Nom : _____ Matricule : _____

INF3500 : Conception et réalisation de systèmes numériques

Examen intra #2 – lundi 9 novembre 2015

Durée: 1 heure. Documentation: Une feuille recto verso 8.5"×11" ou A4 permise.
Pondération: 10%. Calculatrice: Programmable permise.
Directives particulières:

- Ordinateurs interdits. Appareils mobiles interdits.
- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Répondre sur le questionnaire et le remettre.
- Ne posez pas de questions. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites.

Question 1. (1 point)

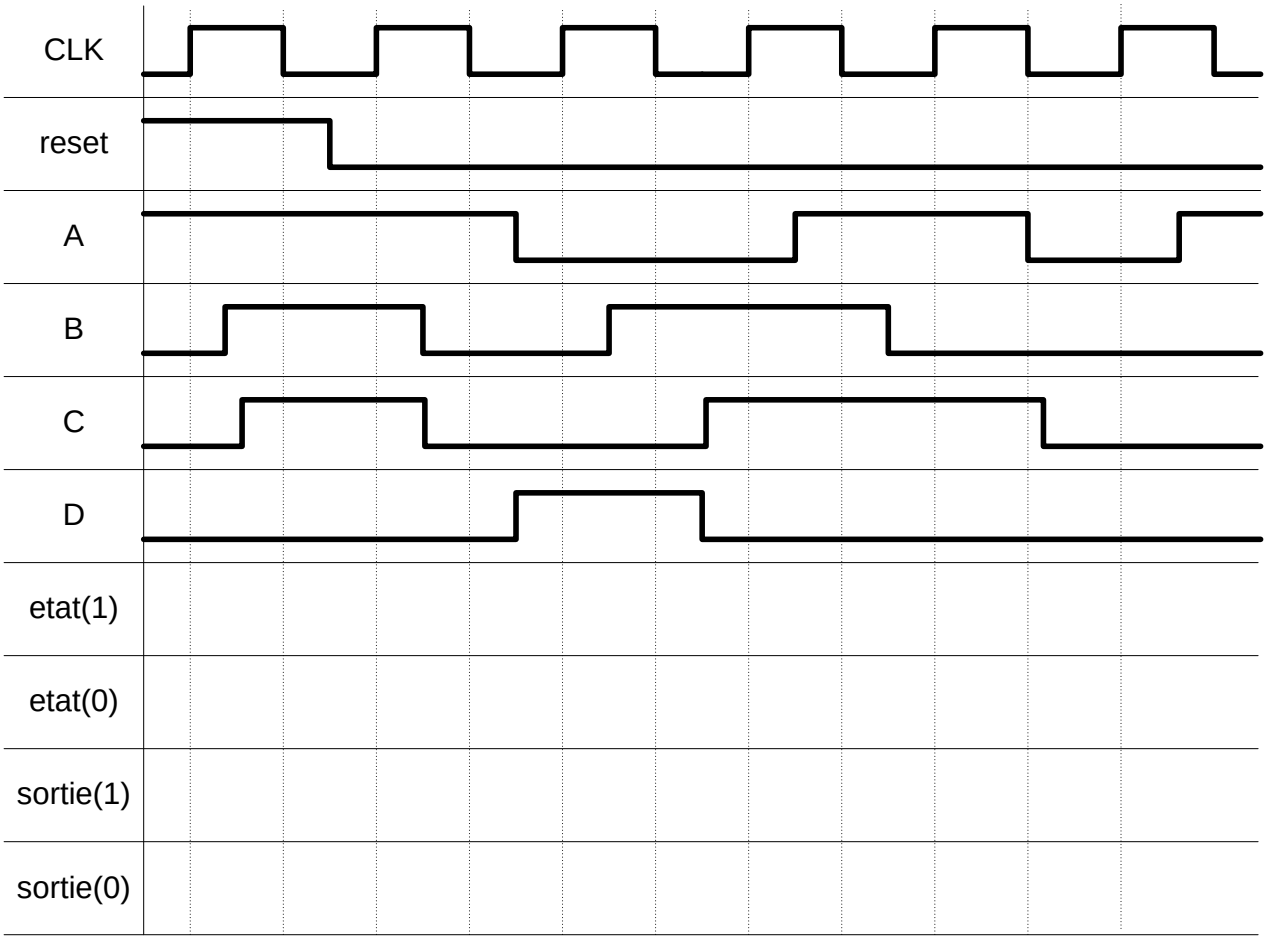
a. (0.5 point) Expliquez pourquoi un ensemble de vecteurs de tests donnant une couverture de code de 100% n'est pas nécessairement suffisant pour vérifier adéquatement un module.

