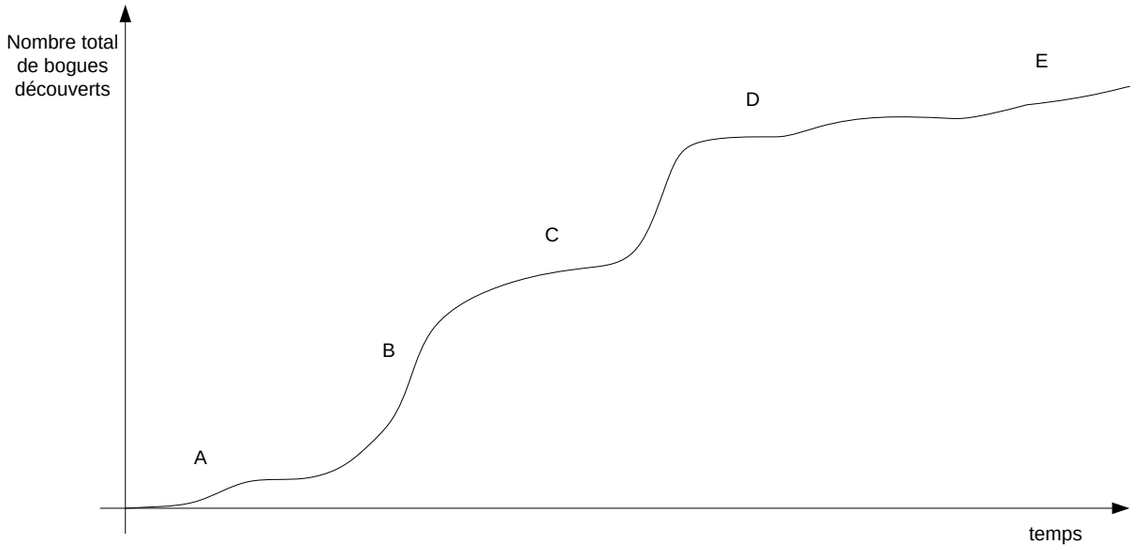
0701 Principes généraux

1. Vous travaillez sur un projet de conception d'un système numérique. Le nombre de bogues découverts en fonction du temps est montré par la courbe suivante. Donnez une explication possible pour chacune des régions identifiées sur la courbe.




0702 Tests exhaustifs

2. Considérez la déclaration d'entité en VHDL suivante pour une unité arithmétique d'un microprocesseur. Combien de vecteurs de test seraient nécessaires pour effectuer un test exhaustif de cette unité? Combien de temps serait nécessaire pour effectuer ce test à raison de $10^6$ tests par seconde? Montrez tous vos calculs.

```
entity unitearithmetique is
    generic (
        W : positive := 32 -- largeur des opérandes
    );
    port (
        A, B : in signed(W - 1 downto 0); -- les opérandes
        choix : in std_logic_vector(7 downto 0); -- le sélecteur d'opération
        F : out signed(W - 1 downto 0) -- le résultat
    );
end unitearithmetique;
```

3. Considérez le code VHDL suivant. Combien de vecteurs de test seraient nécessaires pour effectuer un test exhaustif ? Justifiez complètement votre réponse et montrez tous vos calculs.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity machineAEtats is
    port (
        reset, CLK : in STD_LOGIC;
        x : in STD_LOGIC_VECTOR(1 downto 0);
        sortie : out STD_LOGIC
    );
end machineAEtats;

architecture arch2 of machineAEtats is
type type_etat is (S1, S2, S3, S4);
signal etat : type_etat := S1;
begin

    process(etat)
    begin
        case etat is
            when S1 | S3 =>
                sortie <= '1';
            when S2 | S4 =>
                sortie <= '0';
        end case;
    end process;
```

```
process(CLK, reset) is
begin
    if (reset = '0') then
        etat <= S1;
    elsif (rising_edge(CLK)) then
        case etat is
            when S1 =>
                if x = "00" then
                    etat <= S3;
                else
                    etat <= S2;
                end if;
            when S2 =>
                    etat <= S1;
            when S3 =>
                if x = "11" then
                    etat <= S2;
                else
                    etat <= S4;
                end if;
            when S4 =>
                etat <= S1;
        end case;
    end if;
end process;

end arch2;
```

4. Considérez la déclaration d'entité en VHDL suivante pour un module qui trouve la valeur médiane de cinq nombres signés donnés en entrée : a, b, c, d et e. Ainsi, si `(a, b, c, d, e) = (+3, +8, +5, −7, +2)`, la sortie devrait être `m = +3`.

