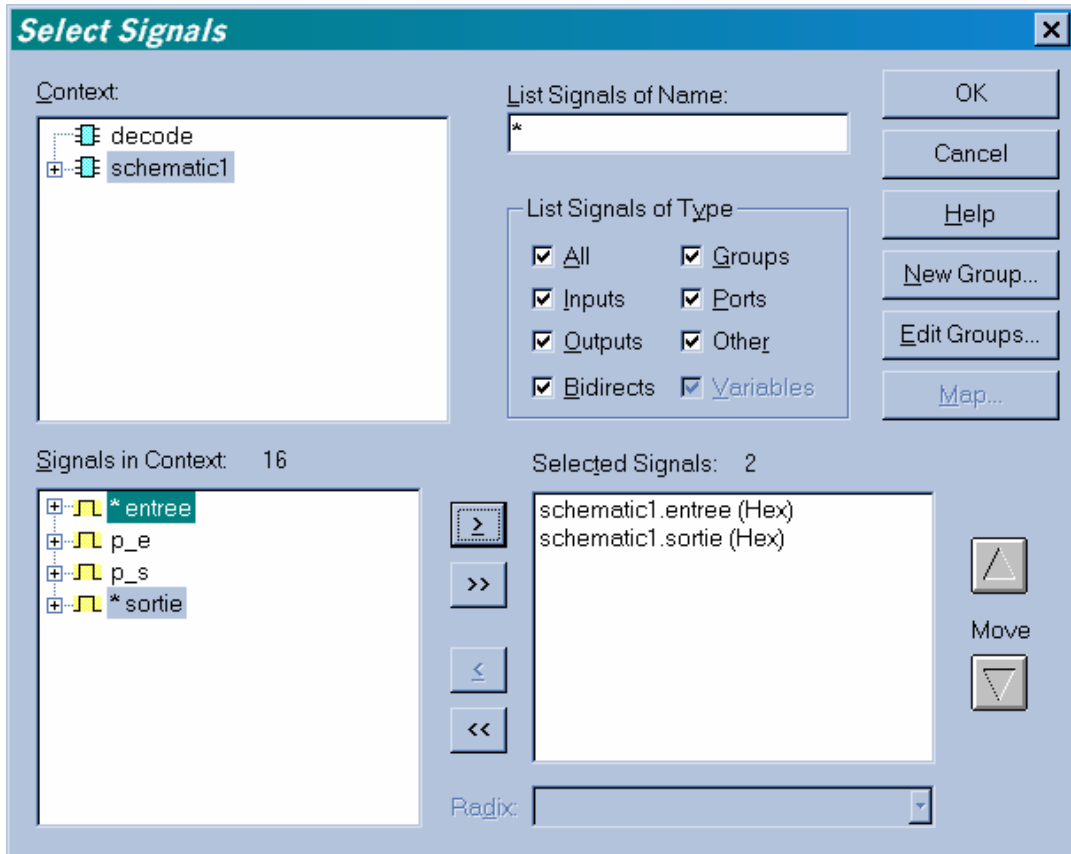
➔ Sélectionner le groupe entree puis **OK**

➔ créer le stimuli suivant

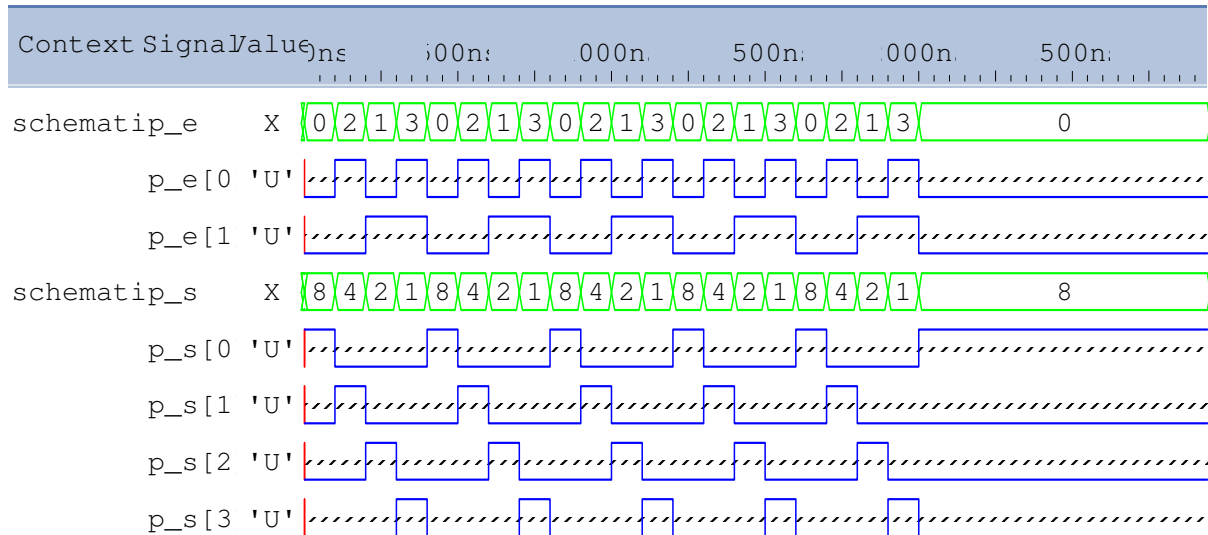




- Trace
- New wave windows
- Selectionner entrée et sortie

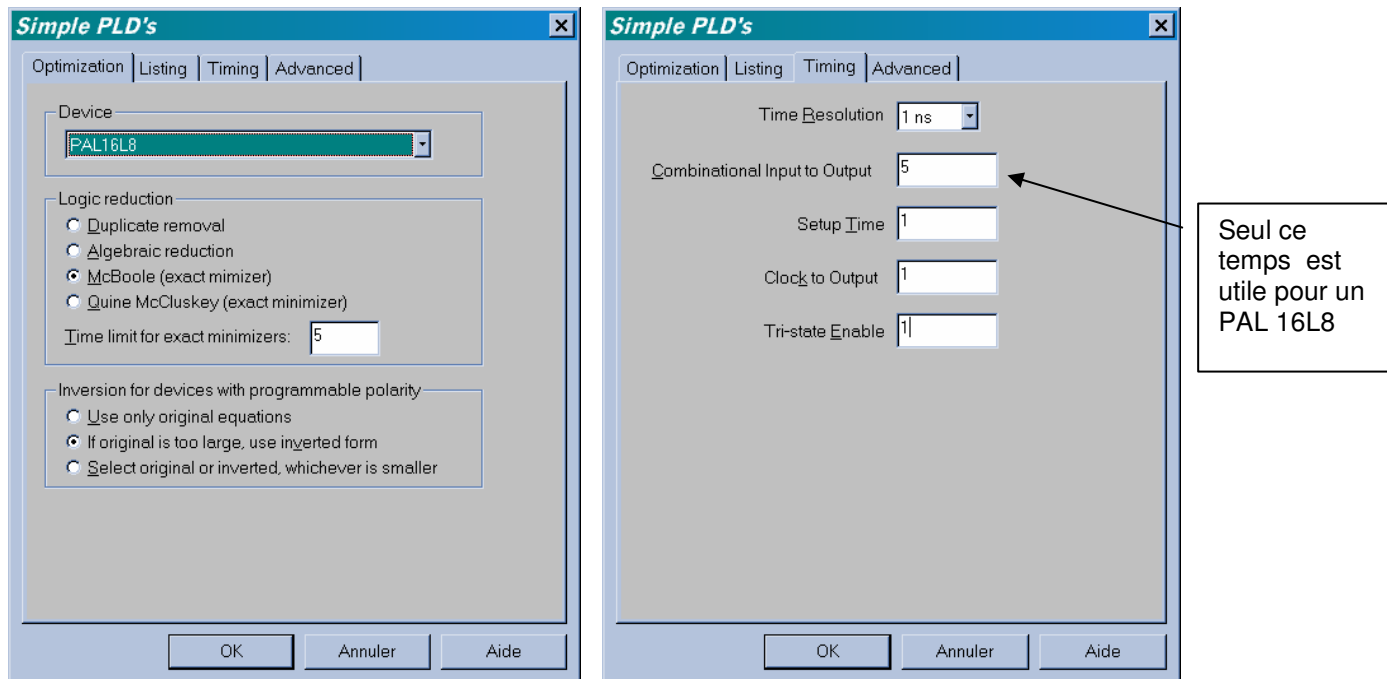


- Option
- Project Run Duration = 4000 ns
- RUN



2.3. Simulation temporelle

- ➔ Compiler le projet, **tools** **compile** *il ne doit y avoir aucune erreur*
- ➔ Sous Capture **Tools** **build**



La réduction McBoole combine la suppression des termes identiques et la réduction algébrique (X.0=0, X.X=X etc...)

- ➔ Sélectionner un PAL16L8, dans Timing entrer 5ns pour Combinational input to Output
Ne pas modifier les autres paramètres

Voici le fichier généré par EXPRESS pour la simulation temporelle

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

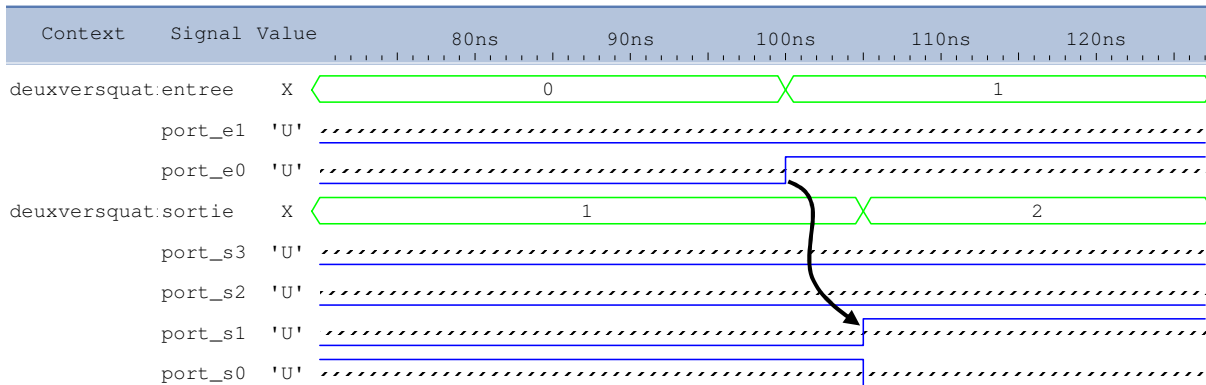
ENTITY decodeur IS
  GENERIC(
    tpd_a_y : time := 5 ns;
    tpd_clk_q : time := 2 ns;
    tpd_enable_y : time := 10 ns;
    tsetup_d_clk : time := 2 ns);
  PORT(
    port_e0 : IN std_logic; port_e1 : IN std_logic; port_s3 : OUT std_logic;
    port_s2 : OUT std_logic; port_s1 : OUT std_logic; port_s0 : OUT std_logic);
END decodeur;

ARCHITECTURE model OF decodeur IS
  SIGNAL VCC : std_logic := '1';
  SIGNAL GND : std_logic := '0';

BEGIN
  port_s0 <= NOT ((port_e1) OR (port_e0)) AFTER tpd_a_y;
  port_s3 <= NOT (( NOT port_e1) OR ( NOT port_e0)) AFTER tpd_a_y;
  port_s2 <= NOT (( NOT port_e1) OR (port_e0)) AFTER tpd_a_y;
  port_s1 <= NOT ((port_e1) OR ( NOT port_e0)) AFTER tpd_a_y;
END model;

```

➔ Refaire une simulation mais en mode **TIMED** et constater la prise en compte des temps de propagation.



2.4. Création du composant deuxversquatre

➔ Sous Capture tools generate parts

Generate Part

Netlist file:

Vendor file type:

Part name:

Part library:

Create new part
 Update pins on existing part in library.

Sort pins

Ascending order
 Descending order

Additional pins

Specify the number of additional pins on part
 Number of pins:

Implementation

Implementation type:

Implementation name:

Implementation file:

Le fichier deuxversquatre.LST décrit complètement le nouveau composant

OrCAD PLD FITTER x1.11 11/5/98 (Source file c:\orcadtrv\exol\Timed\deuxversquatre.jed)

RESOLVED EXPRESSIONS (Reduction 2)

Signal name	Row	Terms
port_s0'	9	port_e1
	10	port_e0
port_s3'	33	port_e1'
	34	port_e0'
port_s2'	25	port_e1'
	26	port_e0
port_s1'	17	port_e1
	18	port_e0'

SIGNAL ASSIGNMENT

Pin	Signal name	Column	Rows			Activity
			Beg	Avail	Used	
1.	port_e0	2	-	-	-	High
2.	port_e1	0	-	-	-	High
3.	-	4	-	-	-	
4.	-	8	-	-	-	
5.	-	12	-	-	-	
6.	-	16	-	-	-	
7.	-	20	-	-	-	
8.	-	24	-	-	-	
9.	-	28	-	-	-	
11.	-	30	-	-	-	
12.	-	-	56	8	0	(Three-state)
13.	-	26	48	8	0	(Three-state)
14.	-	22	40	8	0	(Three-state)
15.	port_s3	18	32	8	2	High (Three-state)
16.	port_s2	14	24	8	2	High (Three-state)
17.	port_s1	10	16	8	2	High (Three-state)
18.	port_s0	6	8	8	2	High (Three-state)
19.	-	-	0	8	0	(Three-state)

			64	8	(13%)	

Brochage du nouveau composant

NOTE (PLD0200) No fatal errors found in input file.
NOTE (PLD0201) No warnings.

OrCAD PLD EXPRESS

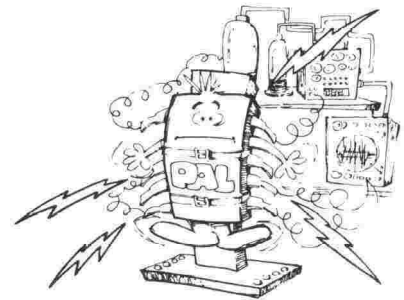
Type: PAL16L8

*

QP20* QF2048* QV1024*

F0*

