Laboratoire Kastler Brossel
Physique quantique et applications

Principe de fonctionnement d'un FPGA et de son utilisation pour la métrologie temps-fréquence

Pierre Cladé

SORBONNE UNIVERSITÉ

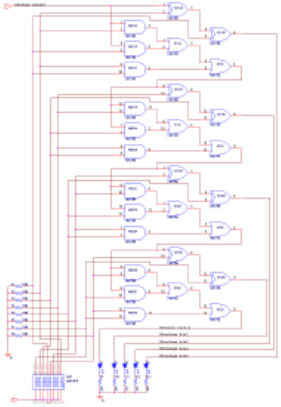
cnrs

ENS
ÉCOLE NORMALE SUPÉRIEURE

COLLÈGE DE FRANCE
1530

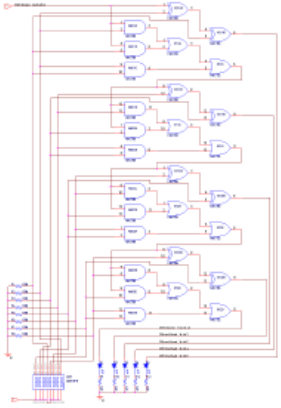
Exemple de circuit logique

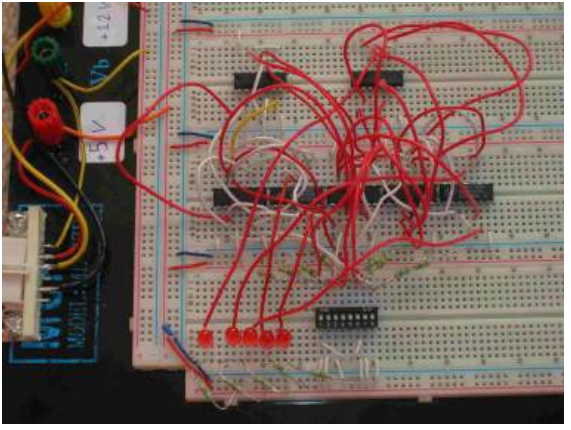
Exemple : un additionneur 4 bits



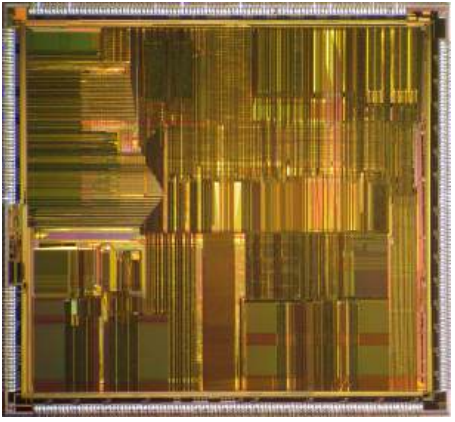
Exemple de circuit logique

Exemple : un additionneur 4 bits





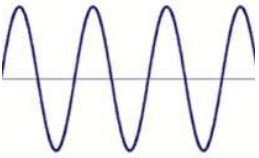
Circuit logique intégré

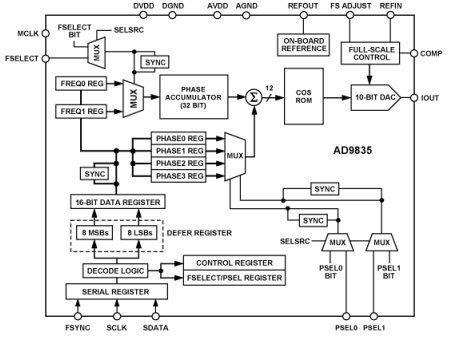


- Exemple du CPU : un grand nombre de fonctionnalités et un système pour exécuter des instructions séquentiellement
- Image du CPU: une cuisine équipée et un chef de cuisine qui suit une recette

Circuit spécialisé

Générateur de fonction (DDS)

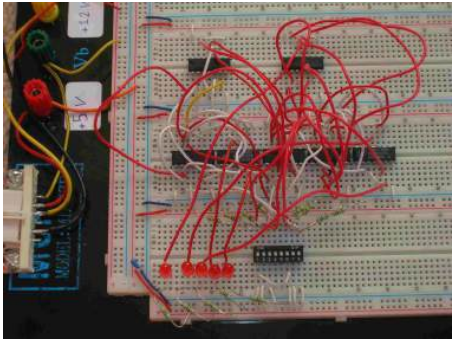




- Traitement des données en parallèle
- Chaque bloc effectue une tâche simple
- Problème : on ne peut pas modifier le circuit.

