

# Formation programmation FPGA

Gwenhaël Goavec-Merou<sup>1 2</sup>

<sup>1</sup>FEMTO-ST, Département Temps-Fréquence , 26, chemin de l'Epitaphe, 25030  
Besançon, France

<sup>2</sup>ARMadeus Systems 40, rue Marc Seguin, 68200 Mulhouse, France

30 aout 2010

## Objectifs

Plan

FPGA

etude formules

Choix

VHDL

Outils

carte  
ARMadeusimplemen-  
tations

Pipeline

Conclusion

- familiarisation avec les FPGAs et le VHDL,
- compréhension des étapes d'implémentation d'un algorithme,
- découverte de l'outil ISE de Xilinx,
- Présentation d'une solution de simulation,
- mise en pratique,

- théorie :
  - ① présentation du FPGA,
  - ② Étude de l'algorithme à mettre en œuvre,
  - ③ présentation du langage VHDL,
- Pause / changement de lieu
- Pratique :
  - ① présentation des outils :
    - ISE pour la synthèse et la génération des binaires.
    - GHDL pour la simulation,
    - validation In-Situ,
  - ② implémentation de l'algorithme,
  - ③ bilan sur le travail sur FPGA.

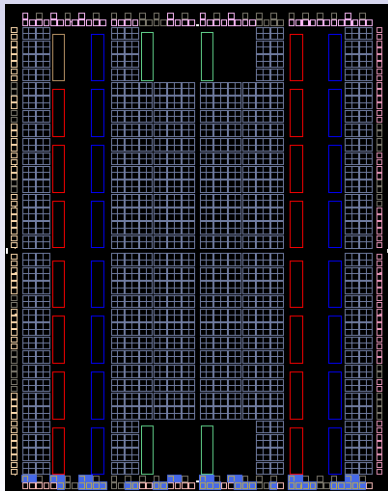
## Composant vierge :

- pas de fonctionnement par défaut,
- pas de routage.
- possibilité de traitement de type :
  - séquentiel,
  - pipeline,
  - parallèle.

## Caractéristiques :

- 100MHz,
- 200K portes,
- 16 multiplieurs (18bits entrées, 36 bits sortie),
- 16 RAMs (16Kb/RAM).

## FPGA Spartan3A



Objectifs

Plan

FPGA

etude formules

Choix

VHDL

Outils

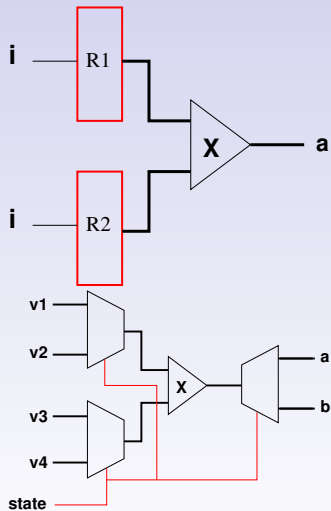
carte

ARMadeus

implemen-  
tations

Pipeline

Conclusion

$$a = r1[i] * r2[i];$$
$$a = v1 * v2;$$
$$b = v3 * v4;$$


S'approprier l'algorithme pour :

- définir les interdépendances (récursivité),
- distinguer les traitements pouvant être réalisés en parallèle.

un filtre FIR (Finite Impulse Response).

$$y[n] = \sum_{i=0}^N b_i * x[n - i]$$

- toutes les valeurs de  $y$  sont indépendantes,
- $b_i * x[n - i]$  indépendant des autres produits,

Données :

- $b_i$  constantes,
- $x[n - i]$  issues de l'acquisition.