```
L0256 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0288 01 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0320 11 01 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0512 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0544 01 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0576 11 10 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0768 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0800 10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0832 11 01 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L1024 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L1056 10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L1088 11 10 11 11 11 11 11 11 11 11 11 11 11 11 11 *
C2FB2*
```



Le fichier JEDEC destiné au programmeur de PAL

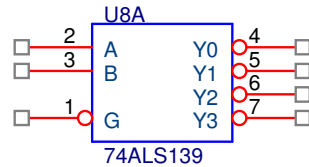
NOTE (PLD0202) 2/21/0 1:45 am (Monday)
NOTE (PLD0204) Elapsed time 1 second

Exercices :

1. Modifier le programme source VHDL afin de réaliser le décodeur (simulation et synthèse) à partir des descriptions suivantes
 - a) <= WHEN ELSE
 - b) WITH SELECT

2. A partir des exemples précédents, réaliser un composant MUX14 qui sera un démultiplexeur 1 vers 4 type LS139

A	B	Y0	Y1	Y2	Y3
0	0	G	1	1	1
0	1	1	G	1	1
1	0	1	1	G	1
1	1	1	1	1	G

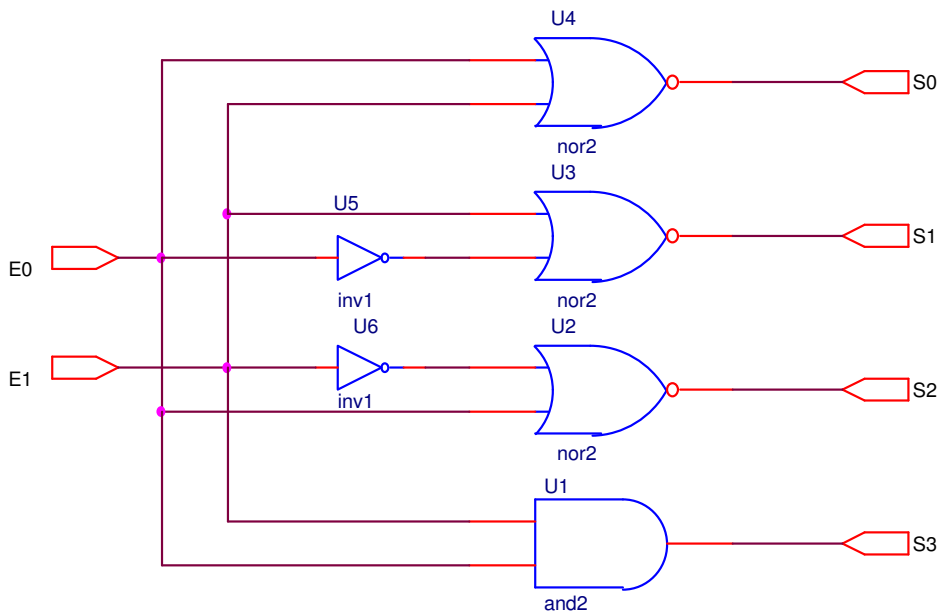


Le simuler, le synthétiser dans un PAL16L8

Express permet également une saisie par schéma :

3. Créer un nouveau projet appelé "decodeschema", procéder comme pour une description VHDL, mais lors de la création du bloc on indiquera comme "implémentation type" : SCHEMATIC VIEW

Le menu "descend hierarchy" ouvrira un nouveau schéma "hiérarchique", saisir le schéma suivant : (les composants se trouvent dans la librairie SPLD)



Réaliser comme précédemment les simulation fonctionnelle et TIMED. On remarquera les "glitches" de la simulation fonctionnelle.

3. Eléments du langage

3.1. Signaux, constantes, variables

- ❑ Les signaux représentent des équipotentielles, ils existent physiquement. Ils doivent être déclarés AVANT le mot BEGIN. Un signal peut être connecté (toujours définitivement) à un autre de même type par `signal1 <= signal2`
- ❑ Les variables n'existent pas physiquement. Une variable a une valeur temporaire affectée par l'opération `var1:=var2`. L'attribution d'une variable à un signal n'est pas toujours synthétisable et demande une conversion de type. Les variables peuvent être utilisées dans les calculs algébriques au contraire des signaux, pour lesquels ne peuvent être utilisés que des opérateurs booléens (sauf en cas d'utilisation d'une bibliothèque)
- ❑ Les constantes sont déclarées comme suit : **constant test : bit_vector := "10100101";**

3.2. Les types d'objets

Ces types prédéfinis de base sont connus dans le Package Standard VHDL'93.

Type	Description
boolean	False ou True
bit	('0','1')
character	Table ASCII limitée
severity_level	(Note, warning, error, failure)
integer	-2^{31} à 2^{31} (-2 147 483 647 à 2 147 483 647)
real	$-1.0e^{38}$ à $1.0e^{38}$
time	-2^{31} à 2^{31} (-2 147 483 647 à 2 147 483 647)
delay_length	0 à time'high (2 147 483 647) (sous-type de time)
natural	0 à integer'high (2^{31}) (sous-type de integer)
positive	1 à integer'high (2^{31}) (sous-type de integer)
String	Array of character (tableau de caractères)
bit_vector	Array of bit (tableau de bits)

Le package `std_logic_1164` inclus de nombreux autres types

```

TYPE std_ulogic IS ( 'U', -- Uninitialized
                    'X', -- Forcing Unknown
                    '0', -- Forcing 0
                    '1', -- Forcing 1
                    'Z', -- High Impedance
                    'W', -- Weak Unknown
                    'L', -- Weak 0
                    'H', -- Weak 1
                    '-' -- Don't care
                );
TYPE std_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_ulogic;

```

Le package `numeric_std` permet de nombreuses opération logiques ET arithmétiques sur les type UNSIGNED et SIGNED

```

type UNSIGNED is array (NATURAL range <>) of STD_LOGIC;
type SIGNED is array (NATURAL range <>) of STD_LOGIC;

```

3.3. Les opérateurs

Classe	Symbole	Fonction	Definit pour
<u>Opérateurs divers</u>	not	complément	bit, booléen
Classe de plus haute priorité	**	exponentiel	entier, réel
	abs	valeur absolue	numérique
	*	multiplication	numérique
<u>Opérateurs multiplicatifs</u>	/	division	
	mod rem	modulo reste	
<u>Signe</u> (unaire)	+	positif	numérique
	-	négatif	
<u>Opérateurs additifs</u> (binaire)	+	addition	numérique 1 dimension
	-	soustraction	
	&	concaténation	
	=	égal	tous les types retournent un booléen
<u>Opérateurs relationnels</u>	/=	différent	
	<	inférieur	Scalaire retourne un booléen
	>	supérieur	
	<=	inférieur ou égal	
	>=	supérieur ou égal	
<u>Opérateurs logiques</u> (binaire)	and	et logique	bit booléen vecteur
Classe de plus faible priorité	or	ou logique	
	nand	et non logique	
	nor	ou non logique	
	xor	ou exclusif	

3.4. Les attributs

Exemple : `If clk'event and clk=1 then ...` (vrai si clk vient de monter)

Attributs	Définitions - informations de retour
'high	Placé près d'un nom de tableau, il retourne la valeur du rang le plus haut (la valeur de retour est de type entier).
'low	Placé près d'un nom de tableau, il retourne la valeur du rang le plus bas (la valeur de retour est de type entier).
'left	Placé près d'un nom de vecteur ou tableau, il retourne la valeur du rang le plus à gauche (la valeur de retour est de type entier).
'right	Placé près d'un nom de vecteur ou tableau, il retourne la valeur du rang le plus à droite (la valeur de retour est de type entier).
'range	Placé près d'un signal, il retourne la valeur de l'intervalle spécifié par l'instruction range lors de la déclaration du signal ou du type utilisé.
'reverse_range	Placé près d'un signal, il retourne la valeur de l'intervalle inverse spécifié par l'instruction range lors de la déclaration du signal ou du type utilisé.
'length	Retourne $X'high - X'low + 1$ (sous la forme d'un entier).
'event	Vrai ou faux si le signal auquel il est associé vient de subir un changement de valeur.
'active	Vrai ou faux si le signal auquel il est associé vient d'être affecté.
'last_value	Retourne la valeur précédente du signal auquel il est associé.
'last_event ou 'last-active	Retournent des informations de temps concernant la date d'affectation ou de transition des signaux auxquels ils sont affectés
'stable(T)	Vrai ou faux si le signal auquel il est associé n'est pas modifié pendant la durée T.
'quiet	Vrai ou faux si le signal auquel il est associé n'est pas affecté pendant la durée T.
'transaction	Retourne une information de type bit qui change d'état lorsque le signal auquel il est associé est affecté.

(tableau d'après T.BLOTIN : le langage de description VHDL)



4. La description séquentielle

A lire : Cours d'Y.Aubril et d'A.Mouflet

- ❑ *La description séquentielle*
- ❑ *La description comportementale*

- ❑ Les process sont **séquentiels**, ils possèdent une **liste de sensibilité** les activant, **les signaux ne sont effectivement modifiés qu'à la fin du process.**
- ❑ *L'instruction **wait** peut remplacer la liste de sensibilité*
- ❑ **La synthèse d'un process n'est pas forcément séquentielle**

Décodeur par process avec *case when others*

Case when est la version process de with select

```
-- VHDL created by OrCAD Express
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Decode is
    PORT (
        S      : OUT STD_LOGIC_VECTOR (3 downto 0);
        E      : IN  STD_LOGIC_VECTOR (1 downto 0));
END Decode;

ARCHITECTURE combinatoire OF Decode IS
    process (E)
        begin
            case E is
                when "00"=> S<="0001";
                when "01"=> S<="0010";
                when "10"=> S<="0100";
                when "11"=> S<="1000";
                when others=> S<="0000";
            end case;
        end process;
```

En logique séquentielle il est obligatoire de décrire toutes les possibilités.

L'utilisation de OTHERS est fortement recommandée, certains compilateurs la rendent obligatoire

Décodeur par process avec if then elsif else end if

```
--VHDL created by OrCAD Express
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Decode is
    PORT (
        S    : OUT STD_LOGIC_VECTOR (3 downto 0);
        E    : IN STD_LOGIC_VECTOR (1 downto 0));
    END Decode;

ARCHITECTURE combinatoire OF Decode IS
process (E)
begin
    if (E = "00") then S <= "0001" ;
    elsif (E = "01") then S <= "0010";
        elsif (E = "10") then S <= "0100" ;
        else S <= "1000" ;
    end if ;
end process ;
```

autre version avec if utilisant l'indexage de l'entrée E

```
-- VHDL created by OrCAD Express
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Decode is
    PORT (
        S    : OUT STD_LOGIC_VECTOR (3 downto 0);
        E: IN STD_LOGIC_VECTOR (1 downto 0));
    END Decode;

ARCHITECTURE combinatoire OF Decode IS
process (E)
begin
    if (E(1)= '0') then
        if (E(0)= '0') then S <= "0001" ;  else S <= "0010" ;
        end if ;
    else
        if (E (0)= '0') then S <= "0100" ;  else S <= "1000" ;
        end if ;
    end if;
end process ;
```

Exercices :

- a) Simuler les descriptions séquentielles proposées du décodeur

5. VHDL et logique séquentielle

5.1. Bascules et verrou

Verrou 1

```
process (LATCH_EN, DATA_IN)
begin
  if (LATCH_EN = '1') then QOUT <= DATA_IN ; endif ;
end process ;
```

Exemples issus du cours VHDL : Stage
VHDL 1998 LT JB DE BAUDRE

Faux verrou

```
Process (LATCH_EN, DATA_IN)
begin
  if (LATCH_EN = '1') then QOUT <= DATA_IN ;
  else QOUT <= '0') ; endif ;
end process ;
```

Bascule D1

```
Process
begin
  wait until (CLOCK'EVENT and CLOCK = '1') ;
  QOUT <= DATA_IN ;
end process ;
```

Bascule D2

```
Process(CLOCK, DATA_IN)
begin
  if (CLOCK'EVENT and CLOCK = '1') then QOUT <= DATA_IN ; endif ;
end process ;
```

Bascule D3

```
Process(RESET,CLOCK, DATA_IN)
begin
  if (RESET = '1') then QOUT <= '0') ; -- reset asynchrone
  elsif (CLOCK'EVENT and CLOCK = '1') then QOUT <= DATA_IN ; -- synchrone
  endif ;
end process ;
```

Bascule D4

Irréalizable car il est « difficile » de synthétiser une bascule dont le reset se ferait sur des « non fronts montants de CLOCK » !

```
Process(RESET,CLOCK, DATA_IN)
begin
  if (CLOCK'EVENT and CLOCK = '1') then QOUT <= DATA_IN ; -- synchrone
  elsif (RESET = '1') then QOUT <= '0') ; -- reset asynchrone
  endif ;
end process ;
```

Bascule D5

DATA_IN n'est pas utile dans la Sensitivity List

```
Process(RESET,CLOCK)
begin
  if (RESET = '1') then QOUT <= '0') ; -- reset asynchrone
  elsif (CLOCK'EVENT and CLOCK = '1') then QOUT <= DATA_IN ; -- synchrone
  endif ;
end process ;
```

5.2. les compteurs

Soit le compteur 8 bits :

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity compteur is
  port(reset, ck : in std_logic;
        compte : out std_logic_vector(7 downto 0));
end;

architecture comportementale of compteur is
  signal compte_n: unsigned(7 downto 0);
begin
  process (reset, ck, compte_n)
  begin
    if reset='1' then
      compte_n <= x"00";
    elsif (ck='1' and ck'event) then
      compte_n <= compte_n + "1";
    end if;
  end process;
  compte <= std_logic_vector(compte_n);
end;
```

La fonction addition se trouve dans cette bibliothèque

Le type UNSIGNED est défini dans numeric_std, il est représenté par un vecteur

Notation Hexadécimale

Opération possible uniquement grâce à numeric_std et au type unsigned

L'assignation est possible après conversion car les types sont de même dimension

- b) Compléter le compteur précédent par une entrée de validation
- c) Compléter le compteur précédent par une entrée de sélection de sens
- d) Compléter le compteur précédent par une entrée de chargement parallèle



Le lecteur pourra s'aider des exemples pages suivantes



```
-- MAX+plus II VHDL Example
-- Copyright (c) 1994 Altera Corporation
ENTITY counters IS
  PORT
  (
    d          : IN  INTEGER RANGE 0 TO 255;
    clk        : IN  BIT;
    clear      : IN  BIT;
    ld         : IN  BIT;
    enable     : IN  BIT;
    up_down    : IN  BIT;
    qa         : OUT  INTEGER RANGE 0 TO 255;
    qb         : OUT  INTEGER RANGE 0 TO 255;
    qc         : OUT  INTEGER RANGE 0 TO 255;
    qd         : OUT  INTEGER RANGE 0 TO 255;
    qm         : OUT  INTEGER RANGE 0 TO 255;
    qn         : OUT  INTEGER RANGE 0 TO 255 );
END counters;
```

Entête de tous
les exemples

```
ARCHITECTURE a OF counters IS
  BEGIN
```

```
-- An enable counter
PROCESS (clk)
  VARIABLE cnt          : INTEGER RANGE 0 TO 255;
  BEGIN
    IF (clk'EVENT AND clk = '1') THEN
      IF enable = '1' THEN
        cnt := cnt + 1;
      END IF;
    END IF;
    qa  <= cnt;
  END PROCESS;
```

```
-- A synchronous load counter
PROCESS (clk)
  VARIABLE cnt          : INTEGER RANGE 0 TO 255;
  BEGIN
    IF (clk'EVENT AND clk = '1') THEN
      IF ld = '0' THEN
        cnt := d;
      ELSE
        cnt := cnt + 1;
      END IF;
    END IF;
    qb  <= cnt;
  END PROCESS;
```

```
-- A synchronous clear counter
PROCESS (clk)
  VARIABLE cnt          : INTEGER RANGE 0 TO 255;
  BEGIN
    IF (clk'EVENT AND clk = '1') THEN
      IF clear = '0' THEN
        cnt := 0;
      ELSE
        cnt := cnt + 1;
      END IF;
    END IF;
    qc  <= cnt;
  END PROCESS;
```

```
-- An up/down counter
PROCESS (clk)
    VARIABLE cnt : INTEGER RANGE 0 TO 255;
    VARIABLE direction : INTEGER;
BEGIN
    IF (up_down = '1') THEN
        direction := 1;
    ELSE
        direction := -1;
    END IF;

    IF (clk'EVENT AND clk = '1') THEN
        cnt := cnt + direction;
    END IF;
    qd <= cnt;
END PROCESS;
```

```
-- A synchronous clear enable up/down counter
PROCESS (clk)
    VARIABLE cnt : INTEGER RANGE 0 TO 255;
    VARIABLE direction : INTEGER;
BEGIN
    IF (up_down = '1') THEN
        direction := 1;
    ELSE
        direction := -1;
    END IF;

    IF (clk'EVENT AND clk = '1') THEN
        IF clear = '0' THEN
            cnt := 0;
        ELSE
            IF enable = '1' THEN
                cnt := cnt + direction;
            END IF;
        END IF;
    END IF;
    qm <= cnt;
END PROCESS;
```