```
entity median5 is
generic( w : positive := 8 );
port(
a, b, c, d, e : in signed(w-1 downto 0);
m : out signed(w-1 downto 0)
);
end median5;
```

Combien de vecteurs de test sont nécessaires pour effectuer un test exhaustif de l'entité median5 ? Montrez tous vos calculs.

0703 Tests de boîte noire

5. Considérez la déclaration d'entité en VHDL suivante pour un module qui trouve la valeur médiane de cinq nombres signés donnés en entrée : a, b, c, d et e. Ainsi, si (a, b, c, d, e) = (+3, +8, +5, −7, +2), la sortie devrait être m = +3.

```
entity median5 is
generic( w : positive := 8 );
port(
a, b, c, d, e : in signed(w-1 downto 0);
m : out signed(w-1 downto 0)
);
end median5;
```

On propose les cinq classes suivantes pour chacune des entrées du circuit (on suppose $w \geq 3$).

$\{ \{-2^{w-1}, -2^{w-1}+1, \ldots -2^{w-2}-1\}, \{-2^{w-2}, -2^{w-2}+1, \ldots -2^{w-3}-1\}, \{-2^{w-3}, \ldots +2^{w-3}-1\}, \{+2^{w-3}, +2^{w-2}+1, \ldots +2^{w-2}-1\}, \{+2^{w-2}, +2^{w-2}+1, \ldots +2^{w-1}-1\} \}$

a. Quelle est la taille minimale de l'ensemble de vecteurs de test pour effectuer un test faible sur l'entité?
b. Quelle est la taille minimale de l'ensemble de vecteurs de test pour effectuer un test fort sur l'entité?

6. Un système de contrôle du chauffage d'une maison montréalaise en hiver a trois entrées: la température actuelle (au degré près), l'heure de la journée (à l'heure près), et le jour (lundi à dimanche). La sortie du système est une commande pour activer ou non (1 ou 0) le chauffage. On suppose que la température extérieure est toujours inférieure à 0 C et que la maison se refroidit quand le chauffage ne fonctionne pas.

Le tableau suivant résume les températures désirées :

| Jour | Heure | Température désirée |
|---|---|---|
| Tous les jours | De 0 h à 6 h et de 22 h à 0 h | 19° C |
| Lundi à vendredi | De 6 h à 8 h et de 16 h à 22 h | 22° C |
| Lundi à vendredi | De 8 h à 16 h | 17° C |
| Samedi et dimanche | De 6 h à 9 h | 19° C |
| Samedi et dimanche | De 9 h à 22 h | 22° C |

a. Proposez un partitionnement en classes pour chacune des trois entrées.
b. Donnez un ensemble de vecteurs de tests pour un test faible selon votre partitionnement proposé.
c. Combien de vecteurs de tests sont nécessaires pour effectuer un test fort ?

7. Donner le code VHDL pour générer un nombre aléatoire avec une distribution uniforme :
a. réel, dans l'intervalle [5.0, 50.0[
b. entier, dans l'intervalle [-10, 10[
c. réel, dans l'intervalle $[-\pi/2, \pi/2[$

0704 Tests de boîte blanche

8. Expliquez pourquoi un ensemble de vecteurs de tests donnant une couverture de code de 100% n'est pas nécessairement suffisant pour vérifier adéquatement un module.

9. Considérez le code VHDL suivant pour un cadenas numérique. Donnez un ensemble de vecteurs de test permettant d'atteindre une couverture de code de 100%.

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

entity cadenas is
   port (
      reset, clk : in STD_LOGIC;
      numero : in unsigned(7 downto 0);
      ouvrir : out STD_LOGIC
   );
end cadenas;

architecture arch of cadenas is
type combinaison is array (natural range <>) of natural range 0 to 255;
constant lacombine: combinaison := (10, 11, 12, 13, 14, 15);
type type_etat is (aucun, n1ok, n2ok, n3ok, n4ok, n5ok, debarre);
signal etat : type_etat := aucun;
begin

   ouvrir <= '1' when etat = debarre else '0';

   process(clk, reset) is
   begin
       if (reset = '0') then
          etat <= aucun;
      elsif (rising_edge(clk)) then
          case etat is
             when aucun =>
                if numero = lacombine(0) then etat <= n1ok;
                else etat <= aucun;
                end if;
             when n1ok =>
                if numero = lacombine(1) then etat <= n2ok;
                elsif numero = lacombine(0) then etat <= n1ok;
                else etat <= aucun;
                end if;
             when n2ok =>
                if numero = lacombine(2) then etat <= n3ok;
                elsif numero = lacombine(0) then etat <= n1ok;
                else etat <= aucun;
                end if;
             when n3ok =>
                if numero = lacombine(3) then etat <= n4ok;
                elsif numero = lacombine(0) then etat <= n1ok;
                else etat <= aucun;
                end if;
             when n4ok =>
                if numero = lacombine(4) then etat <= n5ok;
                elsif numero = lacombine(0) then etat <= n1ok;
                else etat <= aucun;
                end if;
             when n5ok =>
                if numero = lacombine(5) then etat <= debarre;
                elsif numero = lacombine(0) then etat <= n1ok;
                else etat <= aucun;
                end if;
             when debarre =>
                etat <= aucun;
             when others =>
                etat <= aucun;
          end case;
      end if;
   end process;
end arch;
```