b. (0.5 point) Pour la technique de vérification par le partitionnement en classes, expliquez la différence entre un ensemble de vecteurs de test « fort » et un ensemble de vecteurs de tests « faible ».

Question 2. (2 point)

Considérez le code VHDL suivant. Complétez le chronogramme suivant pour ce module.

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

entity monModule5 is
    port (
        reset, clk: in std_logic;
        A, B, C, D : in std_logic;
        sortie : out std_logic_vector(1 downto 0)
    );
end monModule5;

architecture arch of monModule5 is
signal etat : std_logic_vector(1 downto 0);
begin

process(CLK, reset) is
    begin
        if (reset = '1') then
            etat <= "00";
        elsif (rising_edge(CLK)) then
            etat(1) <= (A and B and C) or D;
            etat(0) <= A and not(D);
            sortie(1) <= B and etat(0);
        end if;
    end process;

    sortie(0) <= '0' when
        (etat = "01" or etat = "10") else '1';

end arch;
```

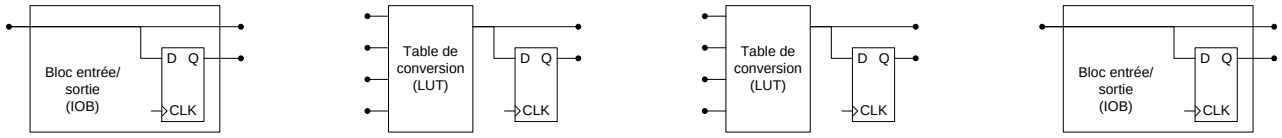


Question 3. (3 points)

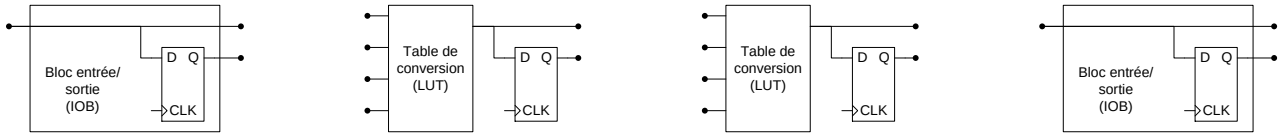
Considérez le code VHDL et le modèle de FPGA suivants. Montrez, sur le modèle du FPGA, un résultat possible de la synthèse et de l'implémentation de ce code. Indiquez directement sur le dessin où chaque signal et port de sortie se situe ainsi que les interconnexions entre les blocs. Les interconnexions peuvent contourner les blocs. Indiquez quand une bascule doit être utilisée. Indiquez clairement, par une équation ou une porte logique, la fonction logique réalisée par chaque LUT que vous utilisez.

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

entity module11 is
   port (
      clk, A, B, C, D, E: in std_logic;
      X, Y, Z: out std_logic
   );
end module11;

architecture arch of module11 is

signal Q : std_logic_vector(1 downto 0) := "10";
signal R, S, T : std_logic := '1';

begin

   R <= '1' when A & B & C = "011" else '0';
   X <= T or Q(0);
   Z <= R or S or T;

   process(clk) is
   begin
      if rising_edge(CLK) then
         Q(1) <= Q(0) and R and S;
         Q(0) <= R xor S;
         Y <= Q(1) and Q(0);
      end if;
   end process;

   process(D, E)
   variable V : std_logic := '0';
   begin
      V := D;
      S <= V or E;
      V := not(D);
      T <= V and E;
   end process;

end arch;
```