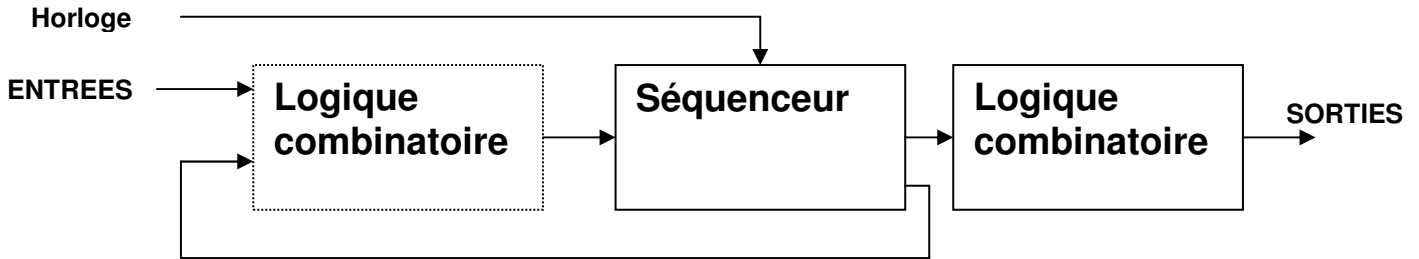
```
-- A modulus 200 up counter
PROCESS (clk)
    VARIABLE cnt : INTEGER RANGE 0 TO 255;
    CONSTANT modulus : INTEGER := 200;
BEGIN
    IF (clk'EVENT AND clk = '1') THEN
        IF cnt = modulus THEN cnt := 0;
        ELSE cnt := cnt + 1;
        END IF;
    END IF;
    qn <= cnt;
END PROCESS;
END a;
```

6. Les machines d'état

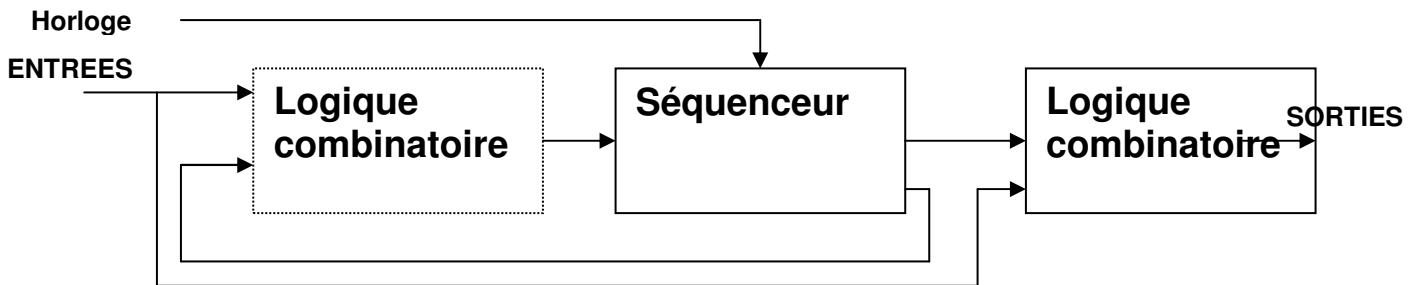
Les machines d'état permettent de décrire des processus séquentiels.

Il existe deux types de description :

MOORE



MEARLY



Chaque bloc est en décrit dans un process.

Exemple d'une structure de machine d'états "simple"

```

Library ieee;
Use ieee.std_logic_1164.all;
ENTITY exemple is
    PORT (.....);
END codami;
ARCHITECTURE behavior OF exemple IS
type etat4 is (un,deux,trois,quatre); -- autant qu'il en faut
signal present,suivant ;
BEGIN
    process(clk,RAZ)
    BEGIN
        if RAZ='1' then etat <=un; elsif (clk='1' and clk'event) then present <=suivant;; end if;
    end process;

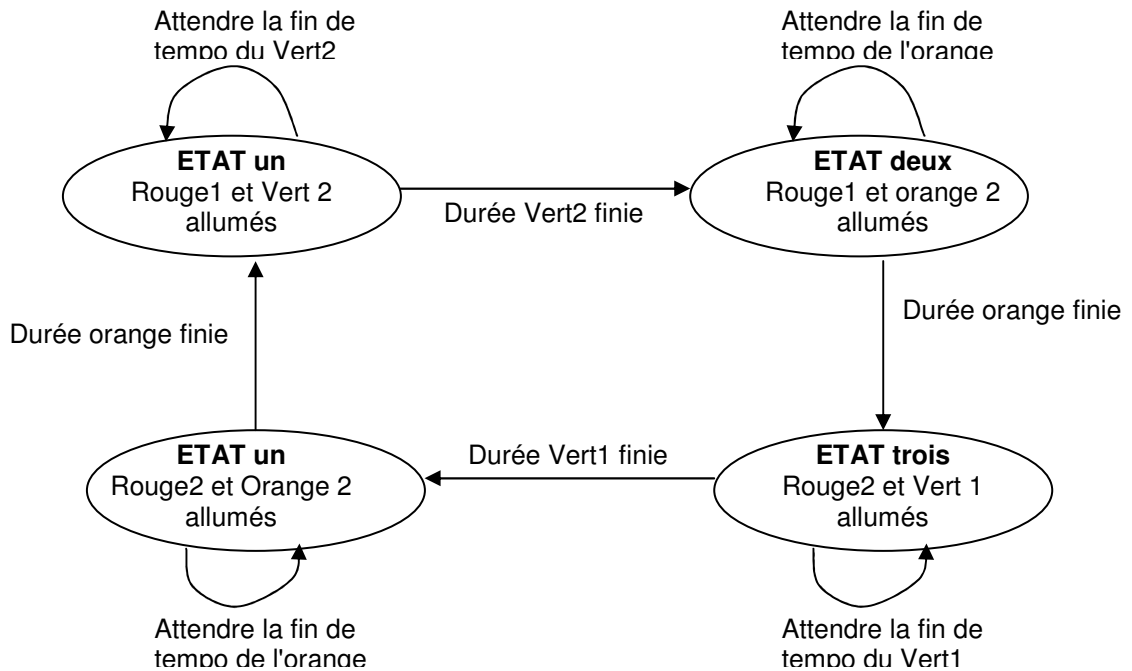
    process (present)
    begin
        case present is
            when un => .....
            when deux => .....
            when trois => .....
            when quatre => .....
        end case;
    end process;
END behavior;
    
```

On crée un nouveau type, qui sera en fait converti ici en en un BUS 2 bits

Deux bus 2 bits

Description combinatoire de l'état des sorties pour chacun des états

Exemple : feux tricolores



```

Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
ENTITY feux IS
    PORT (
        R1    : OUT STD_LOGIC;    -- vert , rouge .... Pour les deux feux des deux routes
        R2    : OUT STD_LOGIC;
        O1    : OUT STD_LOGIC;
        O2    : OUT STD_LOGIC;
        V1    : OUT STD_LOGIC;
        V2    : OUT STD_LOGIC;
        CLK   : IN  STD_LOGIC;
        RAZ   : IN  STD_LOGIC);
END feux;

ARCHITECTURE behavior OF feux IS

    type etat IS (un,deux,trois,quatre);    -- quatres états créés par deux bascules
    signal actuel, futur : etat;           -- deux signaux de type état

    signal compte : integer range 0 to 255; -- compteur d'impulsions d'horloge
    constant DV1: integer:=20;            -- durée du vert 1
    constant DV2: integer:=40;            -- durée du vert 2
    constant DO : integer:=5;             -- durée de l'orange

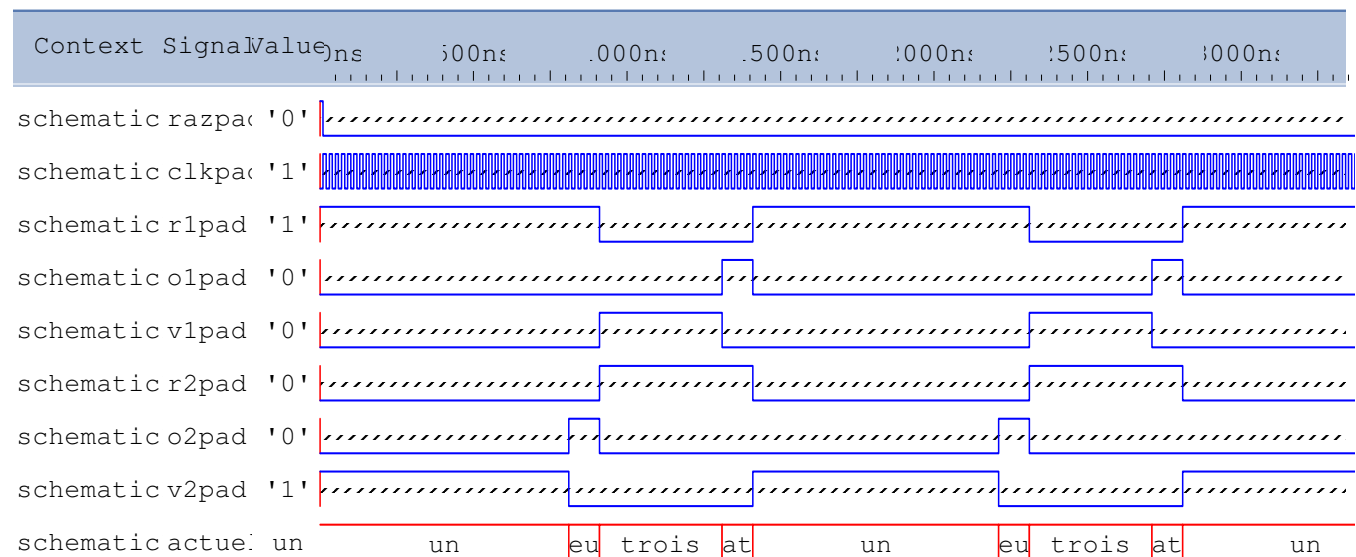
BEGIN
    process(clk,RAZ)
    BEGIN
        if RAZ ='1' then actuel<=un;
        elsif clk'event and clk='1' then actuel<=futur;
        end if;
    end process;
END behavior;
  
```

```

END PROCESS;

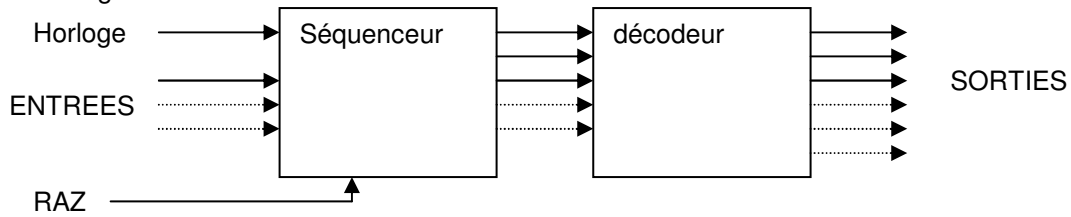
process (actuel,clk) -- machine d'état
begin
  case actuel is
    when un =>
      R1<='1';O1<='0';V1<='0';R2<='0';O2<='0';V2<='1';
      if compte<DV2 then
        if clk'event and clk='1' then compte<=compte+1; end if;
        futur <= un;
      else compte<=0; futur <= deux;
      end if;
    when deux =>
      R1<='1';O1<='0';V1<='0';R2<='0';O2<='1';V2<='0';
      if compte<DO then
        if clk'event and clk='1' then compte<=compte+1; end if;
        futur <= deux;
      else compte<=0; futur <= trois;
      end if;
    when trois =>
      R1<='0';O1<='0';V1<='1';R2<='1';O2<='0';V2<='0';
      if compte<DV1 then
        if clk'event and clk='1' then compte<=compte+1; end if;
        futur <= trois;
      else compte<=0; futur <= quatre;
      end if;
    when quatre =>
      R1<='0';O1<='1';V1<='0';R2<='1';O2<='0';V2<='0';
      if compte<DO then
        if clk'event and clk='1' then compte<=compte+1;end if;
        futur <= quatre;
      else compte<=0; futur <= un;
      end if;
    when others => futur <= un;
  end case;
end process;
END behavior;

```



6.1. Machine d'état à sorties synchrones (Moore)

On écrit ici un séquenceur, dont les sorties commandent une logique asynchrone. On ne fait appel ici qu'à un signal d'état d'où économie de bascules

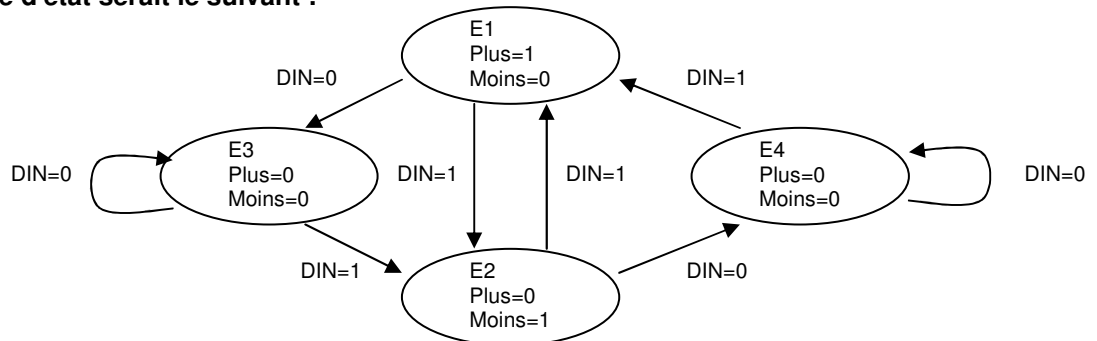


Exemple codeur AMI

Ce type de code est utilisé dans les transmissions téléphonique à longue distance.