FPGA

Un FPGA (field-programmable gate array) est un type de circuit intégré configurable pouvant être programmé à plusieurs reprises après sa fabrication. Les FPGA constituent un sous-ensemble des dispositifs logiques appelés "dispositifs logiques programmables" (PLD).



FPGA

Un FPGA (field-programmable gate array) est un type de circuit intégré configurable pouvant être programmé à plusieurs reprises après sa fabrication. Les FPGA constituent un sous-ensemble des dispositifs logiques appelés "dispositifs logiques programmables" (PLD).



FPGA

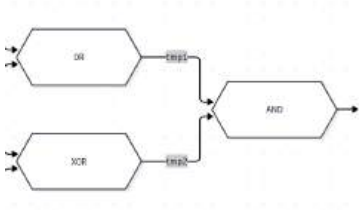
Un FPGA (field-programmable gate array) est un type de circuit intégré configurable pouvant être programmé à plusieurs reprises après sa fabrication. Les FPGA constituent un sous-ensemble des dispositifs logiques appelés "dispositifs logiques programmables" (PLD).



- Premiers FPGA dans les années 1980...
- Toute la gamme de prix



Comment programmer un FPGA ?

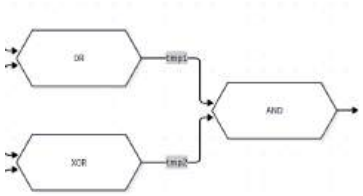


Assemblage de portes logiques.

```
internal signal :  tmp1, tmp2
or(a, b, tmp1)
xor(c, d, tmp2)
and(tmp1, tmp2, x)
```

Comment programmer un FPGA ?

Langage HDL




```
module logique (
   input wire a,
   input wire b,
   input wire c,
   input wire d,
   output wire x);

   assign x<= (a or b) and (c xor d)

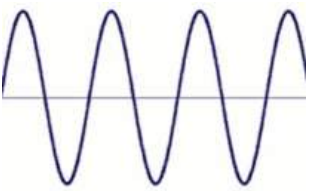
endmodule
```

Plusieurs niveaux de programmation :

- On regroupe la logique dans un bloc
- On assemble les blocs

Exemple : un générateur (DDS)

$$y(t) = sin(2\pi ft)$$



- $t_i = i \times \delta_t$ (8 ns pour le Red Pitaya)
- $y(t_i) = sin(\phi_i)$ avec $\phi_{i+1} = \phi_i + \delta_\phi$ et $\delta_\phi = 2\pi f \delta_t$
- Calcul de la phase et conversion en sin(ϕ) avec une table en mémoire

```
module dds_simple (
    input  wire |clk,
    input  wire [32-1:0] delta_phi,
    output reg  [-1:0]  sine_addr
);

    reg [PHASE_WIDTH-1:0] phase_acc;

    always @(posedge clk) begin
        phase_acc <= phase_acc + delta_phi;
        sine_addr <= phase_acc[PHASE_WIDTH-1 -: ADDR_WIDTH];
    end

endmodule
```