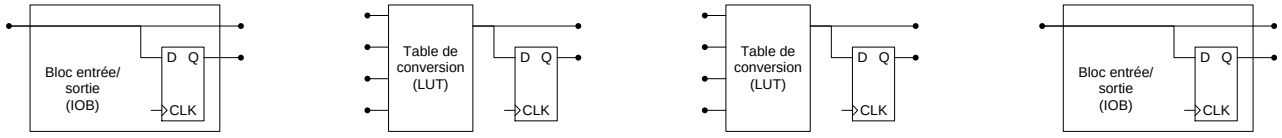
A



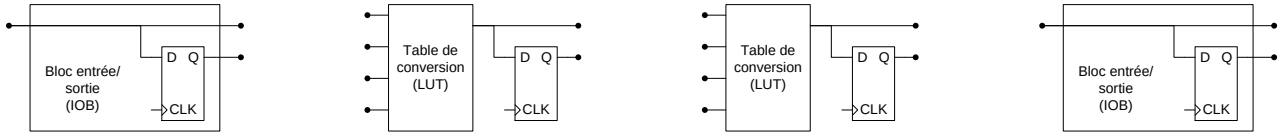
B



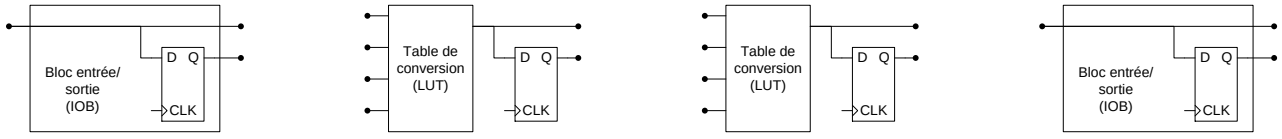
C



D

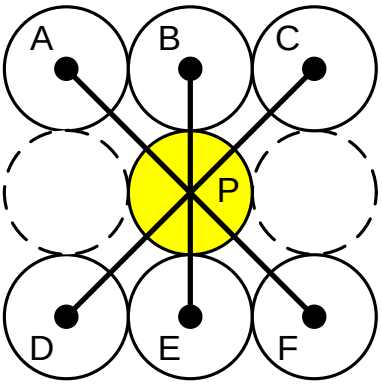


E



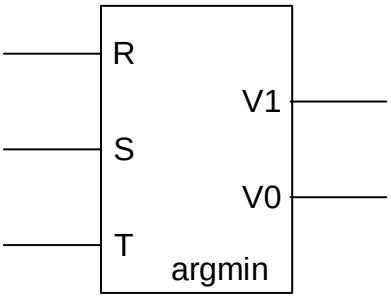
Question 4. (4 points)

L'algorithme ELA (Edge-based Line Average) double la résolution verticale d'une image en créant une nouvelle ligne de pixels entre chaque paire de lignes existantes. Pour chaque paire de lignes, l'algorithme prend trois pixels de la ligne du haut et trois pixels de la ligne du bas, et calcule un pixel de la nouvelle ligne intermédiaire. Les pixels sont représentés en tons de gris sur 8 bits.



L'algorithme accepte en entrée les valeurs des six pixels A, B, C, D, E et F, et produit en sortie le pixel P. Il calcule trois différences absolues : |A – F|, |B – E| et |C – D|. La plus petite de ces trois valeurs indique la direction d'interpolation. La valeur de P est alors donnée par la moyenne des deux pixels le long de la direction d'interpolation. Par exemple, si |A – F| a la plus petite valeur, alors P = (A + F) / 2. Si |B – E| a la plus petite valeur, alors P = (B + E) / 2.

Donnez le diagramme d'un chemin des données pour implémenter l'algorithme ELA. Supposez que les entrées sont disponibles dans des registres. La sortie doit aussi être placée dans un registre.



Vous pouvez utiliser les opérations arithmétiques et logiques, le décalage, des multiplexeurs, et la comparaison de grandeurs. Vous pouvez aussi utiliser un bloc 'argmin'. La fonction 'argmin' indique laquelle de ses trois entrées R, S, et T est la plus petite. Si $R \leq S$ et $R \leq T$, alors V = "00". Si $S \leq R$ et $S \leq T$, alors V = "01". Si $T \leq S$ et $T \leq R$, alors V = "10". Le cas V = "11" n'est pas possible.

