SYNTAXE SHDL PAR L'EXEMPLE

Modules

```
module xor3(a, b, c : s)          // module à 3 entrées a,b,c et une sortie s
  s = a*/b*/c + /a*b*/c + /a*/b*c + a*b*c; // équation combinatoire
end module

module xor6(a, b, c, d, e, f, s)        // le ':' n'est pas obligatoire
  xor3(a, b, c, s1);              // incorporation d'une instance du module xor3
  xor3(d, e, f, s2);              // incorporation d'une 2ème instance du module xor3
  s = s1*/s2 + /s1*s2;            // s = xor des deux sorties intermédiaires
end module

module xor6V(e[5..0] : s)         // même module avec un vecteur d'entrée
  xor3V(e[2..0], s1);
  xor3V(e[5..3], s2);
  s = s1*/s2 + /s1*s2;
end module

module descendant(aclr, h, : down)      // circuit séquentiel
  down := e;                      // := affectation séquentielle (ici : bascule D)
  down.clk = h;                   // horloge, front montant
  down.rst = aclr;                // signal de reset asynchrone
end module
```

Affectations combinatoires

```
x = y*z + u*v*w ;                 // forme générale : somme de termes scalaires
x = 0 ;                           // autorisé car la valeur est 0 ou 1
x = 123 ;                         // INTERDIT (x scalaire, arité 1)
x[7..0] = 123 ;                   // pas de problème car 123 tient sur 8 bits
x[7..0] = 623 ;                   // INTERDIT (incompatibilité d'arités)
x[7..0] = -123 ;
x[7..0] = 0x7E ;
x[7..0] = 0b10101010 ;
x[7..0] = y[7..0]*z[7..0]*u[7..0] ;     // AND position par position
x[7..0] = y[7..0]*z[6..0] ;             // INTERDIT (incompatibilité d'arités)
x[7..0] = y[7..0]+z[7..0]+u[7..0] ;     // OR position par position
x[7..0] = y[7..0]+z[6..0] ;             // INTERDIT (incompatibilité d'arités)
x[7..0] = y*z[7..0] ;                   // factorisation du scalaire y
```

```
x[7..0] = y*z[7..0]*u[7..0]*v ;        // factorisation et AND par position
x[7..0] = y[7..0] * 0b1010 ;
x[7..0] = y[7..0] * 0b1010101010 ;     // INTERDIT (incompatibilité d'arités)
x[7..0] = y[7..0] + 0b1101 ;
```

Tri-state

```
x = y:oe;                          // buffer 3-états de y vers x, commandé par oe
x = /y:/oe;                // idem avec des inversions sur l'entrée et la commande
x = y*z:oe;                        // INTERDIT (tri-state = 1 buffer 3 états)
x[7..0] = y[7..0]:/oe;             // vecteur de buffers 3-états commande commune
x[7..0] = 0b10101010:oe;           // entrée littérale, écrite en binaire
x[7..0] = y[7..0]:0b10101010;
x[7..0] = 0b11110000:0b10101010;
x[7..0] = 0b1010:0b10101010;
x[7..0] = 0b11110000:0b101010;     // INTERDIT (incompatibilité d'arités)
```

Affectations séquentielles

```
q := d ;                           // équation d'évolution de la bascule D
q := /d ;                          // bascule D avec entrée inversée
q := /t*q + t*/q ;                 // bascule T
q := /q*t + /t*q ;                 // bascule T avec entrée inversée
q := /k*q + j*/q ;                 // bascule JK
q := a*q + b*/q ;                  // bascule JK : b=J, /a=K
q := a*q + b*/c ;                  // INTERDIT (ne correspond à aucune bascule)
q := a*q ;                         // INTERDIT (ne correspond à aucune bascule)
q.rst = /nrst ;                    // reset asynchrone sur niveau bas
q.rst = a * b ;                    // INTERDIT (seulement un terme à un seul signal)
q.ena = en ;                       // en = signal d'enable synchrone
q.set = set ;                      // set = signal de mise à 1 asynchrone
q.clk = /h ;                       // horloge de la bascule : front descendant de h
q[7..0] := 123 ;
q[7..0] := /d[7..0] ;              // vecteur de bascules D
q[7..0] := /t[7..0]*q[7..0] + t[7..0]*/q[7..0] ;     // vecteur de bascules T
q[7..0].ena = en ;                 // mise en commun
q[7..0].ena = en[7..0] ;
q[7..0].ena = en[5..0] ;           // INTERDIT (incompatibilité d'arités)
```

Signaux prédéfinis sur Nexys 2

```
mclk                // horloge 50MHz
btn[3..0]           // 4 boutons poussoir
```

```
sw[7..0]            // 8 switches à glissière
ld[7..0]            // 8 leds haute luminosité
ssg[7..0]           // cathodes (dp, g, f, e, d, c, b, a) des afficheurs 7 segments
an[3..0]            // anodes des afficheurs 7 segments
red, grn, blue      // port VGA : 3 signaux de couleur
hs, vs              // port VGA: signaux de synchro. horizontale et verticale
ja_out[7..0]        // connecteur d'extension ja, signaux de sortie
jb_out[7..0]        // connecteur d'extension jb, signaux de sortie
jc_out[7..0]        // connecteur d'extension jc, signaux de sortie
jd_out[7..0]        // connecteur d'extension jd, signaux de sortie
ja_in[7..0]         // connecteur d'extension ja, signaux d'entrée
jb_in[7..0]         // connecteur d'extension jb, signaux d'entrée
jc_in[7..0]         // connecteur d'extension jc, signaux d'entrée
jd_in[7..0]         // connecteur d'extension jd, signaux d'entrée
```