Traitement digital du signal

Filtre passe bas :

$$\frac{dy}{dt} = -\alpha(y(t) - x(t))$$

- $y_{i+1} = y_i - \epsilon(y_i - x_i)$. Cas simple : $\epsilon = 1/2^N$

```
module low_pass_filter (
    input  wire clk,
    input  wire signed [15:0] x_in,
    output reg  signed [15:0] y_out
);

    reg signed [15:0] diff;

    always @(posedge clk) begin
        diff   <= x_in - y_prev;
        y_out  <= y_out + (diff >>> N);
    end

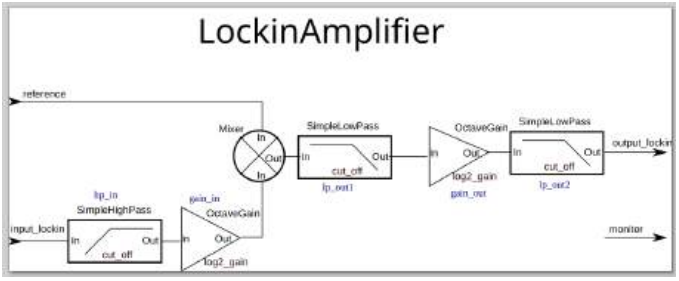
endmodule
```

- Circuit DSP en dur pour le traitement du signal

Lockin amplifier

- On peut réaliser des modules pour chaque fonction de base.
- Les fabricants proposent des modules (IP cores)

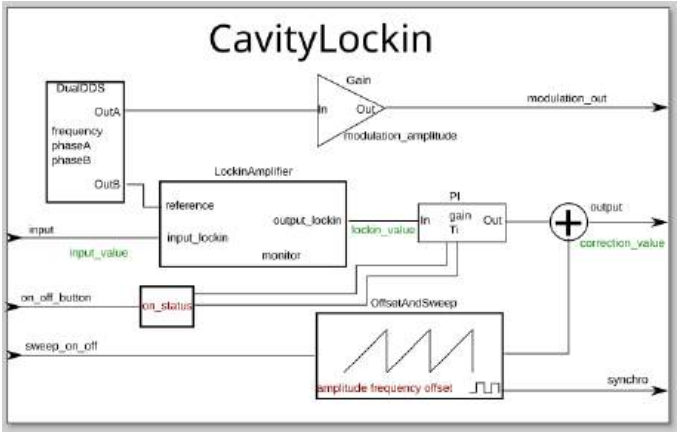
Ensuite, il faut assembler les modules.



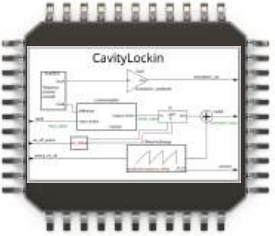
Lockin amplifier

- On peut réaliser des modules pour chaque fonction de base.
- Les fabricants proposent des modules (IP cores)

Ensuite, il faut assembler les modules.