Implémentation sur processeur  
générique

Objectifs

Plan

FPGA

étude formules

Choix

VHDL

Outils

carte

ARMadeus

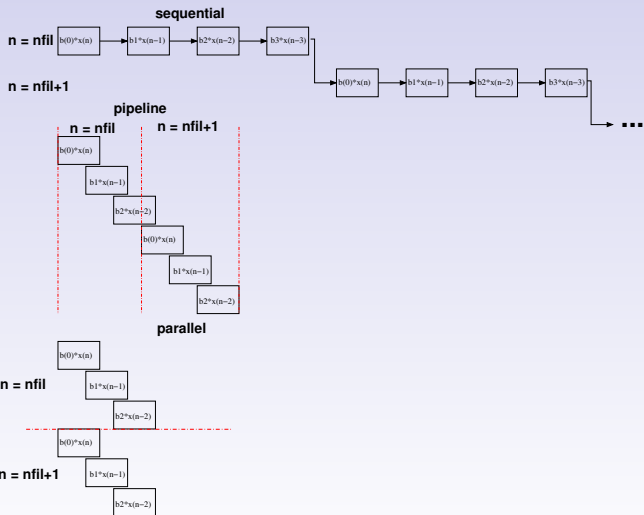
implemen-  
tations

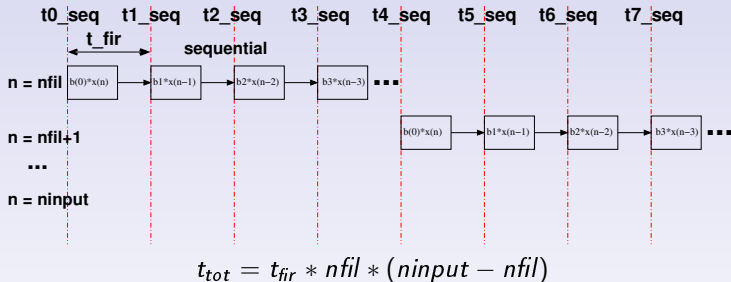
Pipeline

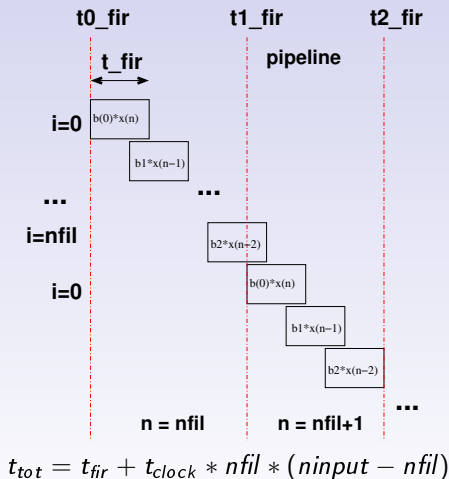
Conclusion

```
unsigned short k , j ;  
long s=0;  
for ( k=0;k<nin -nfil ; k++) {  
    s=0;  
    for ( j=0; j<nfil ; j++)  
        s+=(((fil [j]*in [nfil -1+k-j] )));  
    sortie [k]=s;  
}
```

## Solutions possible d'implémentation







Objectifs

Plan

FPGA

étude formules

Choix

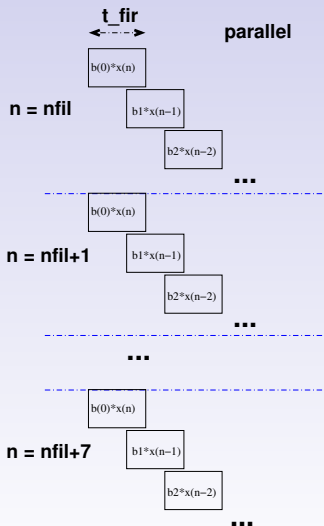
VHDL

Outils

carte  
ARMadeusimplemen-  
tations

Pipeline

Conclusion



$$t_{tot} = t_{fir} + t_{clock} * nfil * ((ninput - nfil) / nflow)$$

Objectifs

Plan

FPGA

étude formules

Choix

**VHDL**

Outils

carte  
ARMAdeusimplemen-  
tations

Pipeline

Conclusion

- langage de description du circuit.
  - structurelle,
  - comportementale,
  - "flot de données".
- traitement concurrent

VHDL : Structure entité et  
instanciation

Objectifs

Plan

FPGA

etude formules

Choix

VHDL

Outils

carte

ARMadeus

implemen-  
tations

Pipeline

Conclusion

## Entity :

```
Entity my_entity is
  generic (id : natural := 1);
  port (
    reset      : in  std_logic;
    clk        : in  std_logic
  );
end entity my_entity;
```

## Architecture :

```
Architecture my_entity_arch of my_entity is
  signal registre : std_logic_vector(3 downto 0);
  signal write_rise : std_logic;
begin
end architecture my_entity_arch;
```