Solutions

#1

a. 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.

b. 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.

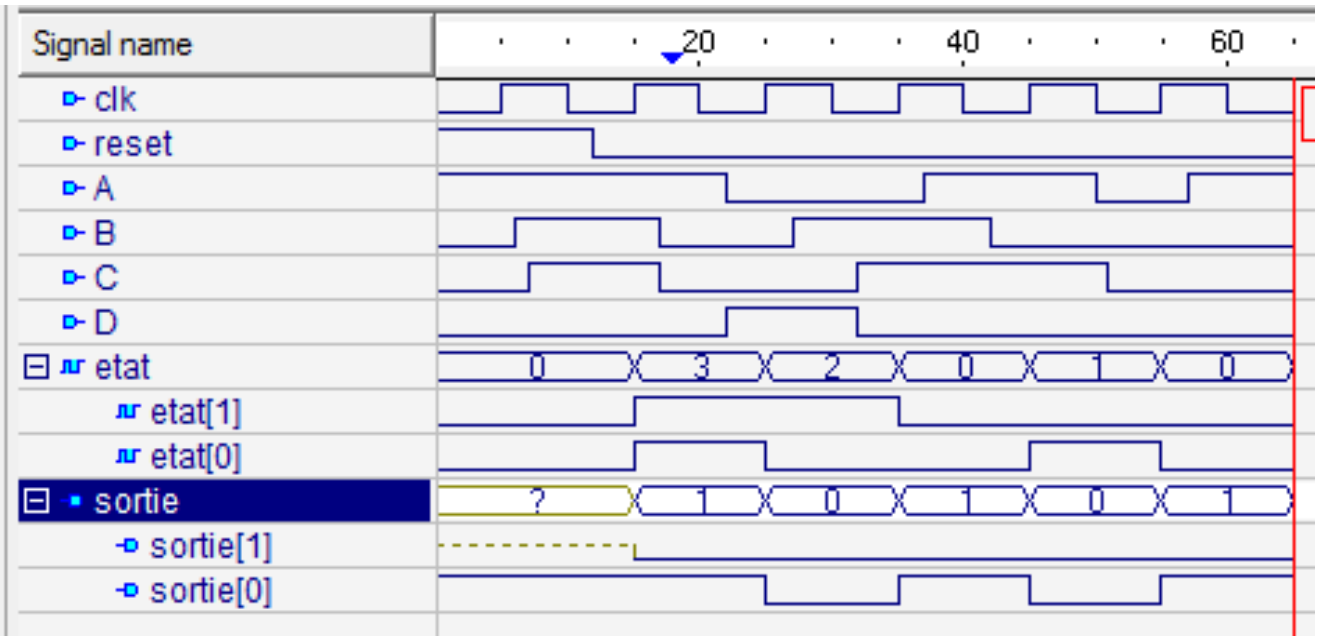
Pour un test fort, un élément de chaque classe doit être choisi en combinaison avec un élément de chacune des autres classes. Le nombre minimal de vecteurs de test est égal au produit du nombre de classes pour chaque entrée.

