TELECOM ParisTech

INSTITUT Mines-Télécom

Communications entre processus . . .

Les files d'attente, les changements de domaine d'horloge

Yves Mathieu

université PARIS-SACLAY

Introduction (1)

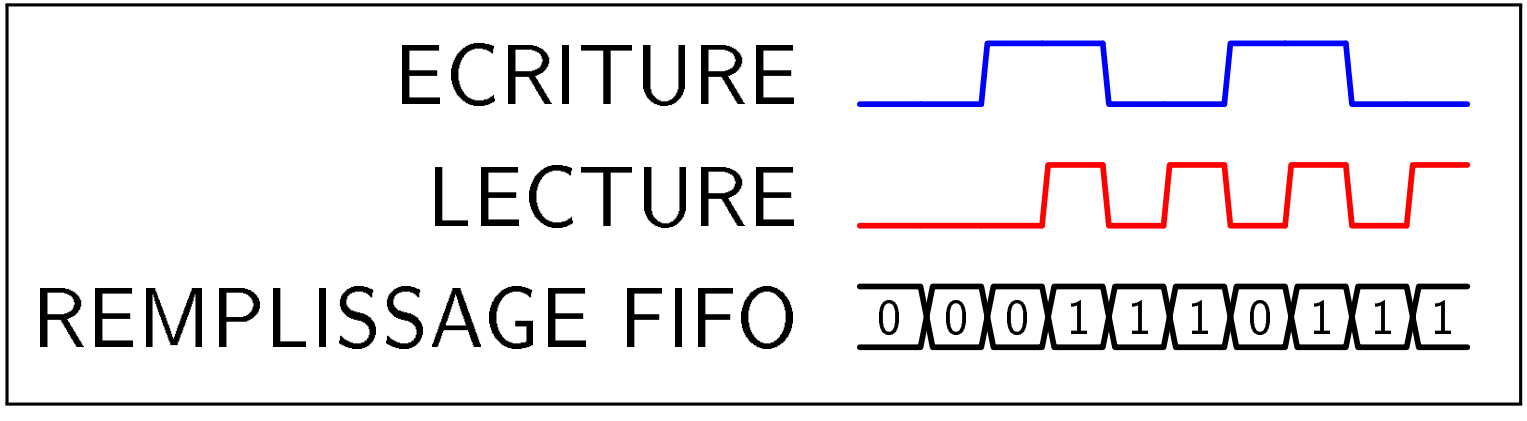
Files d'attente

- Traitement pilotés par les flux de données ("data driven").
- Tampons entre blocs de traitement.
- Files d'attente matérielles : les "FIFO" (First In, First Out)
- Inutile si le flux est "tendu".
- Une seule horloge en lecture et écriture, des intervalles (cycles) sans lecture ou sans écriture
- Intervalle de temps normalisé, N lectures pour N écritures : calcul de la taille de la FIFO, pas (ou peu) de contrôle nécessaire.
- FIFO de taille arbitraire, la détection de FIFO "vide" ou "pleine" pilote l'avancement des processus amont et aval.

Introduction (2)