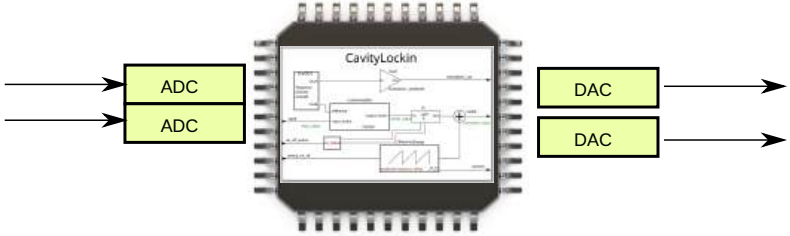


Interface avec le monde exterieur



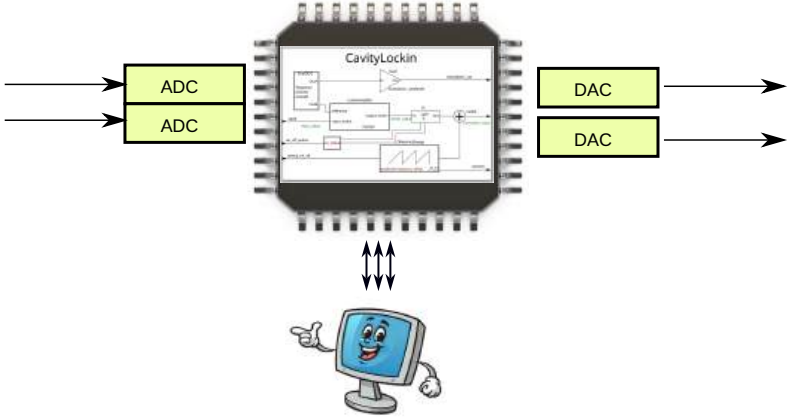
Interface avec le monde exterieur

- Convertisseur numérique analogique



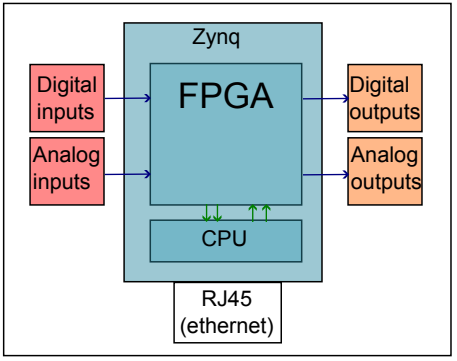
Interface avec le monde exterieur

- Convertisseur numérique analogique
- Contrôle des paramètres avec un ordinateur



La "révolution" du Red Pitaya

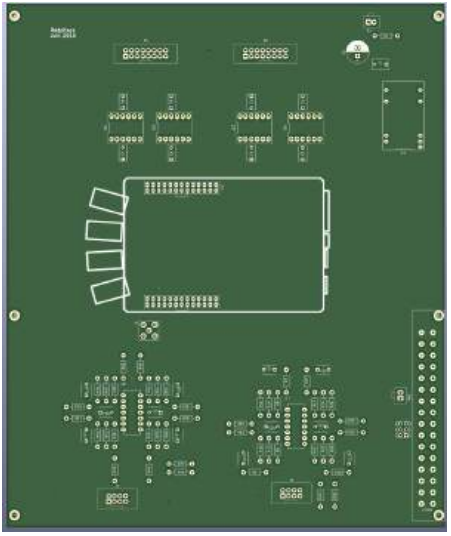




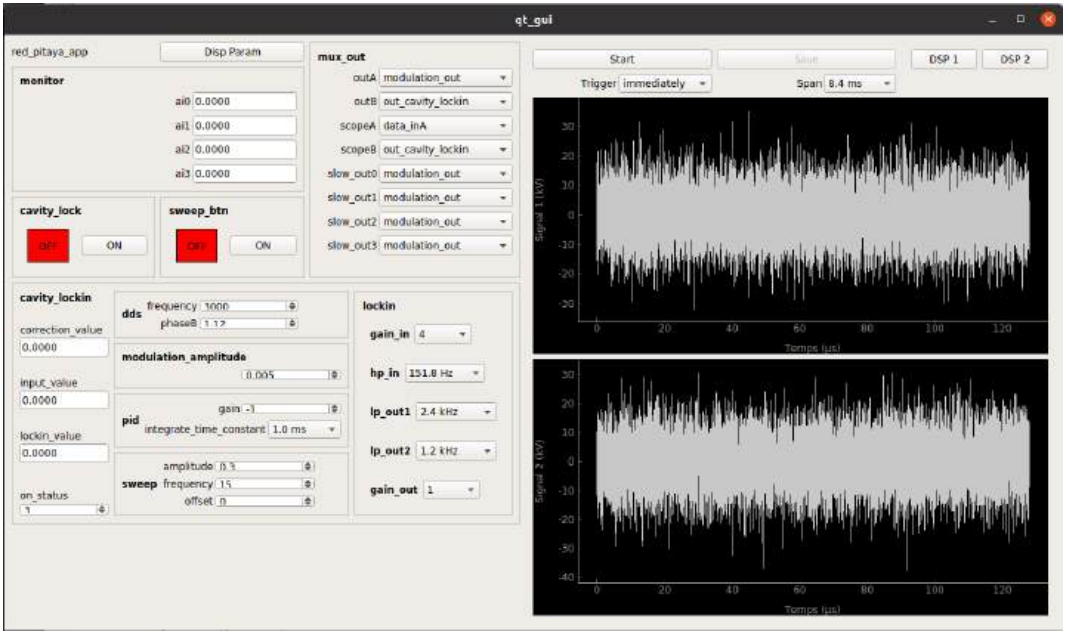
- Première version en 2014
- Logique du FPGA en opensource, plusieurs exemples