- Un zéro logique correspond à une tension nulle
- Un "un" logique correspond alternativement à une tension positive puis à une tension négative

Le diagramme d'état serait le suivant :



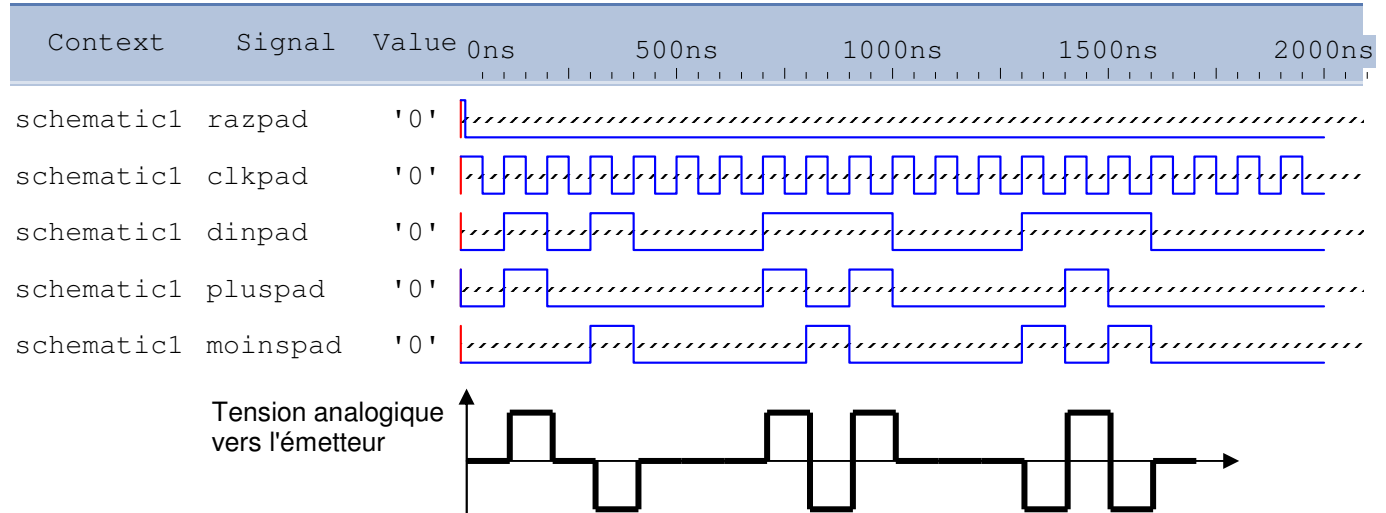
La description VHDL peut ressembler à celle ci

```

Library ieee;
Use ieee.std_logic_1164.all;
ENTITY codami is
    PORT (
        CLK : IN STD_LOGIC; RAZ : IN STD_LOGIC; DIN : IN STD_LOGIC;
        PLUS : OUT STD_LOGIC; MOINS : OUT STD_LOGIC);
END codami;
ARCHITECTURE behavior OF codami IS
type etat4 is (e1,e2,e3,e4);
signal etat : etat4;
BEGIN process(clk,RAZ)
    BEGIN
        if RAZ='1' then etat <=e4; elsif (clk='1' and clk'event) then
            case etat is
                when e1=> ..... } Décrire quel doit
                when e2=> ..... } être l'état suivant
                when e3=> ..... }
                when e4=> ..... }
            end case;
        end if;
    end process;
    process (etat)
    begin
        case etat is
            when e1 => .....
            when e2 => .....
            when e3 => .....
            when e4 => .....
        end case;
    end process;
END behavior;

```

Réaliser le codeur AMI afin d'obtenir le résultat suivant :



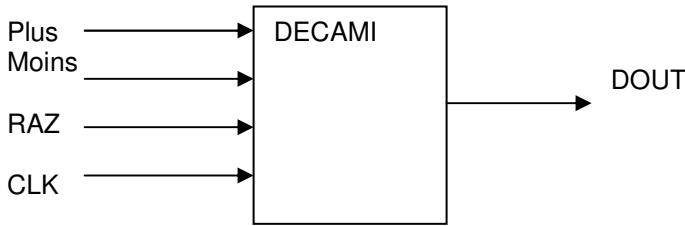
La synthèse n'est pas toujours facile à réaliser, elle dépend de l'écriture du fichier VHDL et d'autres paramètres comme le type de codage binaire de la machine d'état. Pour agir sur celui ci, avant la compilation :

Option Compile Setup

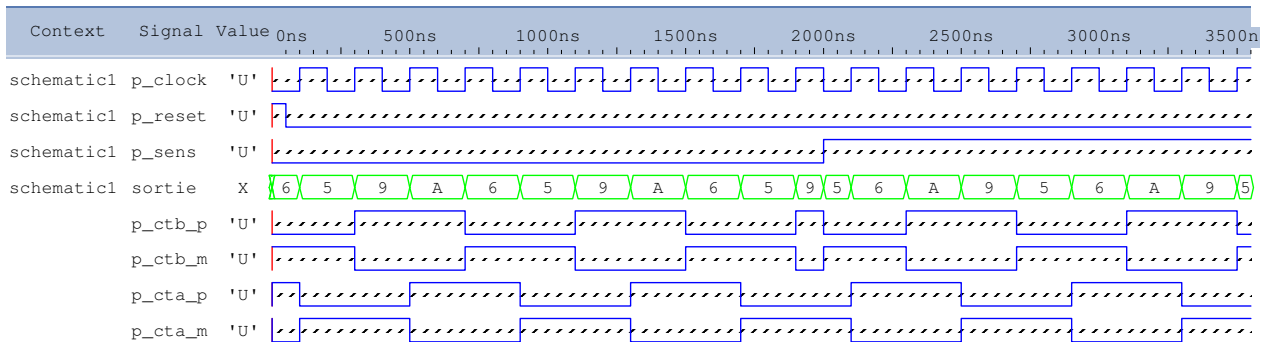


Exercices:

1) proposer une solution pour le décodeur AMI, puis reboucler sur le schéma les signaux plus et moins des deux description et vérifier la bonne réception du signal émis.



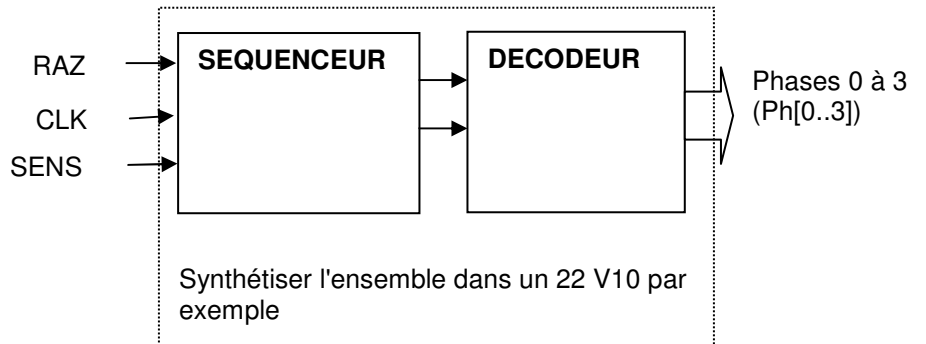
2) réaliser une commande de moteur pas à pas comme suit :



- a) sans le signal SENS
- b) avec le signal SENS

Attention :

-Donner l'attribut CLOCK au signal d'horloge, l'attribut "IO" aux sorties et l'attribut "IN" à l'entrée RESET
 -Construire le fichier VHDL avec deux process : un séquenceur et un décodeur. (On introduira un signal intermédiaire)



7. Présentation de MAX+II et du KIT UP1 d'ALTERA

Voir doc Le KIT ALTERA : fichier ALTERA univ.pdf

7.1. La série MAX 7000

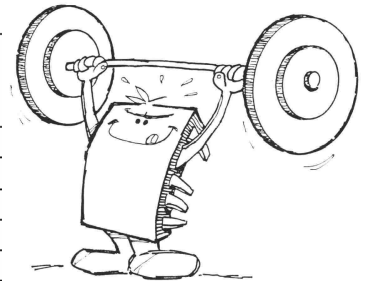
Le composant présent sur le KIT UP1 est EPM7128S

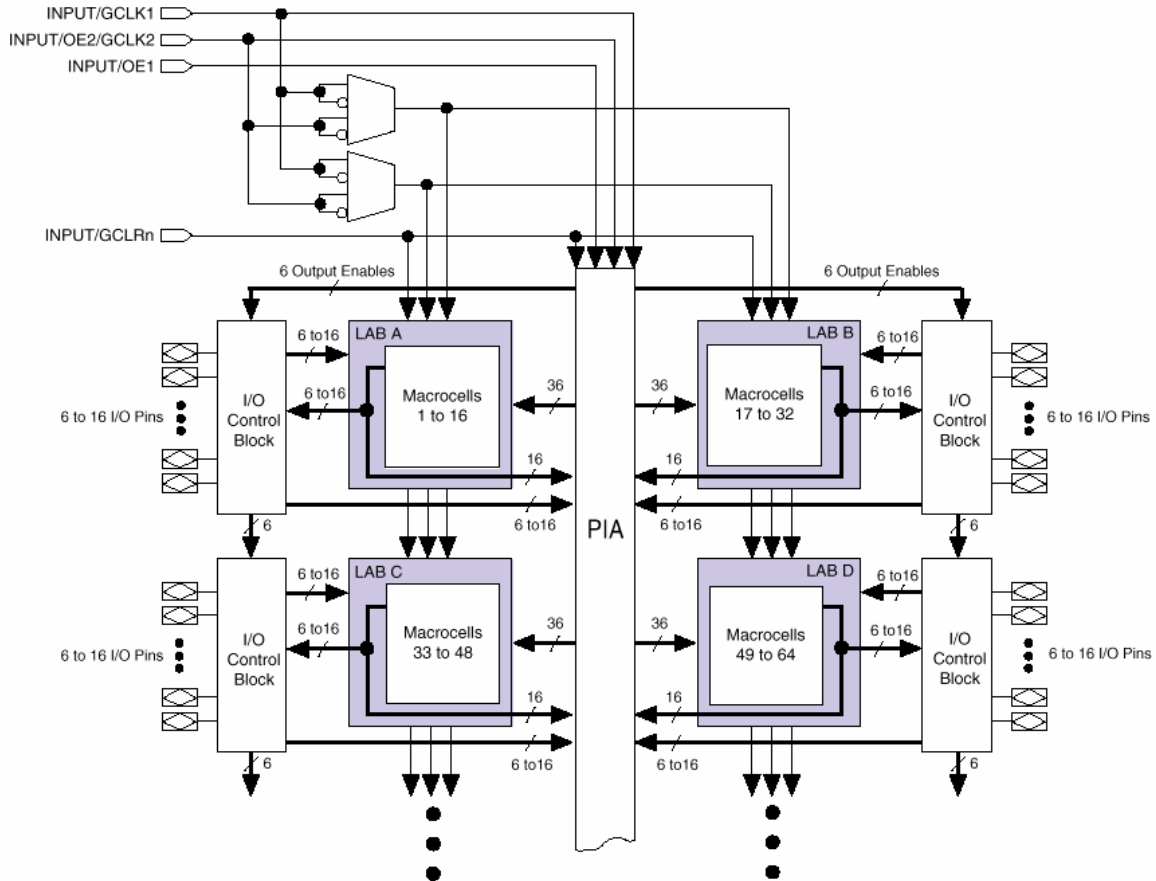
Hologe quartz
27,175 MHz
sur le KIT UP1

7.2. Structure interne du MAX7000

La structure est principalement organisée autour de 8 Logic Array Bloc (LAB) reliés d'une part aux E/S et d'autre part à une fonction d'interconnexion interne : Programmable Interconnect Array (PIA)

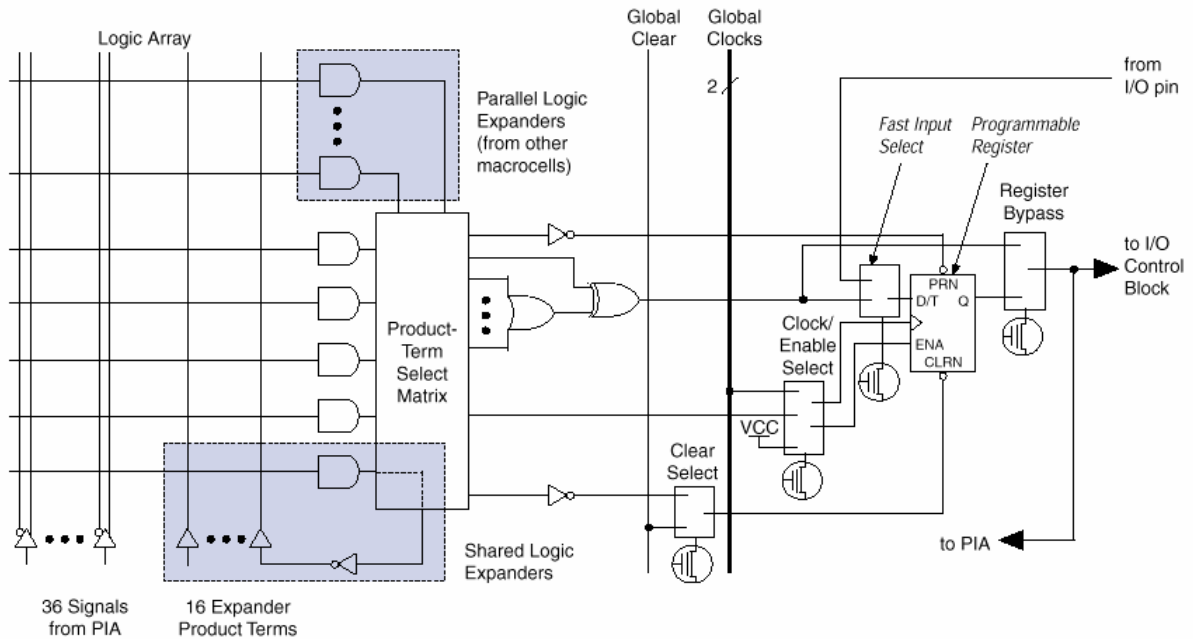
Dedicated Pin	84-Pin PLCC	LAB	MC	84-Pin PLCC	LAB	MC	84-Pin PLCC	LAB	MC	84-Pin PLCC				
INPUT/GCLK1	83	A	1	-	C	33	-	E	71	-	103	-		
INPUT/GCLRn	1		2	-		34	-		72	48	104	67		
INPUT/OE1	84		3	12		35	31		73	49	105	68		
INPUT/OE2/GCLK2	2		4	-		36	-		74	-	106	-		
TDI (3)	14		5	11		37	30		75	50	107	69		
TMS (3)	23		6	10		38	29		76	-	108	-		
TCK (3)	62		7	-		39	-		77	51	109	70		
TDO (3)	71		8	9		40	28		78	-	110	-		
GNDINT	42, 82		9	-		41	-		79	-	111	-		
GNDIO	7, 19, 32, 47, 59, 72		10	-		42	-		80	52	112	71 (3)		
VCCINT (5.0 V only)	3, 43		11	8		43	27		F	81	-	H	113	-
VCCIO (3.3 V or 5.0 V)	13, 26, 38, 53, 66, 78		12	-		44	-			82	-		114	-
No Connect (N.C.)	-		13	6		45	25			83	54		115	73
			14	5		46	24			84	-		116	-
			15	-		47	-			85	55		117	74
			16	4		48	23 (3)			86	56		118	75
		B	17	22	D	49	41	87		-	119		-	
			18	-		50	-	88		57	120		76	
			19	21		51	40	89		-	121		-	
			20	-		52	-	90		-	122		-	
			21	20		53	39	91		58	123		77	
			22	-		54	-	92		-	124		-	
			23	-		55	-	93		60	125		79	
			24	18		56	37	94		61	126		80	
			25	17		57	36	95		-	127		-	
			26	-		58	-	96		62 (3)	128		81	
			27	16		59	35							
			28	-		60	-							
			29	15		61	34							
			30	-		62	-							
			31	-		63	-							
			32	14 (3)		64	33							





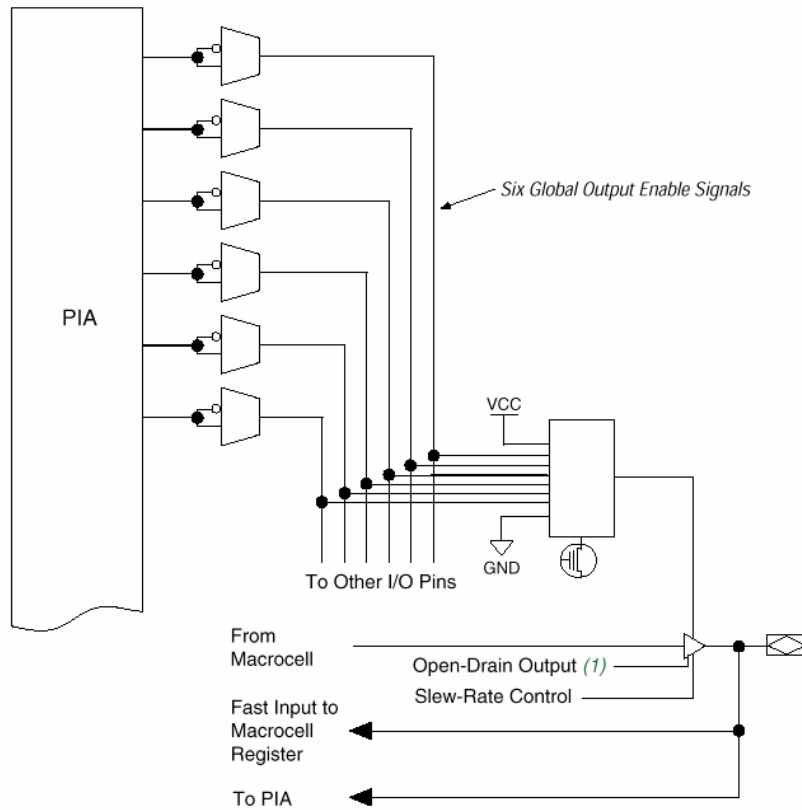
Chaque LAB contient 16 macro cellules (macro cell) configurables
 Chaque macro cell contient une partie combinatoire (matrice) et une fonction séquentielle avec une bascule D configurable en bascule D, JK T, SR, l'horloge peut être synchrone ou asynchrone