FIFO : comportement et tailles calculables a priori

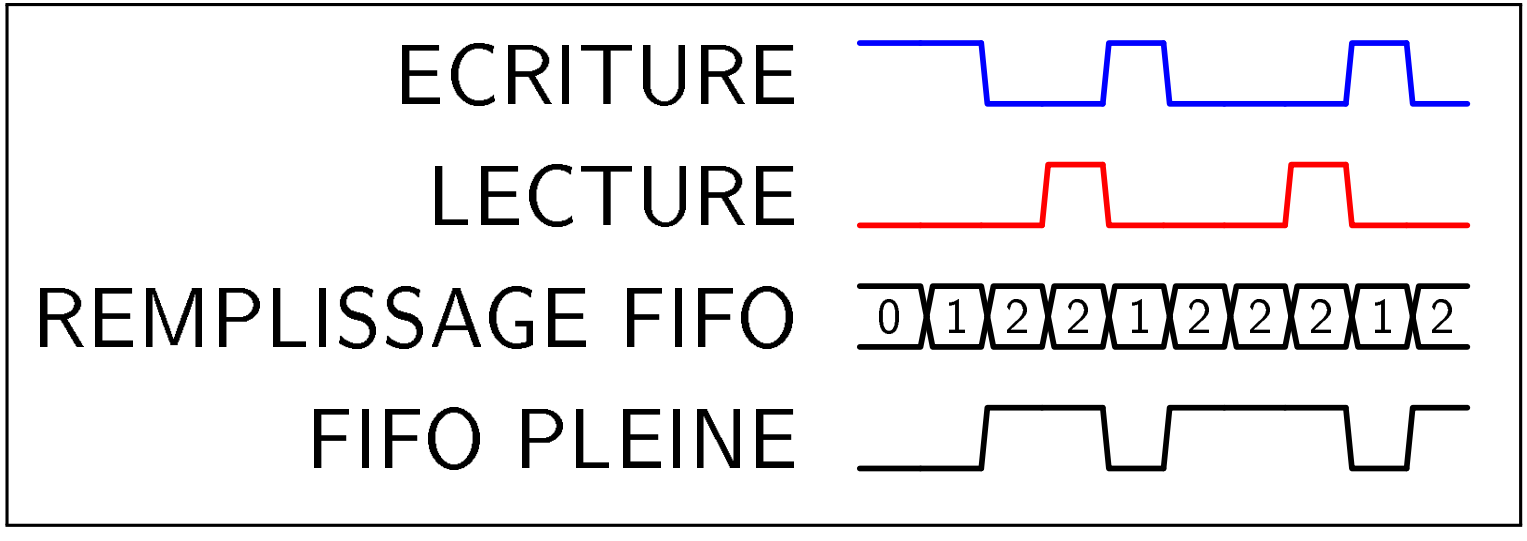
- ECRITURE : 2 cycles vides, suivis de 2 cycles d'écriture.
- LECTURE : 1 cycle de lecture, 1 cycle vide.
- PPCM : période de 4 cycles, contenant 2 lectures ou écritures
- Une FIFO de taille "1" suffit
- Le processus de lecture doit attendre l'arrivée de la première donnée.



Introduction (3)

FIFO : pilotage des processus par la FIFO

- FIFO de taille 2.
- Tentative d'écriture permanente.
- Lecture 1 cycle sur 4.
- Le système se synchronise sur le processus le plus lent



Introduction (4)

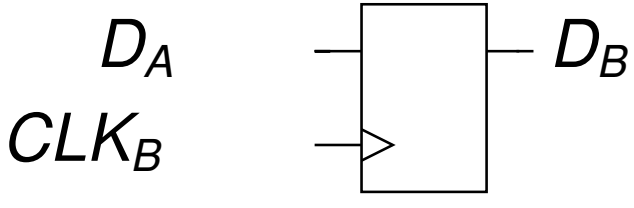
Changement de domaines d'horloge

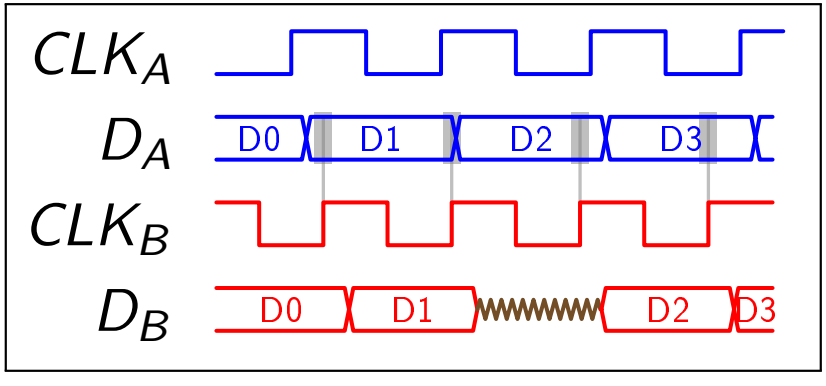
- CDC, pour "Cross Domain Crossing"
- A l'interface entre différents systèmes
- Chaque système a sa propre horloge : pas de relation de phase, de fréquence
- Les problèmes : Métastabilité des bascules, respect du protocole
- Les erreurs sont :
  - subtiles, intermitantes, non prédictibles
  - difficiles à reproduire et déboguer.
  - dépendantes de la température, des tensions, de la technologie
  - sur le matériel, pas dans les simulations...