## Instanciation :

```
my_entity00 : entity work.my_entity
  generic map (id => id)
  port map (
    reset => gls_reset ,
    clk => gls_clk
  );
```

```
Entity compteur is
  generic (id : natural := 1);
  port (
    reset : in std_logic;
    clk   : in std_logic;
    compteur : out std_logic_vector(15 downto 0);
    in_sig  : in std_logic;
    si     : out std_logic_vector(15 downto 0)
  );
end entity compteur;
```

```

Architecture compteur_beh of compteur is
    signal cpt : std_logic_vector(15 downto 0);
    signal s_h : std_logic_vector(7 downto 0);
    signal s_l : std_logic_vector(7 downto 0);
begin
    si <= s_h&s_l;
    compteur <= cpt;
    p_compteur : process(clk_reset)
    begin
        if reset = '1' then
            cpt <= (others => '0');
        elsif rising_edge(clk) then
            cpt <= std_logic_vector(unsigned(cpt)+1);
        end if;
    end process p_compteur;

    p_async : process(in_sig)
    begin
        if in_sig = '1' then
            s_h <= cpt(15 downto 8);
            s_l <= s_l;
        else
            s_l <= cpt(7 downto 0);
            s_h <= s_h;
        end if;
    end process p_async;
end architecture compteur_beh;

```

# VHDL : difficultés

- erreurs dans les choix,
- mauvaise prise en compte des contraintes du langage et du FPGA,
- Problèmes de timing et de ressources.

Objectifs

Plan

FPGA

etude formules

Choix

**VHDL**

Outils

carte

ARMAdeus

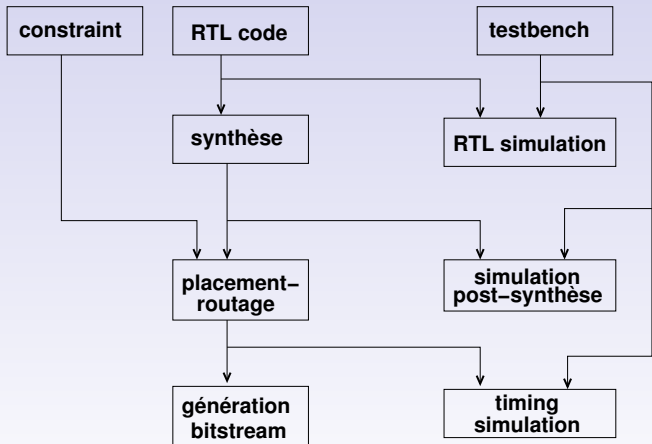
implemen-  
tations

Pipeline

Conclusion

Pause

La suite à l'ENSMM.



G.  
Goavec-Merou

Objectifs

Plan

FPGA

étude formules

Choix

VHDL

Outils

carte  
ARMadeusimplemen-  
tations

Pipeline

Conclusion

The screenshot shows the Xilinx ISE Project Navigator interface. The main window is titled "Design Summary (Synthesized)" and displays the following information:

**multi Project Status (08/23/2010 - 14:15:21)**

|                  |                           |                       |                     |
|------------------|---------------------------|-----------------------|---------------------|
| Project File:    | multi.isc                 | Implementation State: | Synthesized         |
| Module Name:     | top_multi                 | Errors:               | No Errors           |
| Target Device:   | xc3s200a-ft256            | Warnings:             | 23 Warnings (0 new) |
| Product Version: | ISE 11.5                  | Routing Results:      |                     |
| Design Goal:     | Balanced                  | Timing Constraints:   |                     |
| Design Strategy: | Xilinx Default (unlocked) | Final Timing Score:   |                     |

**Device Utilization Summary (estimated values)**

| Logic Utilization          | Used | Available | Utilization |
|----------------------------|------|-----------|-------------|
| Number of Slices           | 170  | 1792      | 9%          |
| Number of Slice Flip Flops | 226  | 3584      | 6%          |
| Number of 4 input LUTs     | 270  | 3584      | 7%          |
| Number of bonded IOBs      | 32   | 195       | 16%         |
| Number of BRAMs            | 3    | 16        | 18%         |
| Number of MULT18x18SIOs    | 1    | 16        | 6%          |

The console window at the bottom shows the message: "Process 'Synthesis' completed successfully".