Macro Cell



IO Control Block

Les E/S peuvent être connectées au PIA, ou à une macro cell

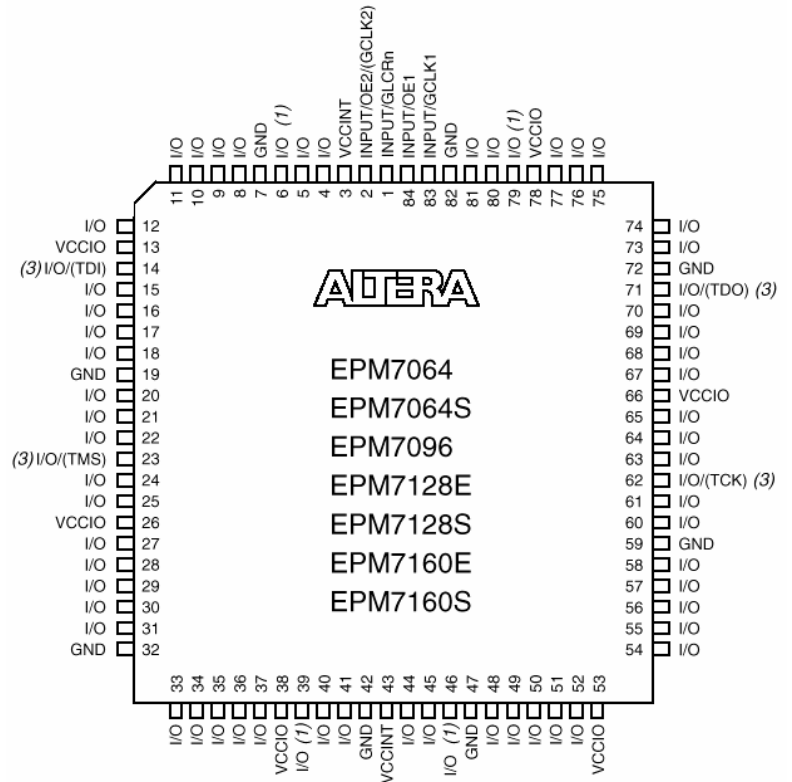


Brochage

Connexions des afficheurs 7 segments pour MAX7000

Segments	Afficheur 1	Afficheur 2
A	58	69
B	60	70
C	61	73
D	63	74
E	64	76
F	65	75
G	67	77
Dp	68	79

RQ: Les LED sont allumées par des "0"
Logiques
Les interrupteurs et poussoirs produisent un "0" lorsqu'ils sont au travail.



84-Pin PLCC

8. Exercices sur MAX+II

Attention, chaque exercice devra être rangé dans un sous répertoire de c:\max2work qui lui sera propre

8.1. Prise en mains de l'outil UP1 associé à MAX+PLUSII.

MAX+PLUSII permet de développer des applications sur les CPLD et FPGA d'ALTERA. UP1 est un KIT éducation supportant :

un CPLD **MAX7000S EPM7128SLC84-7** et un **FPGA FLEX F10K20RC240-4**

En fonction de cavaliers de la carte UP1 on peut programmer l'un ou l'autre des deux circuits.

MAX+PLUSII permet de développer les projets en AHDL (ABEL d'ALTERA), VERILOG, VHDL ou en utilisant les bibliothèques de fonctions logiques et de Mega-Fonctions d'ALTERA

Une interface graphique permet de placer des blocs fonctionnels qui peuvent être décrits dans n'importe lequel des trois langages, ou même sous forme de signaux grâce à l'éditeur d'onde.

Le langage utilisé dans ce TP sera le VHDL

Exercice 1:

On se propose ici de réaliser un simple inverseur entre deux broches du MAX 7000

⇒ Pour lancer MAX+PLUSII, utiliser le raccourci ou C:\MAX+PLUSII\MAXSTART.EXE

Le logiciel propose des structures des fichiers pré définies. Pour y accéder :

⇒ **File / New / Text Editor file** puis **OK** puis **Templates / VHDL Templates**, choisir

⇒ **Full Design Counter**, observer la structure du fichier, Supprimer les lignes inutiles et modifier le fichier comme suit

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY essai1 IS
    PORT
    (
        entree : IN    STD_LOGIC;
        sortie : out std_logic
    );
END essai1;

ARCHITECTURE combi OF essai1 IS
BEGIN
    sortie := not(entree);      -- une erreur (:= au lieu de <=) pour l'essai
END combi;
```

⇒ le sauvegarder avec **File / Save as** puis **tp1\essai1.vhd**.

Le logiciel propose alors de créer le répertoire **tp1**. Il est important de noter que la sauvegarde se fait avec l'extension **.vhd**, faute de quoi le fichier n'est pas interprété comme du VHDL.

⇒ Vérifier que le fichier est bien le projet en cours (barre du haut) sinon le déclarer comme tel : **File / Project / Set Project to Current File**.

Pour ce type d'instruction servant beaucoup, on mémorisera les raccourcis possibles donnés dans le menu.

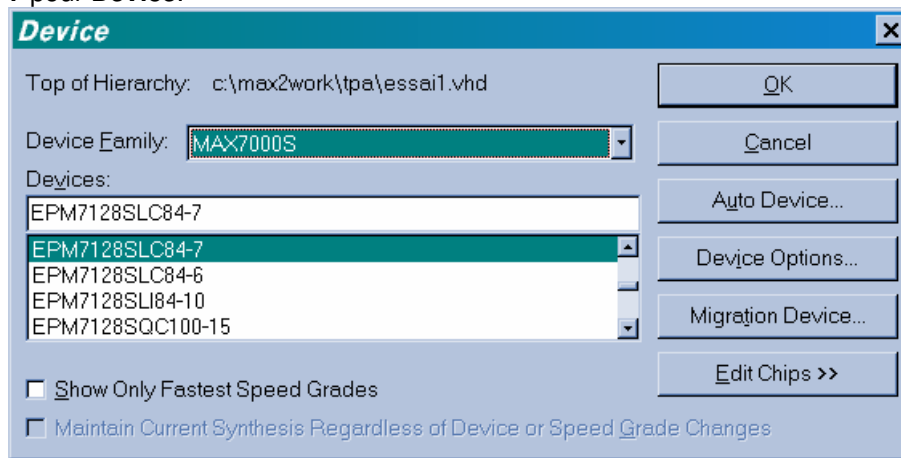
⇒ Compiler le projet : **File / Project / Save & Check**. (CTRL-K)

⇒ Observer les erreurs, en corriger une par **Message, Locate** puis effectuer la correction. Vérifier en recompilant que l'erreur à été corrigée.

⇒ Fermer ce fichier puis choisir d'autres fichiers du menu **VHDL Templates** pour observation.

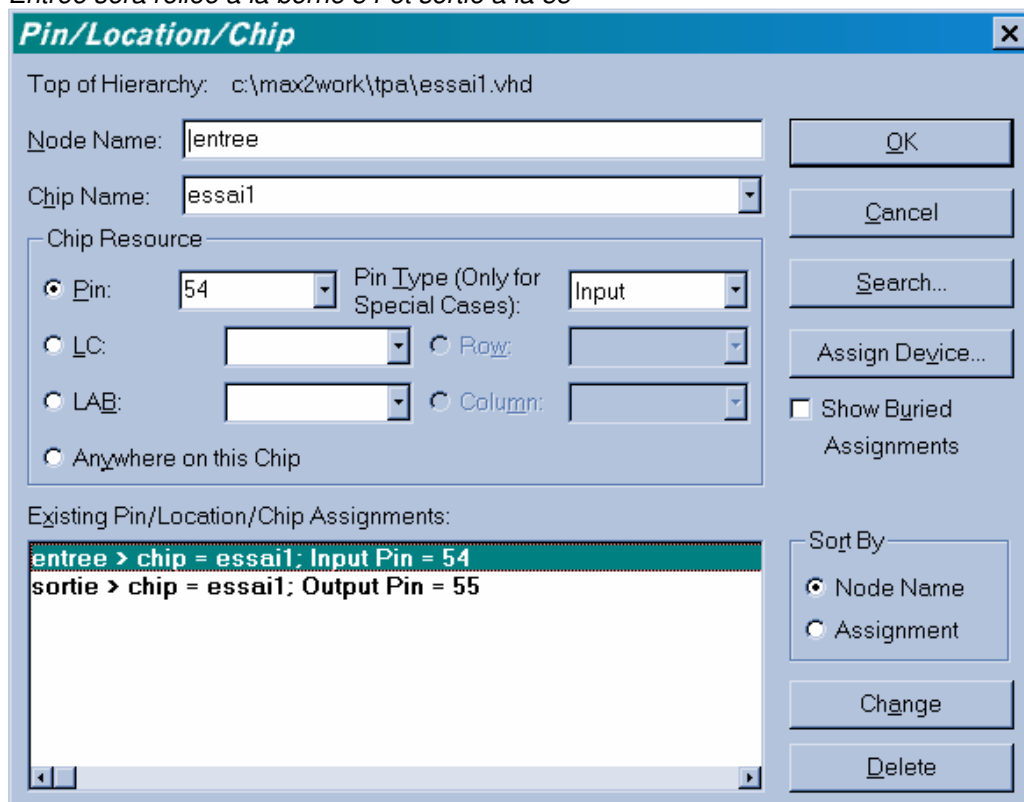
8.1.1. Assignation et compilation

- Déclarer ce projet comme le projet courant puis compiler.
- Choisir le circuit cible : **Assign / Device** puis **MAX7000S** pour **Device Family** et **EPM7128SLC84-7** pour **Device**.



- Pour affecter une borne : **Assign / Pin/Location/Chip** puis donner le nom de l'entrée - sortie dans **Node Name** et le numéro de broche dans **Pin**. Choisir **Add** puis **OK** lorsque toutes les affectations sont faites.

Entree sera reliée à la borne 54 et sortie à la 55



- Vérifier l'affectation de broche par **Max+plus II / Floorplan Editor** puis **Layout / Device View**. *Fermer cet éditeur avant de continuer la compilation.*

- Continuer la compilation par **Max+plus II / compileur** puis **Start**.

8.1.2. Programmation du circuit cible

⇒ Pour cela appeler le programmeur par **Max+plus II / Programmer**.

Si les informations sur le nom de cible et le nom du fichier .pof sont erronées alors :

⇒ cliquer : **JTAG / Multi-Device JTAG Chain Setup**. Déclarer alors le nom du circuit cible dans **Device Name** et celui du fichier de programmation (**divn.pof**, on pourra utiliser le "browser" **Select Programming File...**) dans **Select Programming File Name**. Choisir **Add** puis **OK**.

⇒ Programmer le circuit : **Program** et vérifier le bon fonctionnement sur la carte.

⇒ Sortir du logiciel : **File / Exit Max+plus II** et vérifier que la carte fonctionne toujours. *Le MAX7000S est programmé par grille flottante (EEPROM), la configuration est donc permanente. Vérifier cette caractéristique en débranchant puis rebranchant l'alimentation de la carte. L'inverseur fonctionne toujours.*

8.2. Clignoteur 1S

La carte UP1 possède un quartz à **25.175MHz**, ce signal est appliqué sur la **broche 83**.

On peut par exemple réaliser simplement une base de temps de 1s en divisant cette fréquence

⇒ Réaliser le projet **tp1\divfreq** avec le programme VHDL ci dessous (remarquer l'utilisation d'une variable), connecter clk à la broche 83 et clk_divise à la broche 55 (avec une LED connectée). Vérifier le bon fonctionnement

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY divfreq IS
PORT
    ( clk : IN std_logic;
      clk_divise: INOUT std_logic);    -- pour l'horloge de periode 1 seconde );
END divfreq;

ARCHITECTURE behavior OF divfreq IS
BEGIN
    -- divisons le 25.175 MHZ  pour atteindre la demi seconde

    PROCESS (clk)
        variable cnt : integer range 0 to 12587500;
        CONSTANT diviseur : integer := 12587500; -- 12587500 mettre 10 pour la simulation

    BEGIN
        IF (clk'EVENT AND clk = '1') THEN
            IF cnt = diviseur THEN
                cnt := 0;
                clk_divise <= not clk_divise;
            ELSE cnt:= cnt + 1;
            END IF;
        END IF;

    END PROCESS;

END behavior;
```

8.3. Comptage

Le projet se trouve toujours dans TP1.

⇒ La création d'un compteur est très simplifiée en utilisant **File / New / Text Editor file** puis **OK** puis **Templates / VHDL Templates**, choisir **Full Design Counter**

Compléter la structure afin d'obtenir le fichier suivant :

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
--USE ieee.std_logic_arith.all;-- si sortie est un bus
--USE ieee.std_logic_unsigned.all;

ENTITY cptud IS
  PORT
  (
    clk          : IN   STD_LOGIC;
    raz          : IN   STD_LOGIC;
    ud           : IN   STD_LOGIC;
    sortie       : OUT  INTEGER RANGE 0 TO 255
    --sortie : OUT STD_LOGIC_VECTOR(7 downto 0) -- si sortie est un bus
    -- dans ce cas sortie peut être INOUT, le signal compte devient inutile
  );
END cptud;

