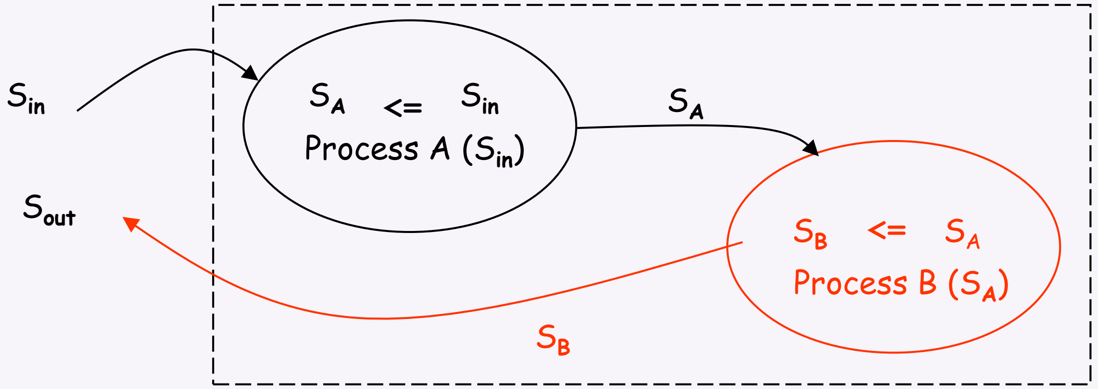
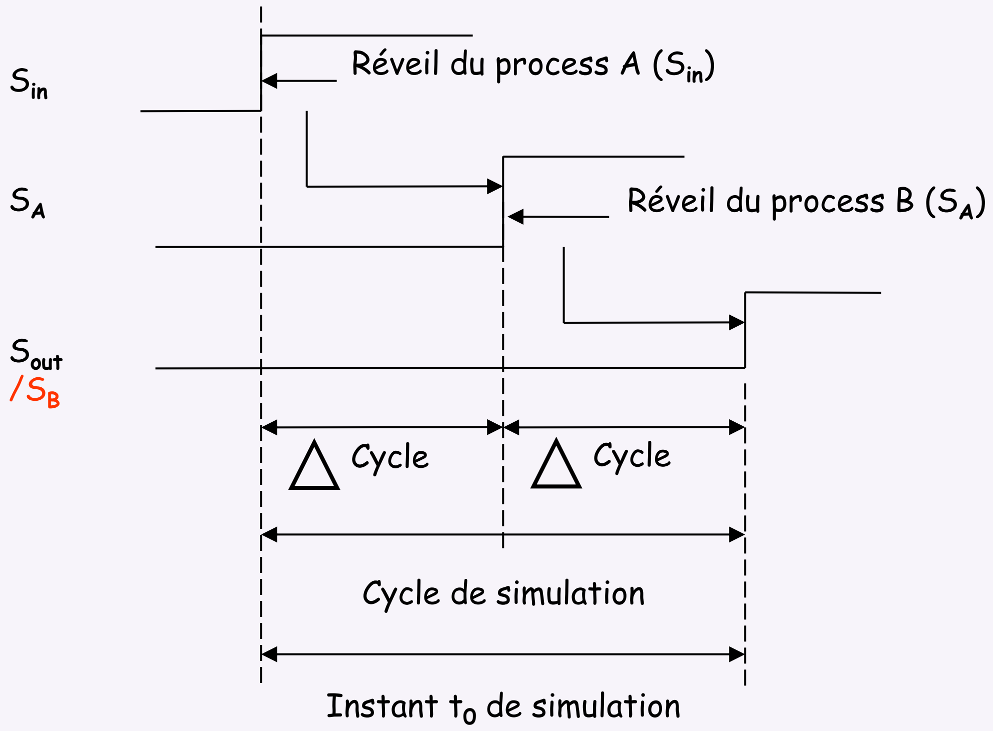
Le langage VHDL