#2 Réponse

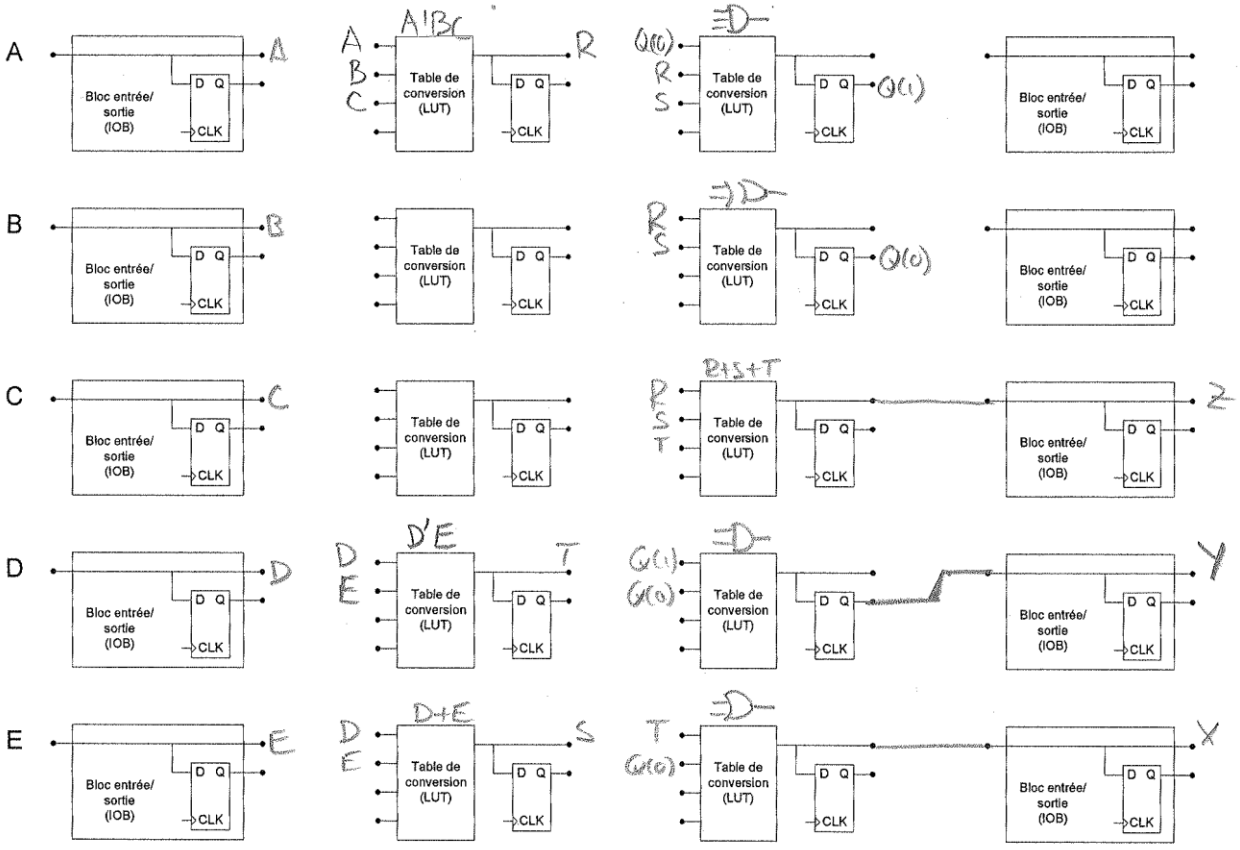
Le port sortie(1) est la sortie d'un registre, donc synchronisé avec l'horloge. Il n'est pas réinitialisé dans le code, donc on n'en connait pas la valeur avant la première transition positive du signal d'horloge qui suit le moment où reset n'a plus la valeur 1.

Le port sortie(1) est une fonction de etat(1) et de B. Pour la deuxième transition d'horloge, on voit que etat(1) change en même temps que l'horloge. En pratique, ce changement se ferait un court moment *après* le signal d'horloge, parce que la bascule qui contient etat(1) a besoin d'un peu de temps pour réagir. Donc quand on évalue sortie(1), on doit prendre la valeur de etat(1) qui précède immédiatement la transition d'horloge, c'est-à-dire '0' dans ce cas-ci, ce qui veut dire que sortie(1) reste à '0' au deuxième front d'horloge.