ARCHITECTURE a OF cptud IS
  SIGNAL compte : INTEGER RANGE 0 TO 255;
-- SIGNAL compte : STD_LOGIC_VECTOR (7 downto 0); -- si sortie est un bus
BEGIN
  PROCESS (clk, raz)
  BEGIN
    IF raz = '1' THEN compte <= 0;
    ELSIF (clk'EVENT AND clk = '1') THEN
      IF UD='1' THEN compte <= compte + 1;
      ELSE compte <= compte - 1;
      END IF;
    END IF;
  END PROCESS;
  sortie <= compte;
END a;
```

⇒ Tapez **CTRL-K** pour compiler, s'il n'y a pas d'erreur valider la simulation fonctionnelle par

⇒ **Processing/Fonctionnal SNF extractor** puis terminer le compilation par **START**.

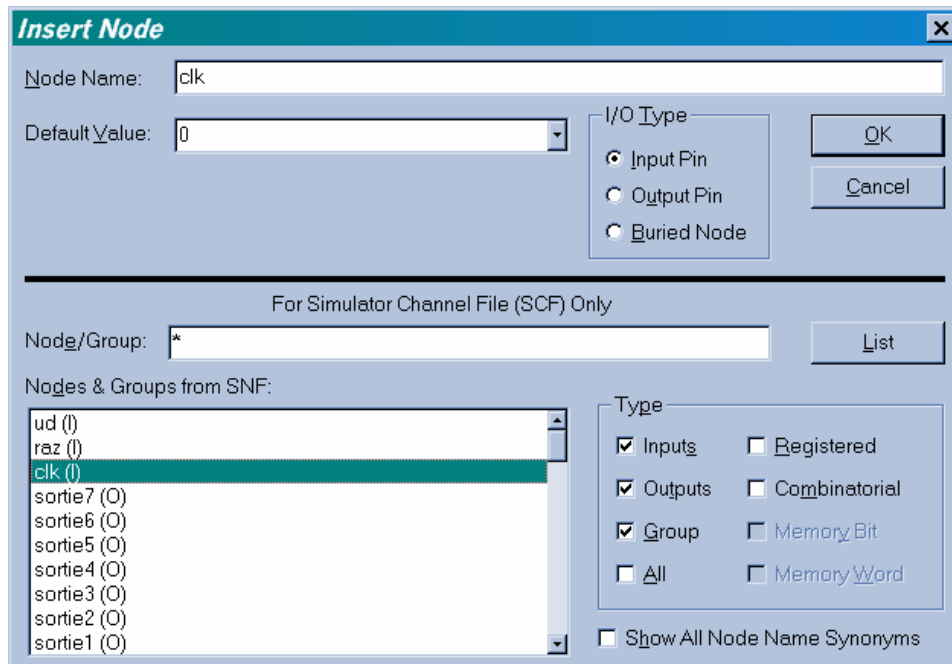
Le compilateur génère un fichier pour la simulation fonctionnelle (qui ne tient pas compte de la cible envisagée)

⇒ Cliquer **MAX+PLUSII/Waveform Editor**

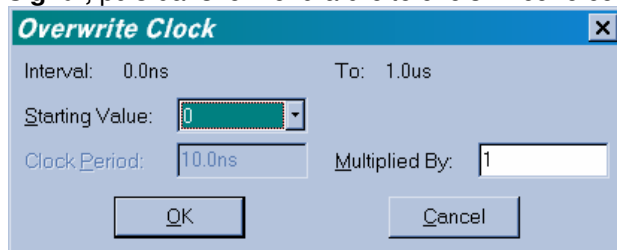
⇒ **Options/Grid size** permet de définir la résolution du simulateur, choisir 5ns ce qui permettra d'obtenir une horloge de 10ns.

⇒ **File/End time** permet de définir la durée totale de la simulation, choisir 1us

DoubleCliquer dans la zone **name**, sélectionner les signaux clk, puis ud, puis raz, puis sortie(O).



La création des stimuli est très simple. Pour l'horloge: cliquer sur **clk** pour sélectionner **TOUT le signal**, puis dans le menu à droite choisir l'icône correspondant à **node with a clock waveform**

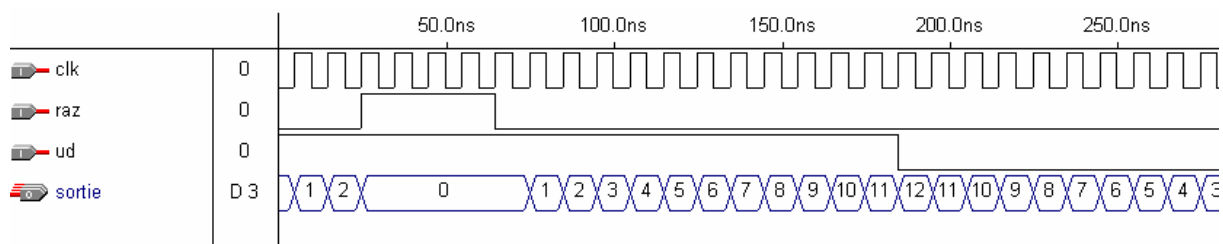


⇒ Cliquer **OK**

Pour RAZ et UD **sélectionner une zone de signal** dans l'éditeur de stimuli puis choisir dans le menu à gauche la valeur 1 ou 0 pour cette zone.

⇒ Enregistrer le fichier de stimuli dans **cptud.scf**

⇒ Pour lancer la simulation: **Max+plus II/simulator START** puis **open SCF**, le résultat doit ressembler à celui ci



⇒ Sélectionner la **fenêtre VHDL, CTRL-K**, désactiver alors **Processing/fonctionnal SNF extractor**, le compilateur va maintenant intégrer le projet dans la cible.

⇒ Choisir le MAX7000 de UP1, puis START, le compilateur assigne automatiquement les broches.

⇒ Le fichier **cptud.rpt** (report) contient toutes les informations concernant le composant réalisé avec le max7000.

Il est possible maintenant d'effectuer une simulation **"timed"** en utilisant le même fichier de simulation.

Rq: l'horloge étant à 25.175 MHz il n'est pas possible de visualiser les résultats sur des LED.

8.4. Création d'un projet autour de fonctions

Pour pouvoir visualiser le comptage des secondes sur les afficheurs 7 segments il faut encore créer un décodeur binaire/7 segments. Voici le fichier VHDL **DEC7SEG.VHD**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
ENTITY dec7seg IS
    PORT (
        digit : in bit_vector (7 downto 0);
        sortieD : OUT BIT_VECTOR(0 to 6); -- segments afficheur Droit
        sortieG : OUT BIT_VECTOR(0 to 6) -- segments afficheur Gauche );
END dec7seg;

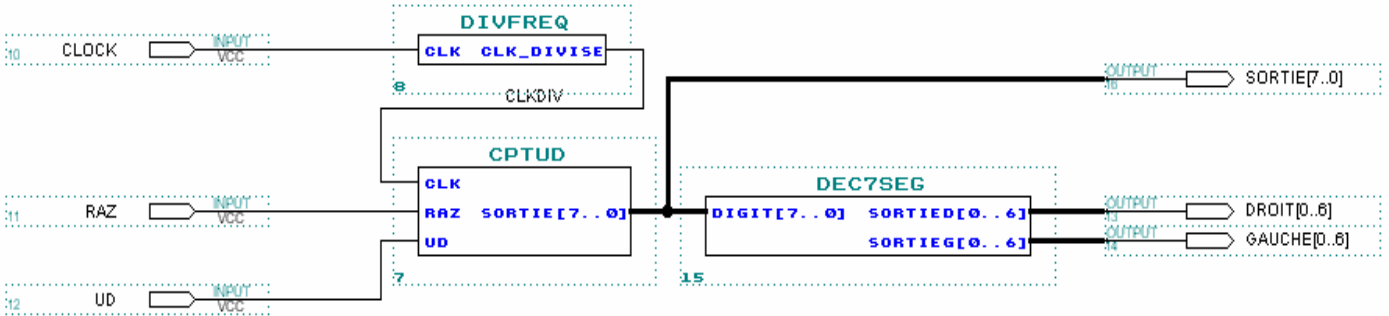
ARCHITECTURE behavior OF dec7seg IS
BEGIN
PROCESS (digit(3 downto 0))
    -- affichage du digitD
BEGIN
CASE digit(3 downto 0) is -- abcdefg segments de l'afficheur droit, allumage par des "0"
    WHEN "0000" => sortieD <= "0000001";
    WHEN "0001" => sortieD <= "1001111";
    WHEN "0010" => sortieD <= "0010010";
    WHEN "0011" => sortieD <= "0000110";
    WHEN "0100" => sortieD <= "1001100";
    WHEN "0101" => sortieD <= "0100100";
    WHEN "0110" => sortieD <= "0100000";
    WHEN "0111" => sortieD <= "0001111";
    WHEN "1000" => sortieD <= "0000000";
    WHEN "1001" => sortieD <= "0000100";
    WHEN "1010" => sortieD <= "0001000";
    WHEN "1011" => sortieD <= "1100000";
    WHEN "1100" => sortieD <= "0110001";
    WHEN "1101" => sortieD <= "1000010";
    WHEN "1110" => sortieD <= "0110000";
    WHEN "1111" => sortieD <= "0111000";
    WHEN others => sortieD <= "1111111";
END CASE;
END PROCESS;

PROCESS (digit(7 downto 4))
    -- affichage du digitG
BEGIN
CASE digit(7 downto 4) is -- abcdefg segments de l'afficheur gauche , allumage par des "0"
    WHEN "0001" => sortieG <= "1001111";
    WHEN "0010" => sortieG <= "0010010";
    WHEN "0011" => sortieG <= "0000110";
    WHEN "0100" => sortieG <= "1001100";
    WHEN "0101" => sortieG <= "0100100";
    WHEN "0110" => sortieG <= "0100000";
    WHEN "0111" => sortieG <= "0001111";
    WHEN "1000" => sortieG <= "0000000";
    WHEN "1001" => sortieG <= "0000100";
    WHEN "1010" => sortieG <= "0001000";
    WHEN "1011" => sortieG <= "1100000";
    WHEN "1100" => sortieG <= "0110001";
    WHEN "1101" => sortieG <= "1000010";
    WHEN "1110" => sortieG <= "0110000";
    WHEN "1111" => sortieG <= "0111000";
    WHEN others => sortieG <= "1111111";
END CASE;
END PROCESS;
END behavior;
```

➤ Analyser et compiler ce fichier, et vérifier qu'il n'y a pas d'erreur

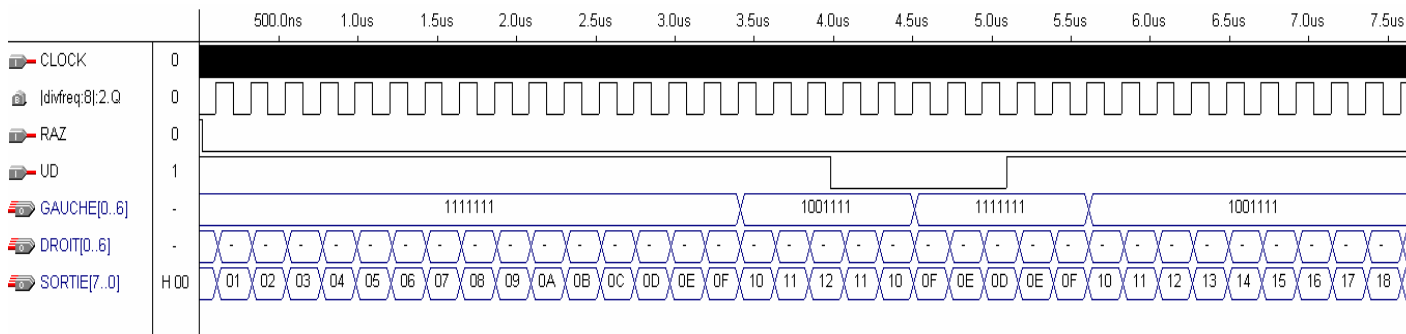
➤ Créer maintenant un schéma par **NEW/graphic editor file** et l'enregistrer sous le nom **tp1/cptaff.gdf** et donner au projet actif ce nom.

➤ Dans la fenêtre d'édition, cliquer à droite puis **enter symbol**. Placer les symboles **divfreq**, **cptud** et **dec7seg**. Pour placer les **input** et **output** tapez leur nom, et les placer de manière à obtenir le schéma suivant :

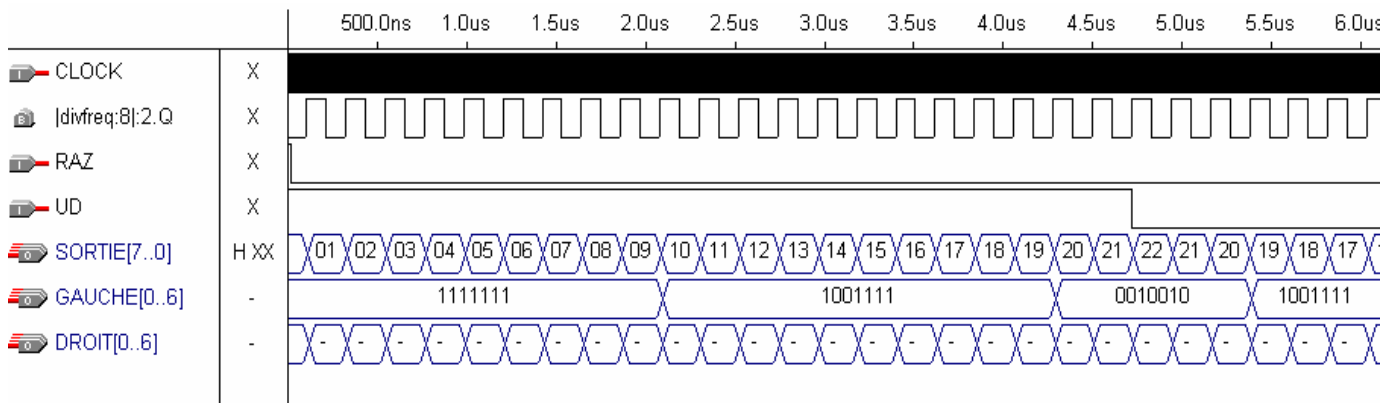


➤ Forcer le choix de UD, RAZ et des sorties 7 segments conformément au KIT UP1, l'entrée CLOCK sera connectée à la broche 83. (25.175MHz)

Le résultat de simulation est le suivant (en modifiant le diviseur de fréquence de 125715000 en 10).



Exercice : modifier la fonction CPTUD en CPTUDDEC, de manière à compter en décimal sur le bus sortie et obtenir les chronogrammes suivants.

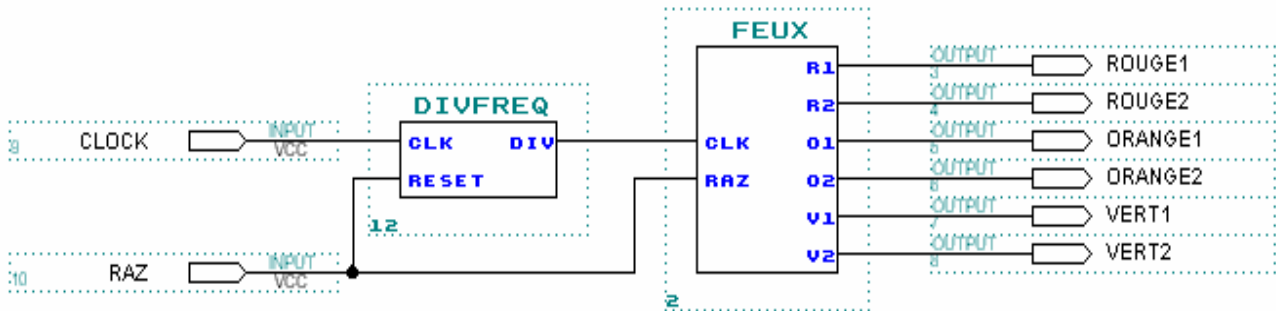


8.5. Feux tricolores

Le séquenceur de feux tricolores est composée de **trois process** :

- Un TIMER qui produit un signal **Tfini** à la fin de la tempo de durée : **duree** qui vaudra **DV1,DV2 ou DO**
- Une machine d'état qui en fonction de **Tfini** défini les séquences des feux
- Un décodeur qui en fonction des états de la machine allume les feux (allumage par 0)

La vitesse d'évolution des feux est fonction du signal CLK, lors de la simulation CLK ne sera pas divisé. Avant synthèse on créera un projet schéma avec la fonction FEUX et la fonction **divfreq** étudiée précédemment de manière à produire une base de temps pour les feux de 1s