Objectifs

Plan

FPGA

étude formules

Choix

VHDL

Outils

carte

ARMadeus

implemen-  
tations

Pipeline

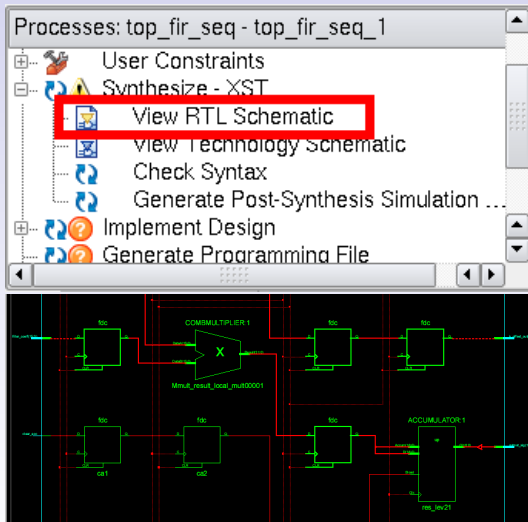
Conclusion



Permet de générer :

- le circuit logique et son schéma,
- la liste des erreurs et des avertissements,
- permet de générer le fichier de simulation de post-synthèse

Visualisation du circuit équivalent :



# ISE : Placement et routage

Objectifs

Plan

FPGA

étude formules

Choix

VHDL

Outils

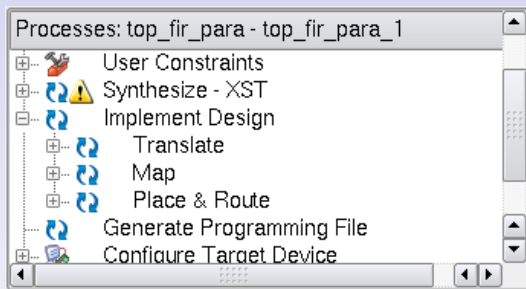
carte

ARMAdeus

implemen-  
tations

Pipeline

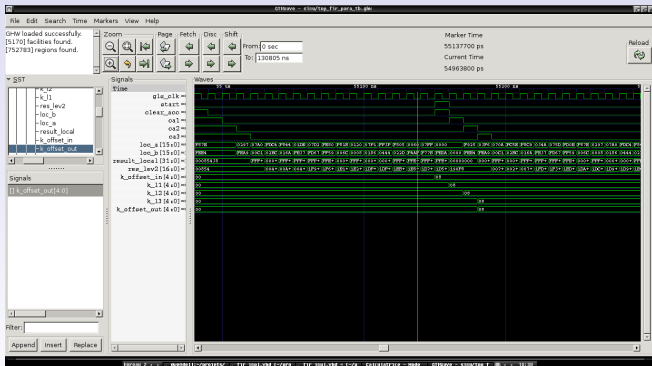
Conclusion

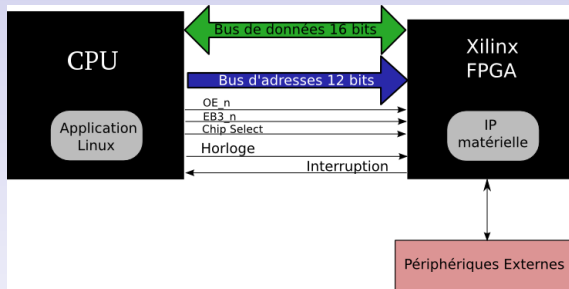


- Gère le placement dans le FPGA,
- Vérifie les contraintes temporels,
- Génère le fichier de post-placement pour la simulation

Permet :

- valider le fonctionnement (bug, comportements erronés)
- étudier le comportement du composant.





- bus de données 16bits,
- bus d'adresses 12bits,
- horloge commune,
- une ligne d'interruption