Solutions

1. Réponses possibles

A : début des tests; B : amélioration des cas de test; C : stabilisation; D : plateau de performance; E : découverte de nouveaux problèmes … le système n'est pas prêt.

2. Réponse

Pour W = 32, il y a 2 × 32 + 8 = 72 ports d'entrée. Il faudrait donc $2^{72} = 4.7 \times 10^{21}$ vecteurs de test.

À raison de $10^6$ tests par seconde, il faudrait environ $4.7 \times 10^{15}$ secondes, soit $150 \times 10^6$ ans.

3. Solution

4 états × $2^2$ combinaisons de l'entrée *x*, donc 16 vecteurs de test.

Ce décompte suppose qu'on puisse se placer dans n'importe quel état pour appliquer les entrées *x*. En pratique cependant, il faut appliquer des entrées supplémentaires pour se rendre dans chacun des états.

4. Solution

Réponse. Il y a 5 entrées binaires de W bits. Un test exhaustif demanderait donc $2^{5W}$ vecteurs de test. Pour W = 8, on aurait $2^{40} \approx 1 \times 10^{12}$ vecteurs de test.

5. Solution :

a. Pour un test faible, chaque classe doit contribuer au moins une fois à un vecteur de test. Le nombre minimal de vecteurs de test est égal au nombre maximal de classes pour chacune des entrées. Dans le présent problème, toutes les entrées ont les mêmes cinq classes, donc un test faible serait composé de 5 vecteurs de test.

b. Pour un test fort, un élément de chaque classe doit être choisi en combinaison avec un élément de chacune des autres classes. Le nombre minimal de vecteurs de test est égal au produit du nombre de classes pour chaque entrée. Dans le présent problème, il y a cinq classes pour chacune des cinq entrées, donc il faut $5^5 = 3125$ vecteurs de test.

Note : La valeur de W est sans importance. Quand on choisit des vecteurs de test en fonction du partitionnement en classe, chaque valeur dans une classe est réputée être représentative pour toutes les autres valeurs dans cette même classe. Donc qu'il y en ait 1000 ou une seule ne change rien.

6. Solution :

a. Classes :

Le principe du partitionnement en classe est qu'un élément d'une classe est réputé être représentatif de tous les éléments de sa classe.