Exercice : Compléter le programme VHDL ci dessous

```

Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_arith.all;

ENTITY feux is
    PORT (
        R1,R2,O1,O2,V1,V2    : OUT STD_LOGIC;
        CLK,RAZ              : IN STD_LOGIC;
    );
END feux;

ARCHITECTURE behavior OF feux IS
    type etat is (un,deux,trois,quatre);
    signal actuel, futur : etat;
    signal compte, duree : integer range 0 to 255;
    signal Tfini : std_logic;
    constant DV1: integer:=8;      -- les temps sont relatifs à la période de CLK
    constant DV2: integer:=5;
    constant DO : integer:=1;

BEGIN
    PROCESS(clk,RAZ)              --le timer est ici
    BEGIN
        if RAZ='1' then compte<=0;Tfini<='0';
        elsif (clk'event and clk='1') then
            if compte = duree then
                compte <=?;
                Tfini <=?;
            else
                Tfini<=?;
                compte <=?;
            end if;
        end if;
    END PROCESS;
    
```

```

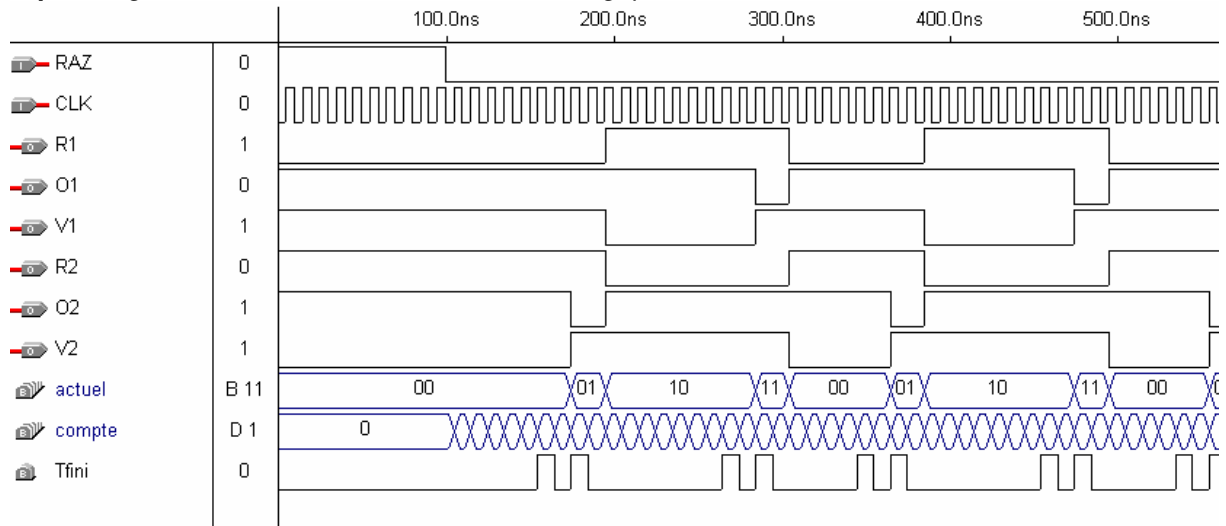
PROCESS (clk,RAZ)
BEGIN
  if RAZ ='1' then actuel<=un;duree<=DV2;
  elsif (clk'event and clk='1') then
    actuel<=futur;
    case actuel is
      when un => -- feux 2 vert, DV2 en cours
        if Tfini='?' then futur <= ?; duree <= ?; end if;
      when deux => -- feux 2 orange, DO en cours
        ???????
      when trois => -- feux 1 vert, DV1 en cours
        ???????
      when quatre => -- feux 1 orange, DO en cours
        ???????
      when others => ???????
    end case;
  end if;
END PROCESS;

PROCESS(actuel)
BEGIN
  case actuel is -- attention un '0' allume une LED sur UP1
    when un => R1<='0';O1<='1';V1<='1';R2<='1';O2<='1';V2<='0';
    when deux => ???????
    when trois => ???????
    when quatre => ???????
    when others => ???????
  end case;
END PROCESS;
END behavior;

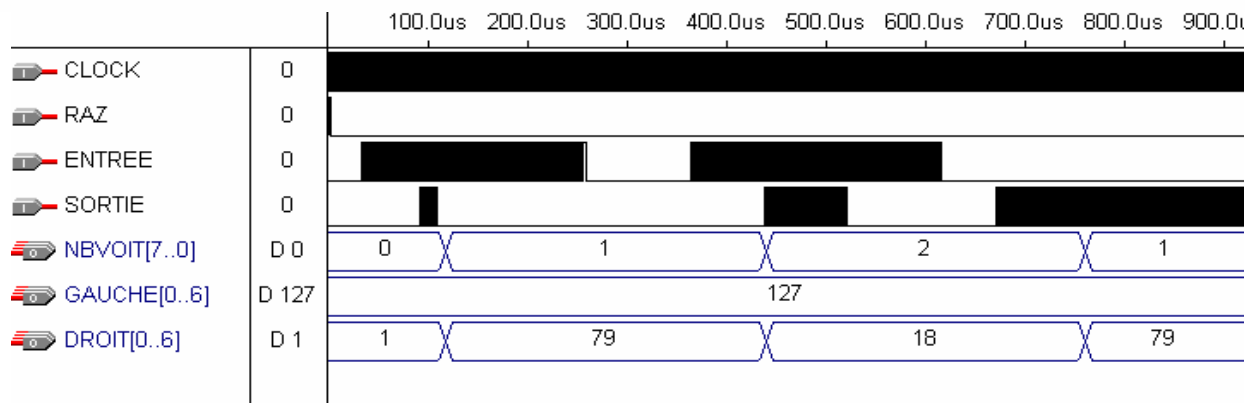
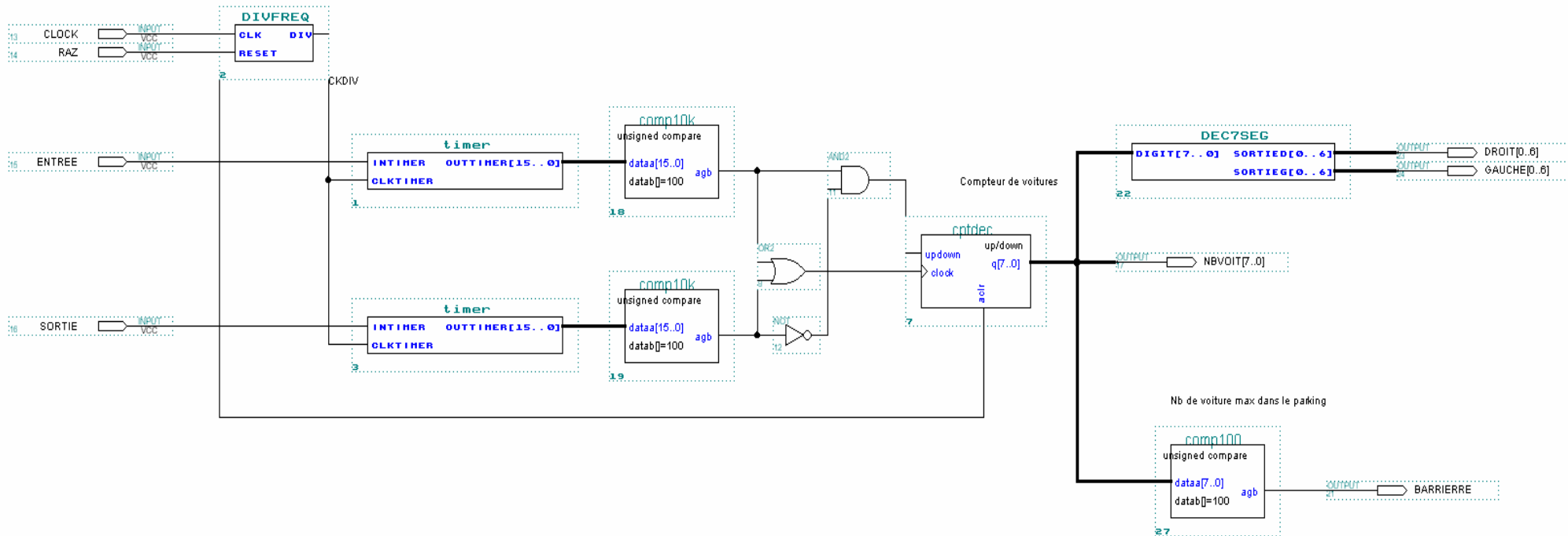
```

Résultats attendus

Rq : les signaux de sorties sont actifs au niveau logique 0 conformément aux LED du KIT UP1



8.6. Utilisation de TIMER : gestionnaire de parking



8.6.1. Macro_Fonctions

MAX+PLUSII permet la création de fonctions logiques à l'aide de composants de type portes mais aussi à l'aide de **MACRO-FONCTIONS** appelées **lpm: Lybrary of parameterized modules**)
L'aide de MAX+PLUSII fournie les détails sur ses fonctions.

Le projet "gestion de parking" est construit et peut être simulé avant de suivre ce TP. (répertoire TIMER)

On se propose ici de montrer le démarche de création d'un "timer" à l'aide des macro-fonctions **lpm** de MAX+PLUSII

Description du TIMER

Le compteur du timer sera initialisé sur le front montant de son horloge et comptera les impulsions en entrée sur l'état haut de celle ci , il mémorisera le nombre compté sur le front descendant.

- ⇒ Créer un nouveau projet graphique **TPTIMER** dans un répertoire du même nom.
- ⇒ Cliquer sur l'espace de dessin puis **cliquer droit./enter symbol** puis **Mega Wizard Plug-in manager** puis **create a new custom megafonction variation**
- ⇒ Choisir **arithmetic LPM COUNTER** et valider **VHDL**. Donner comme nom **cpt**.
- ⇒ Valider ensuite **UP ONLY- 16 bits – Plain Binary – Asynchronous Clear**

On peut également charger directement un compteur pré construit et le paramétrer

- ⇒ Sur le schéma cliquer droit puis **enter/symbol** puis sélectionner la **librairie mega_lpm** puis le composant **lpm_counter**.
- ⇒ Le placer sur la feuille puis cliquer droit puis **edit Port/parameters**, l'écran de configuration apparaît.
- ⇒ On peut alors **déclarer Used ou Unused les ports du compteur**. Donner ensuite la valeur 16 à **lpm_with** qui correspond à la capacité de comptage.

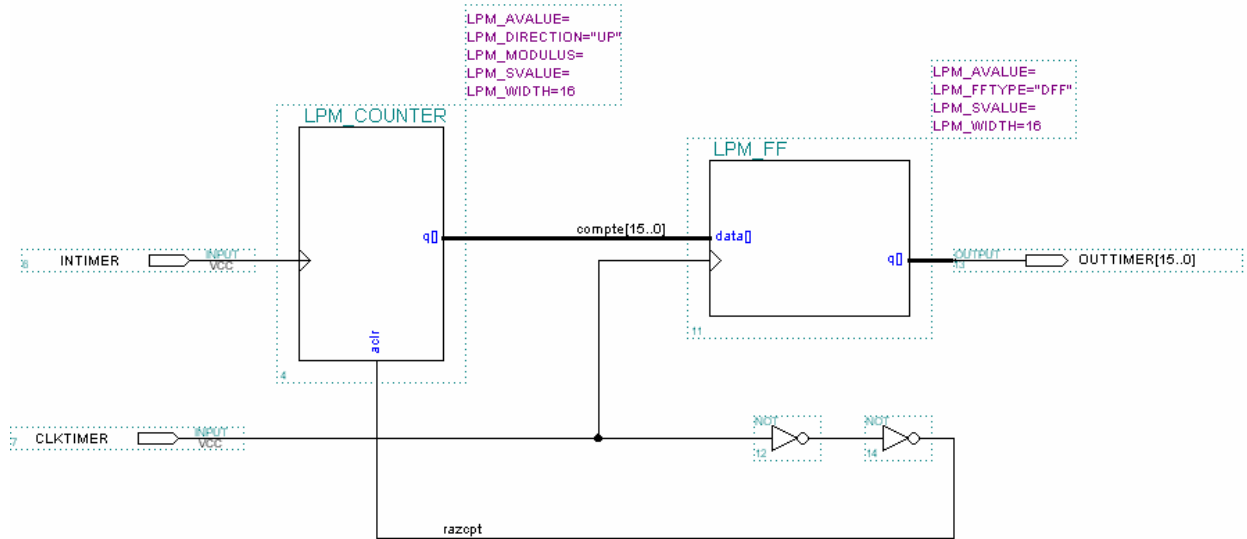
The screenshot shows the 'Edit Ports/Parameters' dialog box for the LPM_COUNTER component. The 'Function Name' is 'LPM_COUNTER'. The 'Ports' section shows a table of ports with their status and inversion settings. The 'Parameters' section shows a table of parameters with their values.

Name	Status	Inversion
aclr	Used	None
aconst	Unused	None
aload	Unused	None
aset	Unused	None
clk_en	Used	None

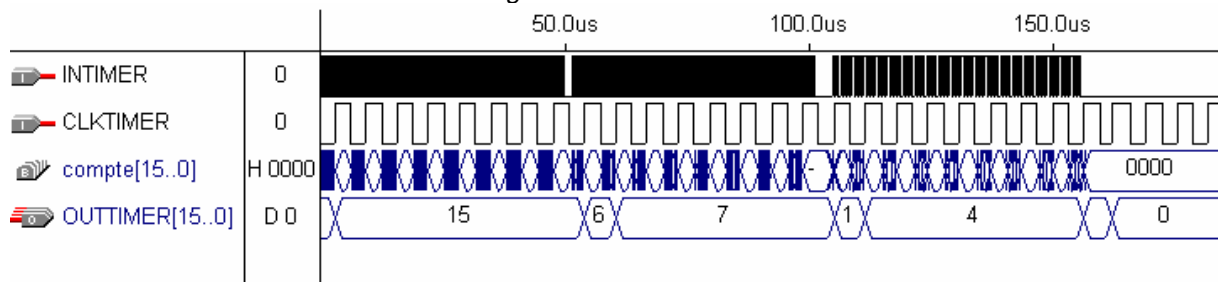
Name	Value
LPM_AVALUE	<none>
LPM_DIRECTION	<none>
LPM_MODULUS	<none>
LPM_SVALUE	<none>
LPM_WIDTH	<none>

La fonction **mémoire** sera construite avec un **composant custom LPM_FF**.

⇒ Réaliser le TIMER de manière à obtenir le schéma suivant :



La simulation du timer donnera les chronogrammes suivants :



⇒ Construire le symbole du timer par **File/create default symbol**

⇒ Tester ce symbole en créant un schéma l'utilisant et vérifier son fonctionnement

8.7. La série FLEX 10K

Le composant présent sur le KIT UP1 est le EPF10K20

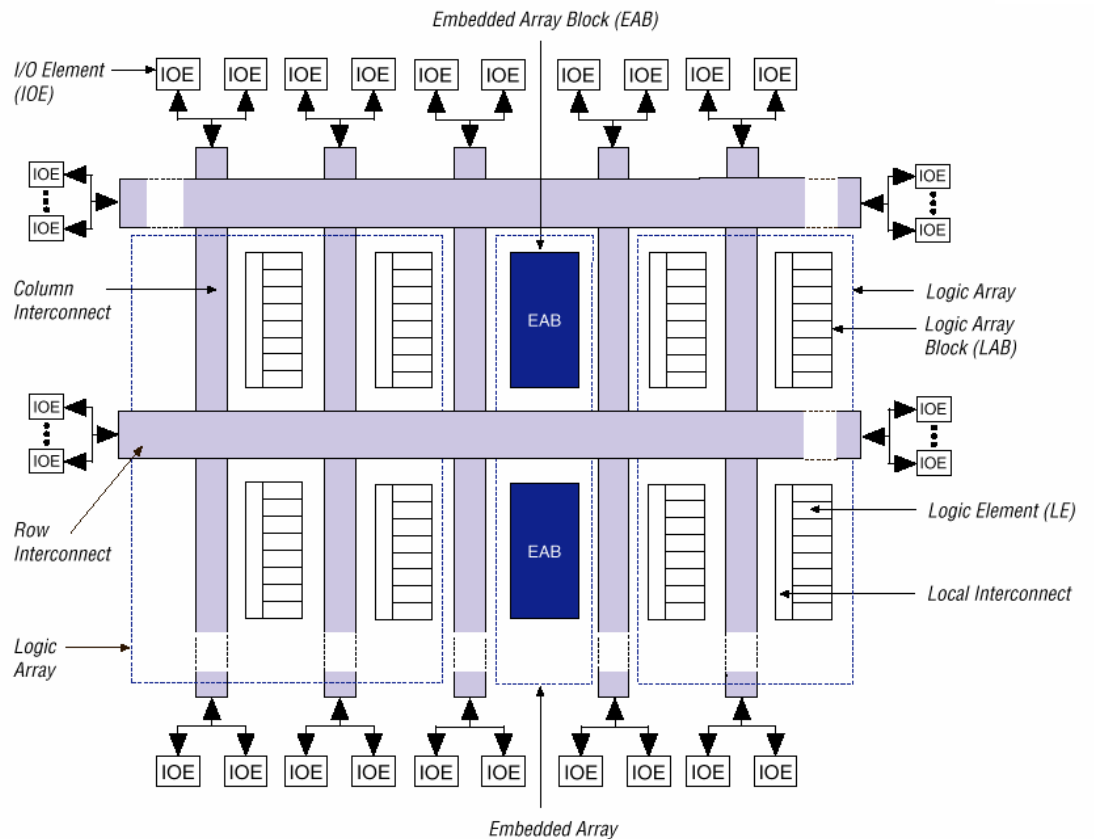
Table 1. FLEX 10K Device Features

Feature	EPF10K10 EPF10K10A	EPF10K20	EPF10K30 EPF10K30A	EPF10K40	EPF10K50 EPF10K50V
Typical gates (logic and RAM) (1)	10,000	20,000	30,000	40,000	50,000
Maximum system gates	31,000	63,000	69,000	93,000	116,000
Logic elements (LEs)	576	1,152	1,728	2,304	2,880
Logic array blocks (LABs)	72	144	216	288	360
Embedded array blocks (EABs)	3	6	6	8	10
Total RAM bits	6,144	12,288	12,288	16,384	20,480
Maximum user I/O pins	150	189	246	189	310