Objectifs

Plan

FPGA

étude formules

Choix

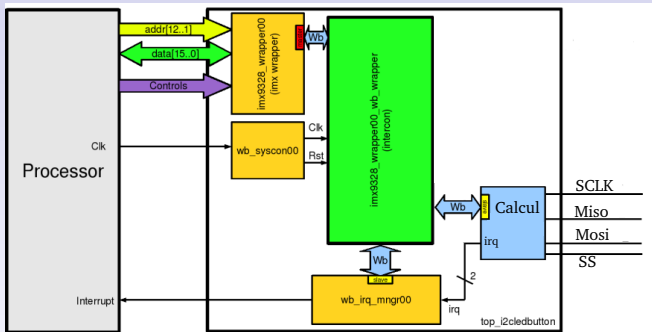
VHDL

Outils

carte  
ARMadeusimplemen-  
tations

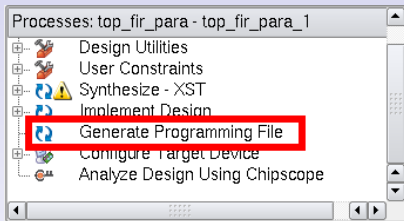
Pipeline

Conclusion



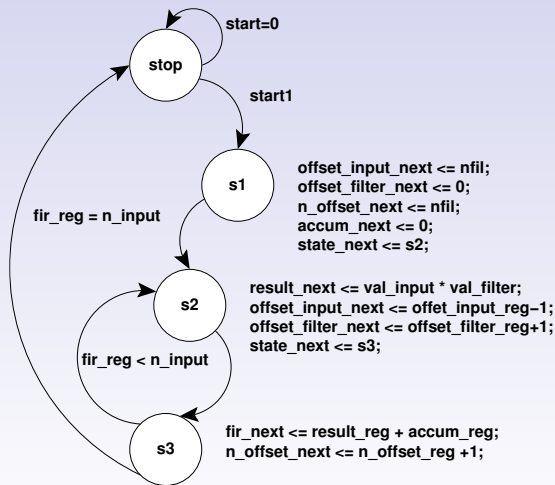
- Permet une communication simple entre le processeur et le FPGA,
- automatiquement généré par POD.

## Génération du bitstream :



- le fichier doit être copié sur la carte,  
cp objs/top\_fir\_xx.bit /opt/local/bin
- modprobe fpgaloader pour charger le driver,
- dd if=top\_fir.bit of=/dev/fpgaloader pour flasher

## Utilisation d'une Machine d'état fini.



```
register_process : process (clk , reset )
begin
  if (reset = '1') then
    output_offset_reg <= (others => '0');
    filter_offset_reg <= (others => '0');
    accum_reg <= (others => '0');
    result_l1_reg <= (others => '0');
    [...]
    output_we_reg <= '0';
  elsif rising_edge(clk) then
    output_offset_reg <= output_offset_next;
    filter_offset_reg <= filter_offset_next;
    accum_reg <= accum_next;
    result_l1_reg <= result_l1_next;
    [...]
    output_we_reg <= output_we_next;
  end if;
end process register_process;
```

```

state_process : process (state_reg, [...],
                        accum_reg, start, result_l1_reg)
begin
    output_sig_next <= output_sig_reg;
    n_next <= n_reg;
    result_l1_next <= result_l1_reg;
    [...]
    output_we_next <= output_we_reg;
    case state_reg is
    when state_stop =>
        output_we_next <= '0';
        accum_next <= (others => '0');
        if start = '1' then
            filter_offset_next <= ZERO(9 downto 0);
            output_offset_next <= ZERO(9 downto 0);
            state_next <= state_s1;
        end if;
    when state_s1 =>
        output_we_next <= '0';
        result_l1_next <= std_logic_vector(signed(input_sig)
            * signed(filter_ccoeff));
        state_next <= state_s2;
    when state_s2 =>
        state_next <= state_s1;
        accum_next <= std_logic_vector(signed(
            result_l1_reg(24 downto 8)) +
            signed(accum_reg));
        if unsigned(filter_offset_reg) = NBFIL then
            output_we_next <= '1';
            accum_next <= (others => '0');

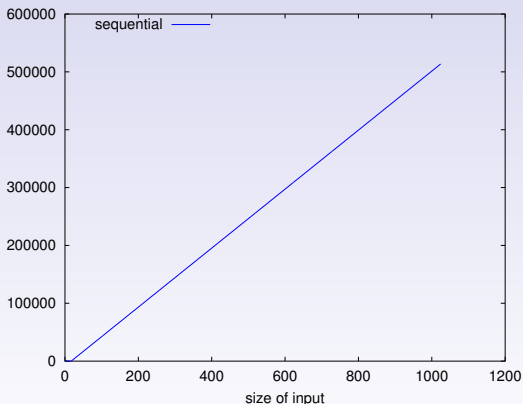
```

## Résultat Séquentiel

Taille maximale : 1024 données issues de l'acquisition.