Introduction (5)

Métastabilité

- Bascule D: Horloge et données sans relation de causalité
- Non respect des temps de prépositionnement et de maintien.
- Comportement non prédictible de la sortie de la bascule
- Peut se propager dans la logique en aval





Introduction (6)

Métastabilité

Les conséquences de la métastabilité peuvent être modélisées:

$$MTBF = \frac{\exp(t_x/\tau)}{F_{clk_A} \cdot F_{clk_B} \cdot T_o}$$

Avec:

- $MTBF$: Mean Time Between Failures
- t_x : Temps laissé à la bascule pour sortir de l'état métastable.
- τ : Constante de normalisation
- F_{clk_A} : Fréquence de l'horloge des données entrantes
- F_{clk_B} : Fréquence de l'horloge de sortie
- T_o : Fenêtre de métastabilité $\simeq (T_{setup} + T_{hold})$

Introduction (7)

MTBF

ATTENTION :

- Le MTBF d'un système multi-domaines d'horloge n'est JAMAIS infini
- Il existe des techniques pour l'augmenter arbitrairement
- Ces techniques on des conséquences sur la limitation des performances de ces systèmes
- Les outils pour FPGA proposent de d'estimer le MTBF des systèmes conçus.

Files d'attente

Principe général

- Données stockées dans une mémoire synchrone.
- Un compteur d'adresse d'écriture modulo la taille de la mémoire.
- Un compteur d'adresse de lecture modulo la taille de la mémoire.
- Si le compteur de lecture rattrape le compteur d'écriture alors la FIFO est vide.
- Si le compteur d'écriture rattrape le compteur de lecture alors la FIFO est pleine.
- Dans la pratique les compteurs font un bit de plus que nécessaire: test des bits de poids fort

Files d'attente

FIFO simple: Les signaux d'entrée/sortie

```
// Une fifo synchrone
// Tous les signaux de sortie sont séquentiels.
// Les ports d'entrée et de sortie se comportent comme des "esclaves"
// Sur requete d'écriture une donnée est poussée dans la FIFO.
// Sur requete de lecture une donnée sort de la FIFO au cycle suivant.
// La FIFO génère des indicateurs "wfull" et "rempty"
// ATTENTION La FIFO ne vérifie pas les incohérences:
module fifo_ref #( parameter DATA_WIDTH  = 8, DEPTH_WIDTH = 5)
                  ( input logic clk,
                    input logic rst,
                    // Port d'entrée
                    input logic wreq,
                    input logic [DATA_WIDTH-1:0] wdata,
                    output logic wfull,
                    // Port de sortie
                    input logic rreq,
                    output logic [DATA_WIDTH-1:0] rdata,
                    output logic rempty);

// Nombre de données de la FIFO
localparam DEPTH = 2**DEPTH_WIDTH;
```

Files d'attente

FIFO simple: Déclarations : mémoire, compteurs

```
// La mémoire  de la FIFO
(* synthesis, ignore_ram_rw_collision =  "true" *)
(* altera_attribute = "-name  add_pass_through_logic_to_inferred_rams off"*)
logic [DATA_WIDTH-1:0]     mem [DEPTH-1:0];

// Les compteurs d'écriture et de lecture
// ATTENTION nous utilisons un bit de plus que nécessaire
// pour faciliter la détection des dépassement
typedef struct packed {
              logic segment ;
              logic [DEPTH_WIDTH-1:0] address ;
              } index_t ;
```

Files d'attente

FIFO simple: Code de la mémoire

```
// Le code de la mémoire double port
// La lecture ou l'écriture ne se fait
// que sur requête externe
index_t  r_index,w_index;
always_ff @(posedge clk)
begin
  if (wreq)
      mem[w_index.address] <= wdata;
  if (rreq)
      rdata <= mem[r_index.address];
end
```