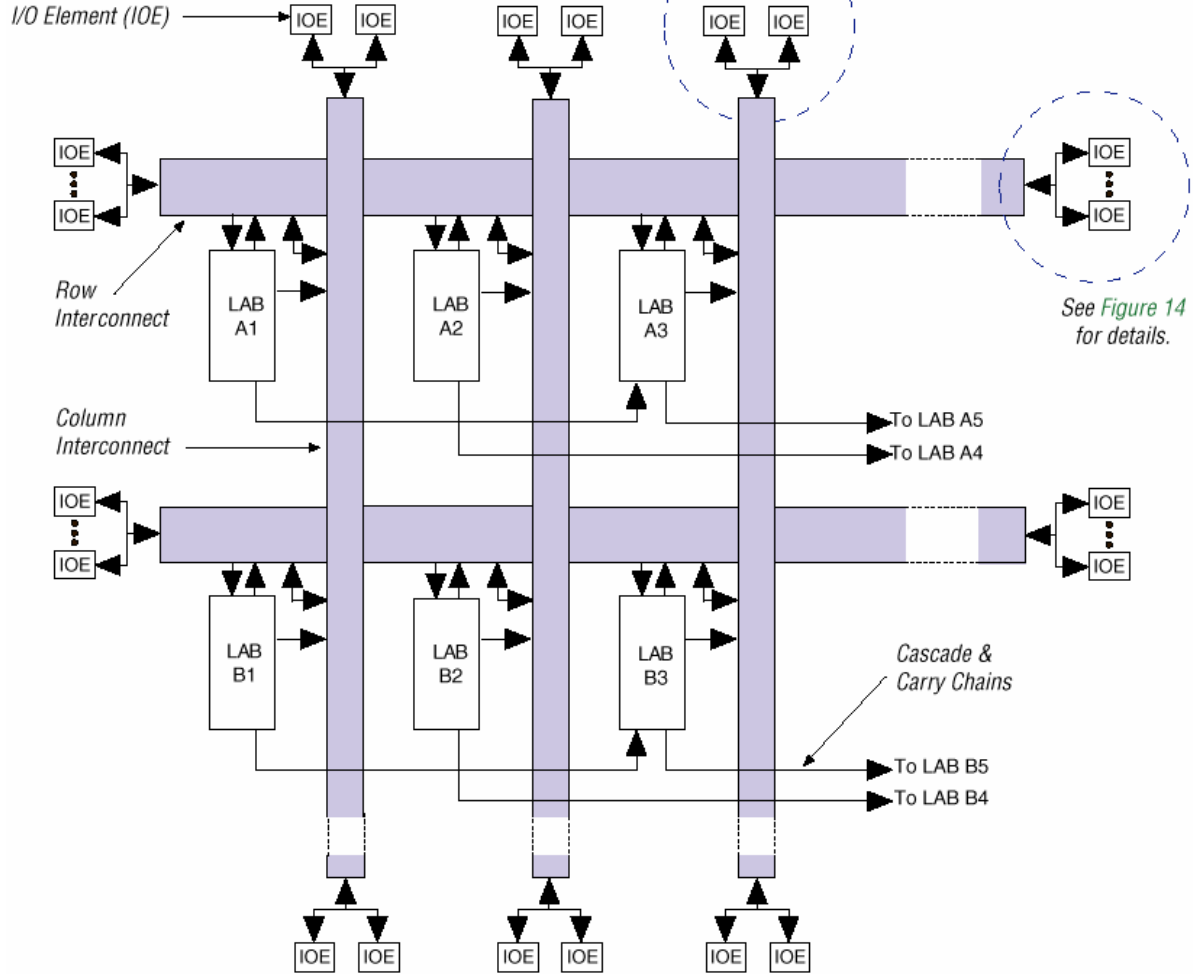
Table 2. FLEX 10K Device Features

Feature	EPF10K70	EPF10K100 EPF10K100A	EPF10K130V	EPF10K250A
Typical gates (logic and RAM) (1)	70,000	100,000	130,000	250,000
Maximum system gates	118,000	158,000	211,000	310,000
LEs	3,744	4,992	6,656	12,160
LABs	468	624	832	1,520
EABs	9	12	16	20
Total RAM bits	18,432	24,576	32,768	40,960
Maximum user I/O pins	358	406	470	470

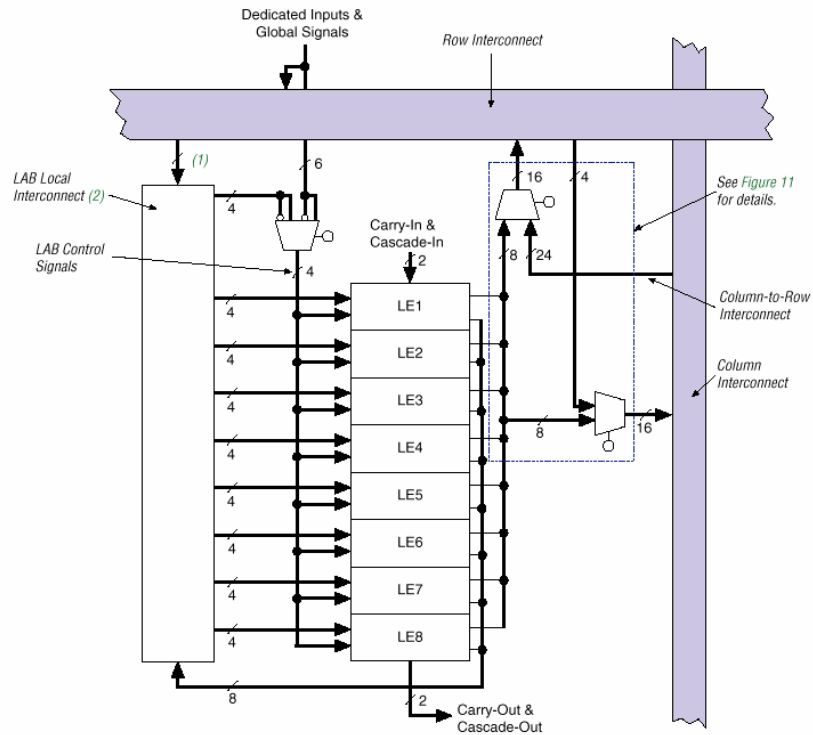
Structure interne



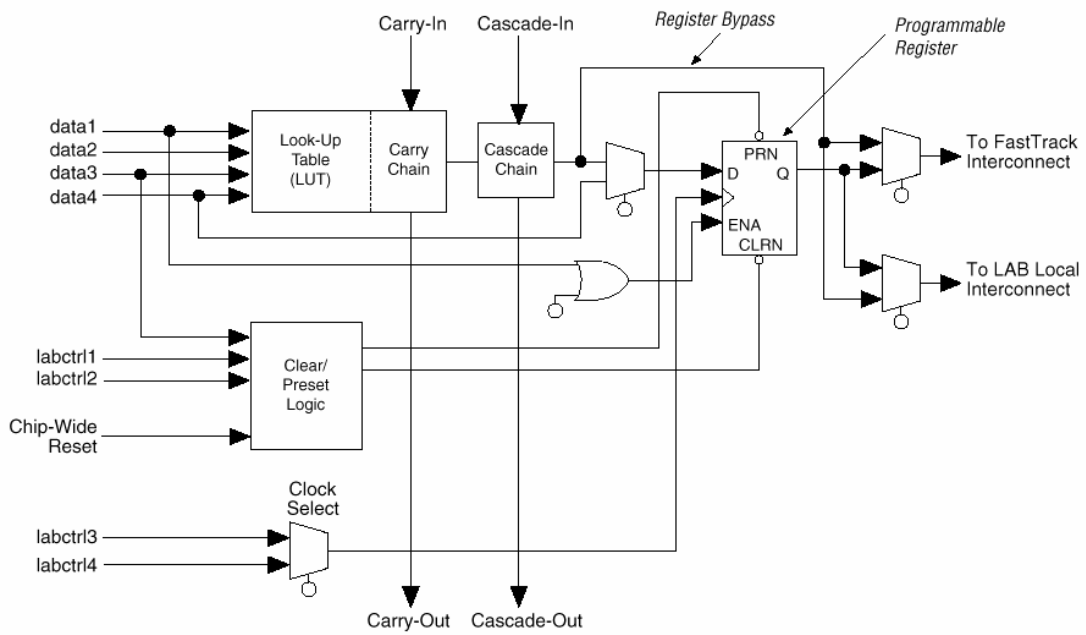
Interconnexions Logic Array LA



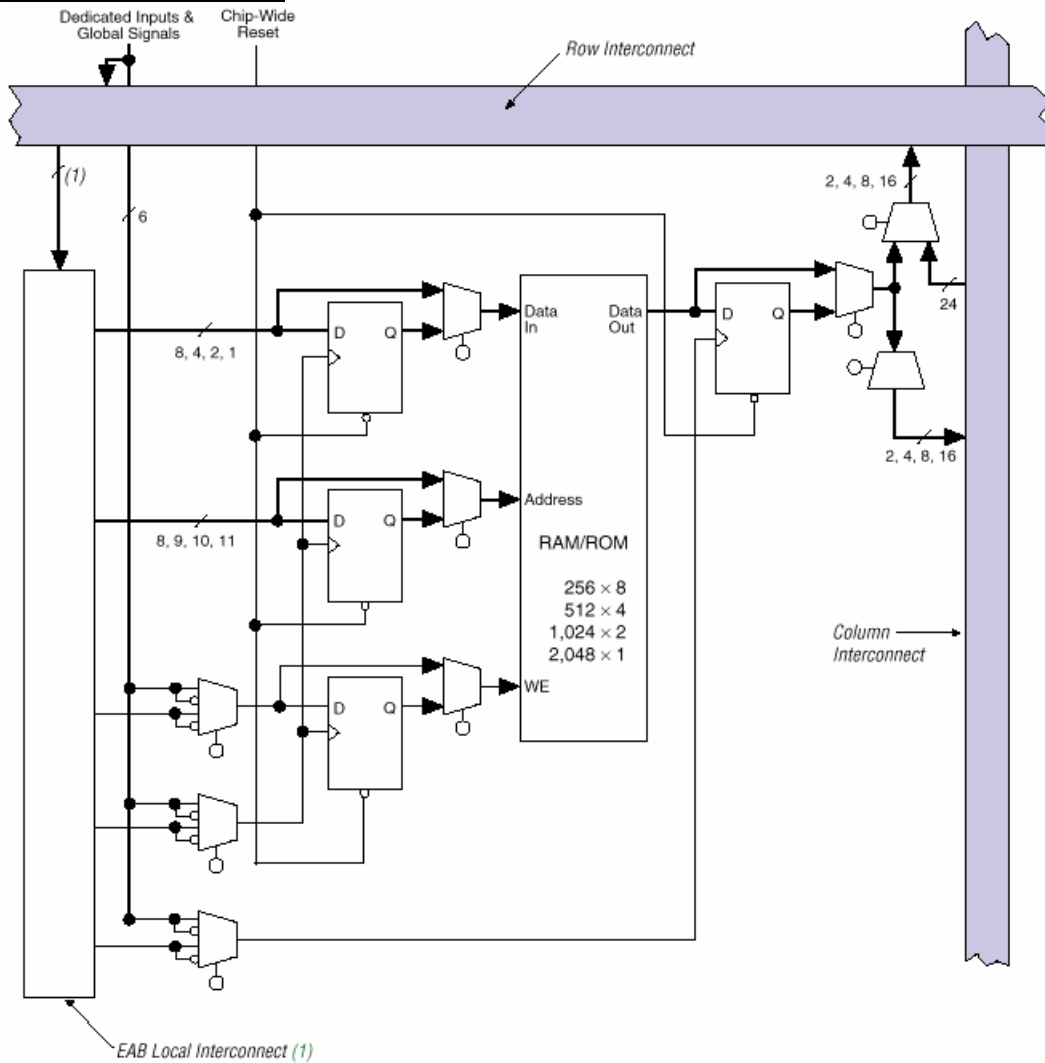
Logic Array Block LAB



Logic element LE



Embedded Array Block EAB



9. Bibliographie

- ❑ Initiation au langage VHDL. 2^{ème} édition . Michel Auniaux. . Dunod
- ❑ <http://www.cnfm.fr> Le site de la programmation en France avec de nombreuses infos sur le VHDL et en particulier XILINX et ALTERA
- ❑ <http://www.supelec-rennes.fr/ren/rd/etscm/base/altera/up1.htm> Le site de Jacques Weiss, avec des docs sur le VHDL et des exemples d'applications sur la carte UP1 d'ALTERA
- ❑ <http://users.ece.gatech.edu/~hamblen/ALTERA/altera.htm> des infos et des liens sur le VHDL
- ❑ <http://perso.wanadoo.fr/joelle.maillefert/> Des cours sur le C, le C++ Builder et sur le VHDL
- ❑ <http://www.alsdesign.fr> pour demander une version d'évaluation de la suite OrCAD
- ❑ <http://www.chez.com/amouf/syntaxe.htm> syntaxe du VHDL par Alain Moufflet
- ❑ <http://www.home.ch/~spaw4366/publica.htm> un cours pour ingénieurs
- ❑ <http://www.multimania.com/sbernard/dossier/vhdl/vhdl.htm> plein de liens mais souvent en Anglais
- ❑ [Les circuits logiques programmables : Faisons le point](#) - fichier .pdf et .ppt zippé 717Ko - L.P. Ampère de Marseille
<http://www.electron.cndp.fr/documents/Ressources/Contributions/pellet-azoulay/pld.exe>
- ❑ [Fonctions logiques élémentaires et langages comportementaux \(ABEL-VHDL\)](#) - fichier .pdf 353Ko - Gérard Bonnet
<http://electron.toulouse.iufm.fr/iufm/cours/vhdl/AbelVhdl.pdf>
- ❑ [Le langage de description VHDL](#) - fichier .pdf 208Ko - Thierry Blotin
<http://electron.toulouse.iufm.fr/iufm/cours/vhdl/vhdl.pdf>
- ❑ [Le langage VHDL pour traduire le fonctionnement d'une machine d'états](#) - fichier .pdf 503Ko - Patrick Cohen
<http://electron.toulouse.iufm.fr/iufm/cours/vhdl/vhdl.pdf>
- ❑ [Familles de composants programmables Actel, Altera, Atmel, Star et Xilinx](#) - Jacques Weiss
<http://electron.toulouse.iufm.fr/iufm/cours/vhdl/vhdl.pdf>
- ❑ [Génération d'images vidéo par circuit logique programmable](#) - fichier .pdf 144Ko - Gérard Bonnet
<http://www.electron.cndp.fr/documents/Ressources/Contributions/video/videopl.pdf>