On constate que si le système fonctionne correctement le mardi, on s'attend à ce que ce soit la même chose pour les autres jours de la semaine. S'il fonctionne le dimanche, ce sera pareil pour le samedi. On obtient alors :

Pour les jours : {lundi, mardi, mercredi, jeudi, vendredi} et {samedi, dimanche}

Si le système fonctionne correctement à 2 h du matin, on s'attend à ce que ce soit ok toute la nuit, etc. La difficulté pour le partitionnement des heures provient du chevauchement des journées de semaine (du lundi à vendredi) et de fin de semaine (samedi et dimanche). Ainsi, il faut une classe séparée pour {6 h, 7 h} et pour {8 h}. En effet, la température commandée à 8 h est différente de celle à 7 h du lundi au vendredi, donc il faut deux classes séparées. De façon similaire, la température commandée à 9 h est différente de la température commandée à 8 h les samedis et dimanche, donc il faut deux classes séparées. On obtient alors :

Pour les heures: {0, 1, 2, 3, 4, 5}, {6, 7}, {8}, {9, 10, 11, 12, 13, 14, 15}, {16, 17, 18, 19, 20, 21}, {22, 23}.

Pour les températures: {..., 15, 16}, {17, 18}, {19, 20, 21}, {22, 23, 24, 25, ...}

b. {jour, heure, température actuelle}:

{lundi, 0, 15}, {lundi, 6, 17}, {lundi, 8, 19}, {lundi, 10, 22}, {lundi, 16, 15}, {samedi, 22, 15}

c. Il faudrait 2 × 6 × 4 = 48 vecteurs de test.

7. Réponses basées sur le code montré dans les diapositives.

a.

```
uniform(seed1, seed2, aleatoire); -- 0.0 < aleatoire < 1.0
aleatoire := 5.0 + 45.0 * aleatoire;
-- attention, la quantité 5.0 n'est pas dans l'intervalle pouvant être atteint
```

b.

```
uniform(seed1, seed2, aleatoire); -- 0.0 < aleatoire < 1.0
aleatoire := -10.0 + 20.0 * aleatoire;
t := integer(aleatoire);
```

c.

```
uniform(seed1, seed2, aleatoire); -- 0.0 < aleatoire < 1.0
aleatoire := -MATH_PI / 2.0 + MATH_PI * aleatoire;
-- attention, la quantité -π/2 n'est pas dans l'intervalle pouvant être atteint
```

8. Réponse : La couverture de code n'indique que si certaines situations ont été exercées ou non, sans égard à la fonctionnalité du système.

La métrique de couverture de code complète les autres types de tests et donne une certaine assurance au concepteur que le circuit est bien vérifié. Mais une couverture de 100% pour un ensemble de vecteurs de test ne garantit pas que le circuit rencontre toutes ses spécifications, uniquement que chaque énoncé de la description du modèle a été exécuté au moins une fois. Et comme pour les autres stratégies de sélection de vecteurs de test, la sortie doit être correcte pour chaque vecteur appliqué.

En guise d'exemple, imaginons un module qui doive faire soit l'addition ou la soustraction de deux nombres de 8 bits. Supposons que la partie soustraction n'a pas été incluse dans le modèle. Supposons qu'on construise un ensemble de vecteurs de test qui vérifie l'addition correctement et qui donne 100% de couverture de code. Tout semble correct, sauf que la partie de la spécification concernant la soustraction n'a été ni incluse dans la description ni vérifiée.

9. L'ensemble de vecteurs de test suivant permet de traverser chacun des états et chacune des transitions entre les états de la machine à états au moins une fois.

On ne considère pas dans ce genre de question ni l'horloge ni le signal de réinitialisation.

Il est utile de considérer le diagramme d'états pour énumérer les vecteurs : on doit s'assurer de traverser chaque transition d'état.

```
constant vecteurs : tableau := (
10, 99,
10, 11, 99,
10, 11, 12, 99,
10, 11, 12, 13, 99,
10, 11, 12, 13, 14, 99,
10, 11, 12, 13, 14, 15, 99,
10, 10,
10, 11, 10,
10, 11, 12, 10,
10, 11, 12, 13, 10,
10, 11, 12, 13, 14, 10,
10, 11, 12, 13, 14, 15, 99
);
```