Files d'attente

FIFO simple: Logique de comptage

```
always_ff @(posedge clk ) begin:indexes
  // Variables combinatoires internes
  index_t  next_r_index,next_w_index;
  logic next_empty_or_full, next_empty, next_full ,next_same_segment;
  // Calcul combinatoire des valeurs pour le cycle suivant
  next_r_index = (r_index + rreq) ;
  next_w_index = (w_index + wreq) ;
  next_empty_or_full = (next_r_index.address == next_w_index.address) ;
  next_same_segment = (next_r_index.segment == next_w_index.segment) ;
  next_empty = ( next_empty_or_full &  next_same_segment );
  next_full  = ( next_empty_or_full & ~next_same_segment );
  // Sauvegarde dans les registres
  r_index <= next_r_index;
  w_index <= next_w_index;
  rempty <=  next_empty ;
  wfull  <=  next_full  ;
  if (rst) begin
      r_index    <= '0  ; w_index    <= '0  ;
      rempty     <= 1'b1; wfull      <= 1'b0;
  end
end
endmodule
```

Files d'attente

FIFO simple: A vous de jouer

- Compilez le code de **fifo_ref.sv** et **tb_fifo_ref.sv**
- Le testbench génère des requêtes aléatoires en entrée
- Le testbench génère des requêtes régulières en sortie
- Préparez la simulation, en affichant les différents signaux
- Passez en mode analogique le signal "nbdata" (Format/Analog(automatic)
- Lancez la simulation (run -all) et observez les résultats
- Vous pouvez jouer sur la taille de la FIFO et observer les changements.

Files d'attente

Insertion dans un système Maitre/Esclave

- Les deux ports de la FIFO simple sont "Esclaves".
- L'insertion d'une FIFO dans un protocole Maître/Esclave nécessite une adaptation.
- Anticipation des lectures pour que le port de sortie se comporte en Maître.
- Facile à adapter à un protocole genre WishBone.
- Exemple : une FIFO avec anticipation des lectures peut se connecter simplement entre :
  - Un maître wishbone ne faisant que des écritures
  - et un esclave wishbone ne faisant que des lectures
  - Application : Ecriture d'une image décodée dans une mémoire d'image.

Files d'attente

FIFO avec anticipation: Les signaux d'entrée/sortie

```
// Un fifo synchrone
// Le port d'entrée se comporte comme un esclave, et signale la réussite
// de l'écriture par un acquitement combinatoire
// Le port de sortie se comporte comme un maître et  signale la présence
// d'une donnée par une requète synchrone
// Le maître doit maintenir la requère en entrée tant
// que l'acquitement n'est pas arrivé.
module fifo_sm #( parameter DATA_WIDTH  = 8, DEPTH_WIDTH = 5)
                ( input logic clk,
                  input logic rst,
                  // Port d'entrée
                  input logic wreq,
                  input logic [DATA_WIDTH-1:0] wdata,
                  output logic wack,
                  // Port de sortie
                  output logic rready,
                  output logic [DATA_WIDTH-1:0] rdata,
                  input  logic rack);

// Nombre de données de la FIFO
localparam DEPTH = 2**DEPTH_WIDTH;
```

Files d'attente

FIFO avec anticipation: Déclarations : mémoire, compteurs

```
// La mémoire  de la FIFO
// Attention les mémoires double port inférées ont un comportement non
// prévisible lorsqu'il y a une lecture et une écriture à la même adresse.
// Les outils de synthèse ne savent pas détecter si ce cas peut arriver ou
// non. Ils peuvent ajouter une logique de "bypass" inutile autour de la mémoire
// Attention, par construction nous garantissons que la lecture et l'ecriture ne
// sont pas à la même adresse. Nous utilisons des attributs spécifiques aux outi
// pour guider la synthèse
// The 2 attributes are equivalent; the first targets Precision RTL synthesis
// and the second QuartusII
(* synthesis, ignore_ram_rw_collision =  "true" *)
(* altera_attribute = "-name  add_pass_through_logic_to_inferred_rams off"*)
logic [DATA_WIDTH-1:0]     mem [DEPTH-1:0];

// Les compteurs d'écriture et de lecture
// ATTENTION nous utilisons un bit de plus que nécessaire
// pour faciliter la détection des dépassement
typedef struct packed {
                logic segment ;
                logic [DEPTH_WIDTH-1:0] address ;
                } index_t ;
```