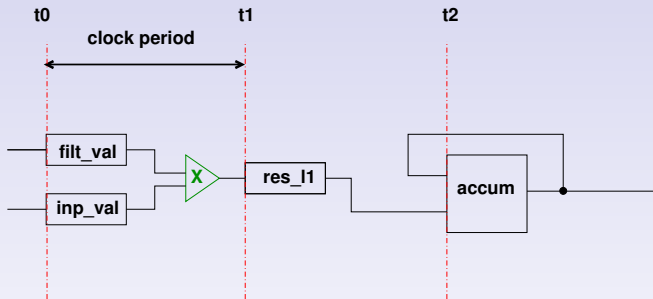
Usage : 7% des ressources, 3 RAMs & 1 multiplieur.

Temps de calcul : 7680ns pour 32 valeurs et 16 coefficients de filtre



$$nfil = 32, 1 < n < 1024$$

$$t_{tot} = t_{fir} * nfil * (ninput - nfil)$$



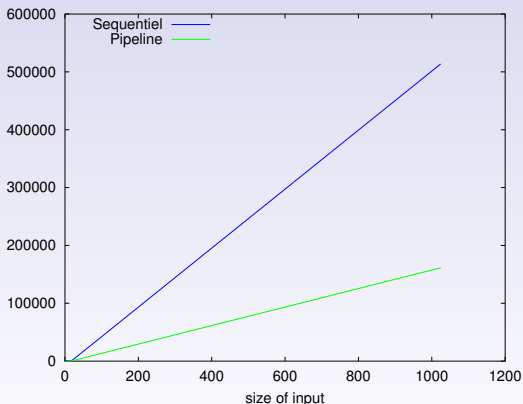
- réduction de la latence par une meilleure utilisation des ressources.

## Résultats pipeline

Limitations : 1024 valeurs issues de l'acquisition.

Usage : 6% des ressources, 3 RAMs & 1 multiplieur.

Temps de calcul : 2780 ns pour 32 valeurs et 16 coefficients de filtre.



$$nfil = 32, 1 < n < 1024$$

$$t_{tot} = t_{fir} + t_{clock} * nfil * (ninput - nfil)$$

Objectifs

Plan

FPGA

étude formules

Choix

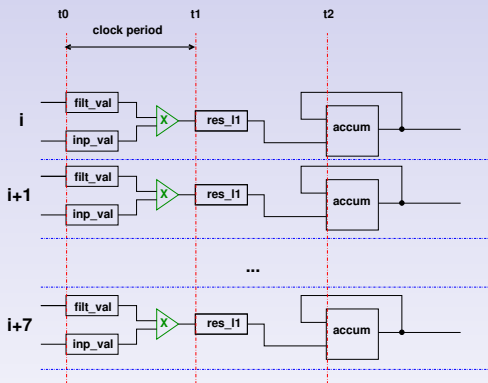
VHDL

Outils

carte  
ARMadeusimplemen-  
tations

Pipeline

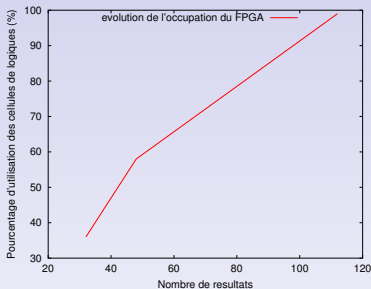
Conclusion



Ce traitement implique :

- la génération de plusieurs instances du pipeline,
- la duplication des données dont l'accès n'est pas séquentiel,
- l'impossibilité de stocker les résultats en RAM (accès parallèle).

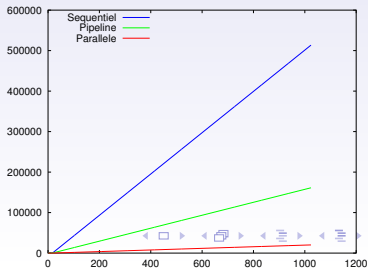
## Résultats Parallèle



### Limitations :

- 1024 valeurs issues de l'acquisition.
- nombre de résultats :  $n_{input} - n_{fil} \leq 112$ .
- 8 flux parallèle.

Durée pour 32 valeurs  
et 16 filtres : 350ns



- la compréhension du FPGA est primordiale,
- une analyse détaillée du problème est nécessaire,
- la qualité du design implique une bonne vision du circuit logique équivalent

Présentation disponible sur  
<http://www.trabucayre.com/formationPPF.pdf>