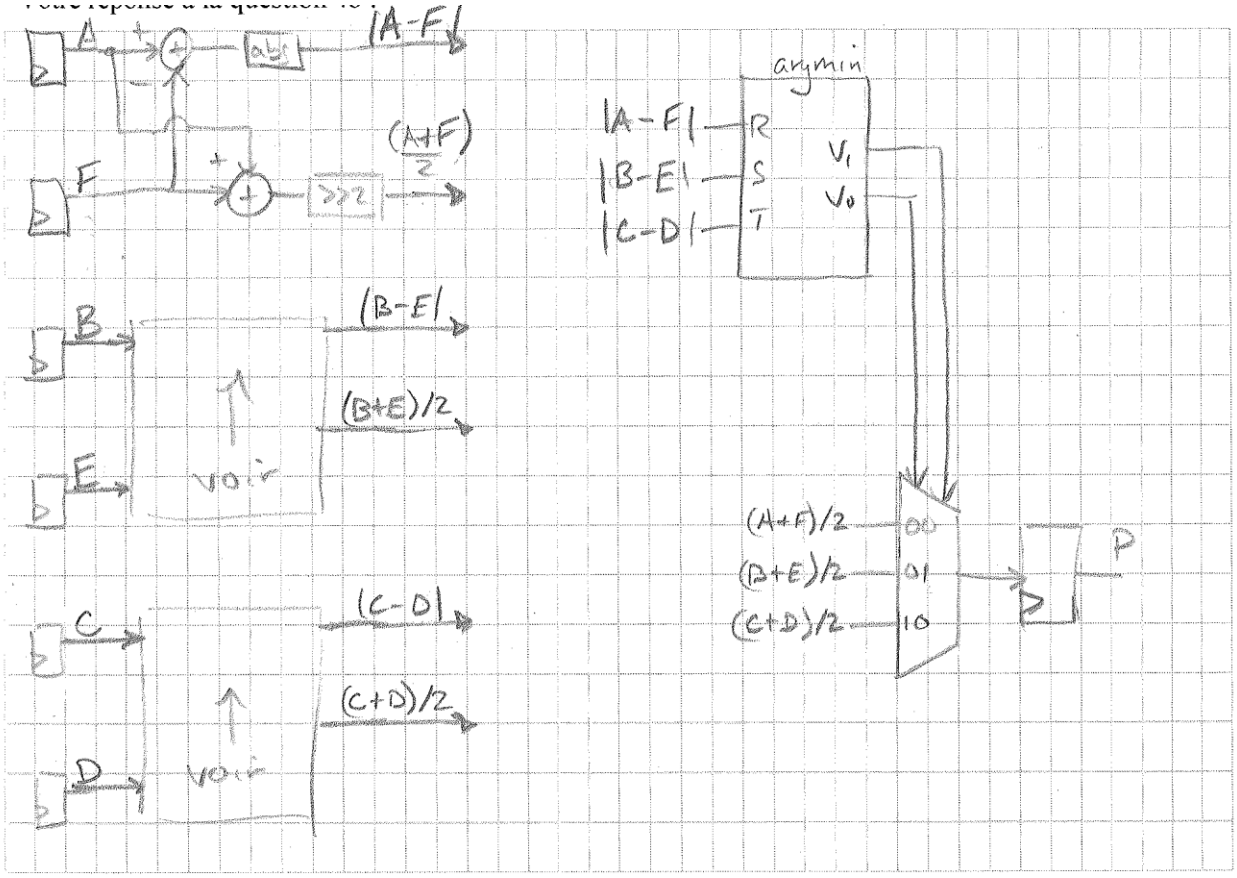
On observe que le port sortie(0) est synchronisé avec l'horloge, mais ce n'est pas parce qu'il passe par une bascule. C'est une sortie de Moore, c'est-à-dire une fonction uniquement de l'état interne du circuit.



#3 Réponse


A
Bloc entrée/ sortie (IOB)
D Q
CLK
A
A!BC
A
B
C
Table de conversion (LUT)
R
Q(1)
R
S
B
Q(0)
C
R+S+T
P
S
T
Z
D
D'E
D
E
T
G(1)
G(0)
Y
E
D+E
S
T
X

#4 Solution


A
F
B
E
C
D
|A-F|
(A+F)/2
|B-E|
(B+E)/2
|C-D|
(C+D)/2
voir
voir
argmin
|A-F|
|B-E|
|C-D|
R
S
T
V1
V0
(A+F)/2
(B+E)/2
(C+D)/2
00
01
10
P