Files d'attente

FIFO avec anticipation: Code de la mémoire

```
// Le code de la mémoire double port
// L'écriture ne se fait que si toutes les conditions
// sont réunie (requète et fifo non pleine)
// La lecture est systèmatiquement anticipée
// dans l'attente d'un acquittement futur.
index_t  r_index,w_index;
always_ff @(posedge clk)
begin
  if (wack)
      mem[w_index.address] <= wdata;
end
assign  rdata = mem[r_index.address];
```

Files d'attente

FIFO avec anticipation: Gestion des requêtes

```
// Drapeaux internes à la FIFO
logic wfull,rempty ;

// Réponses aux requêtes en écriture
assign wack  = wreq & ~wfull ;

// Création de requêtes en lecture
assign rready =  ~rempty ;
```

Files d'attente

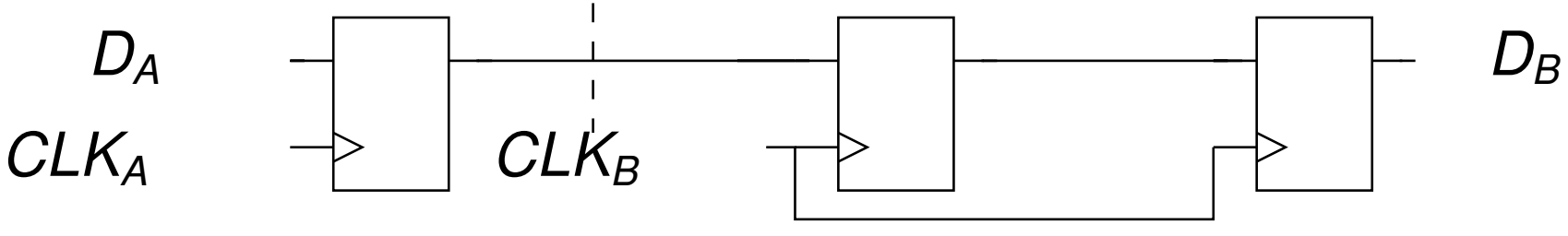
FIFO avec anticipation: Logique de comptage

```
always_ff @(posedge clk ) begin:indexes
    // Variables combinatoires internes
    index_t  next_r_index,next_w_index;
    logic next_empty_or_full, next_empty, next_full ,next_same_segment;
    // Calcul combinatoire des valeurs pour le cycle suivant
    next_r_index = r_index + rack ;
    next_w_index = w_index + wack;
    next_empty_or_full = (next_r_index.address == next_w_index.address);
    next_same_segment  = (next_r_index.segment == next_w_index.segment);
    next_empty         = ( next_empty_or_full & next_same_segment );
    next_full          = ( next_empty_or_full & ~next_same_segment );
    // Sauvegarde dans les registres
    r_index <= next_r_index;
    w_index <= next_w_index;
    rempty <= next_empty ;
    wfull <= next_full ;
    if (rst)
    begin
        r_index <= '0   ; w_index <= '0   ;
        rempty  <= 1'b1 ; wfull   <= 1'b0 ;
    end
end
```

Domaines d'horloge

Rééchantillonnage

- Un signal de contrôle simple (exemple : requête sur 1 bit)
- Passage du domaine Clk_A au domaine Clk_B
- Le MTBF augmente énormément (plusieurs années) si l'on utilise plusieur bascules en série.
- **R1** : Utiliser deux bascules dans le domaine Clk_B
- Le MTBF diminue si le signal rééchantillonné provient d'une reconvergence de calcul combinatoire.
- **R2** : Sortie d'une bascule dans le domaine Clk_A