Hardware


ANALOG OUTPUT
DIGITAL INPUT
ANALOG INPUT
OUT2
IN2
CLK EXT
OUT1
IN1
ON



Interface graphique



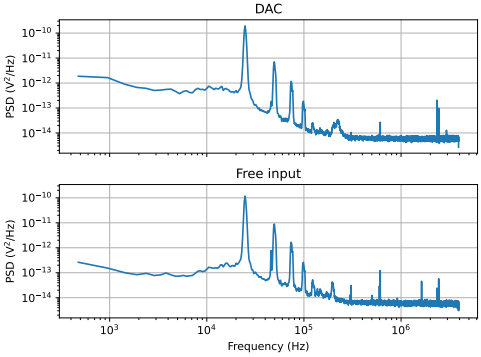
Eclypse Z7 (environ 600€)

- Open hardware
- Possibilité de mettre des cartes filles

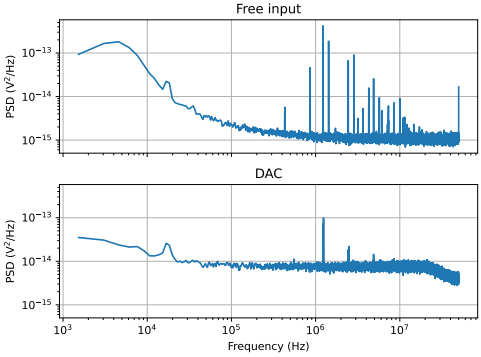


Comparaison Eclypse et Red Pitaya gen1

Red Pitaya



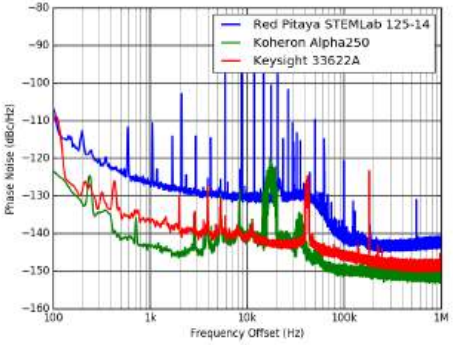
Eclypse



Koheron Alpha250 (environ 2500€)

- 250 MEch/s (vs. 125 MEch/s pour le Red Pitaya)

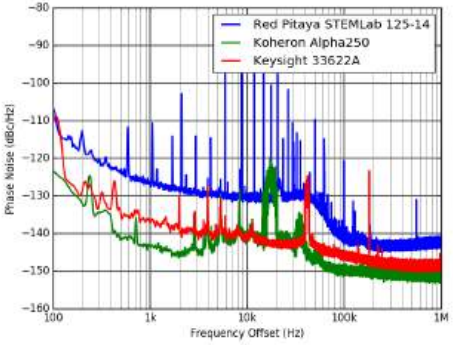




Koheron Alpha250 (environ 2500€)

- 250 MEch/s (vs. 125 MEch/s pour le Red Pitaya)





- Red Pitaya gen2 (octobre 2025, 800€)

Conclusions

- Des produits de plus en plus pratiques à utiliser. De nouvelles performances en termes de bruit.
- Côté fabricants de FPGA : des circuits plus rapides, intégration de convertisseurs dans la puce du FPGA.

- Une solution à considérer pour des applications dans le domaine RF ($<$100 MHz).
- Asservissement avec des délais de l'ordre de 100 ns
- Fonctionne à très basse fréquence
- Intégration facile dans des systèmes plus larges : interface Python, monitoring, analyse des signaux
- Une variété de solutions logicielles qui peut paraître compliquée.