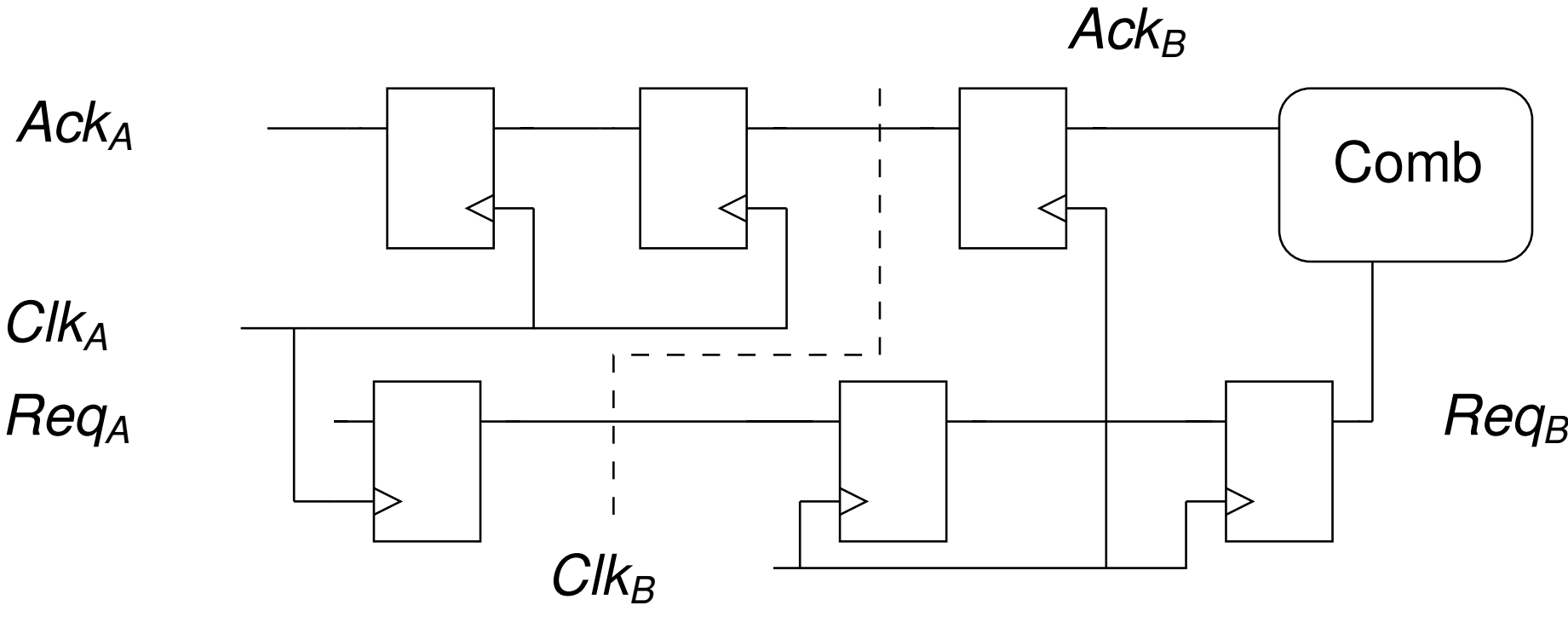
Domaines d'horloge

Conservation de protocole

- Incertitude sur le nombre de cycles à 1 ou 0 d'un signal.
- **R3** : Pas de signalisation basée sur un comptage de cycles.
- **R4** : Le signal doit rester stable suffisament longtemps dans chaque domaine d'horloge.
- Conversion des protocoles par impulsions en protocole par changement d'état
- Solution générale pour une transmission en boucle ouverte (sans feedback)

Domaines d'horloge

Transmision en boucle fermée (req/ack)



- Jusqu'à 5 cycles perdus dans la boucle de synchronisation
- **R5** Privilégier des synchronisations globales (par paquets de transferts)

Domaines d'horloge

Transmision d'une données (n bits)

- Pas de garantie d'arrivées des bits dans le même cycle...
- **R6** Ne rééchantillonner que les signaux de contrôle
- **R7** Le récepteur échantillonne les données à partir de l'information du contrôle.
- **R8** Conserver les données dans l'attente de l'acquitetment

Domaines d'horloge

la FIFO asynchrone

- Couteau Suisse de la resynchronisation.
- Deux horloges pour la FIFO.
- Réfléchissons ensemble au problème...
- C'est tout pour aujourd'hui.