Hervé Le Provost, CEA, Irfu

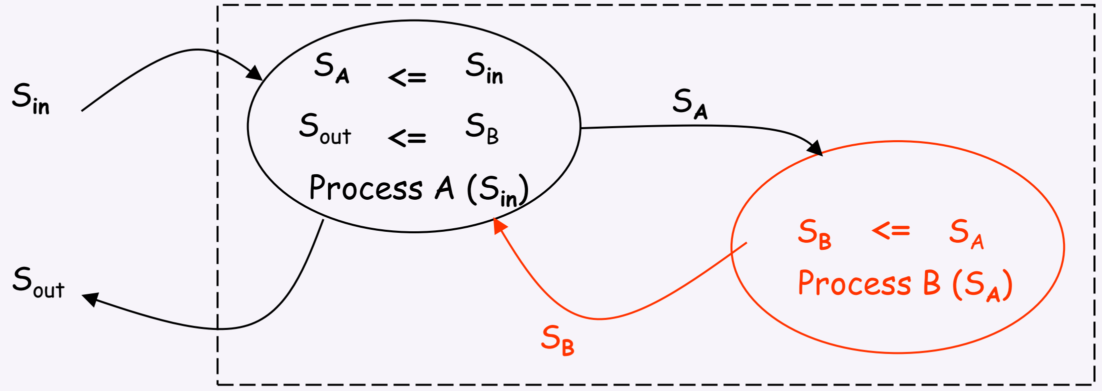
herve.le-provost@cea.fr

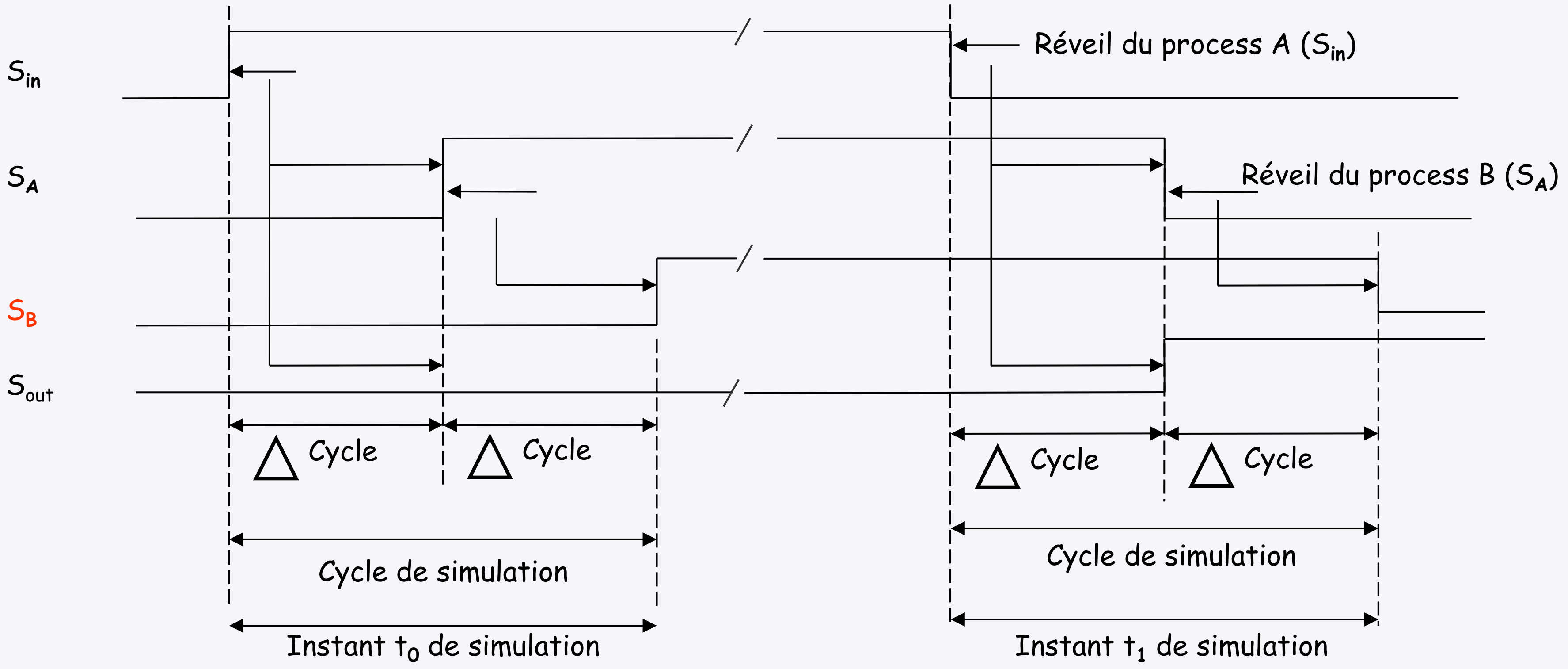
Cycle de Simulation I




S_{in}
Réveil du process A (S_{in})
S_A
Réveil du process B (S_A)
S_{out} /S_B
Δ Cycle
Δ Cycle
Cycle de simulation
Instant t_0 de simulation

Cycle de Simulation 2





S_{in}
S_A
S_B
S_{out}
Réveil du process A (S_{in})
Réveil du process B (S_A)
△ Cycle
△ Cycle
△ Cycle
△ Cycle
Cycle de simulation
Cycle de simulation
Instant t_0 de simulation
Instant t_1 de simulation

Cycle de simulation 3

Que fait ce code VHDL ?

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

entity plantage is
end plantage;

architecture rtl of plantage is
signal sa,sb: std_logic := '0';
begin
        pSa : process (sb)
        begin
                sa <= not sb;
        end process pSa;

        pSb : process (sa)
        begin
                sb <= sa;
        end process pSb;

end rtl;
```

```
sa <= sb;
```

état stable atteint mais incorrect sur le fond

Une bascule D en VHDL

VHDL RTL (Register Transfer Level)

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

entity bascule is
        port(
                nCl, clk, d : in std_logic;
                q : out  std_logic
        );
end bascule;

architecture rtlBehavior of bascule is
begin
        pBasc : process (clk, nCl)
        begin
                if (nCl = '0') then
                        q <= '0';
                elsif (clk'event and clk = '1') then
                        q <= d;
                end if;
        end process pBasc;

architecture rtlDataFlow of bascule is
begin
        q <= '0' when (nCl = '0') else
                d when (clk'event and clk='1');
end rtlDataFlow;
```

Déclaration des librairies et packages utilisés (Cf. document QUALIS)

Déclaration des ports externes

Déclaration de l'architecture associée à l'entité

Liste de sensibilité

Instruction concurrente "process"

Instructions séquentielles

Instruction concurrente ou séquentielle ?

Déclaration d'une seconde architecture associée à une même entité

Banc de test d'une bascule D en VHDL I

VHDL NON RTL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY basculeTb_vhd IS
END basculeTb_vhd;

ARCHITECTURE behavior OF basculeTb_vhd IS

        COMPONENT bascule
        PORT(
                nCl, clk, d : IN std_logic;
                q : OUT std_logic
                );
        END COMPONENT;
        --Inputs
        SIGNAL nCl :  std_logic;
        SIGNAL clk :  std_logic;
        SIGNAL d :  std_logic;
        --Outputs
        SIGNAL q :  std_logic;

…
```

Entité vide

Déclaration du composant (association avec le nom)

Déclaration des signaux interne à l'architecture (communication entre les process)

Banc de test d'une bascule D en VHDL 2

VHDL NON RTL

Instanciation composant

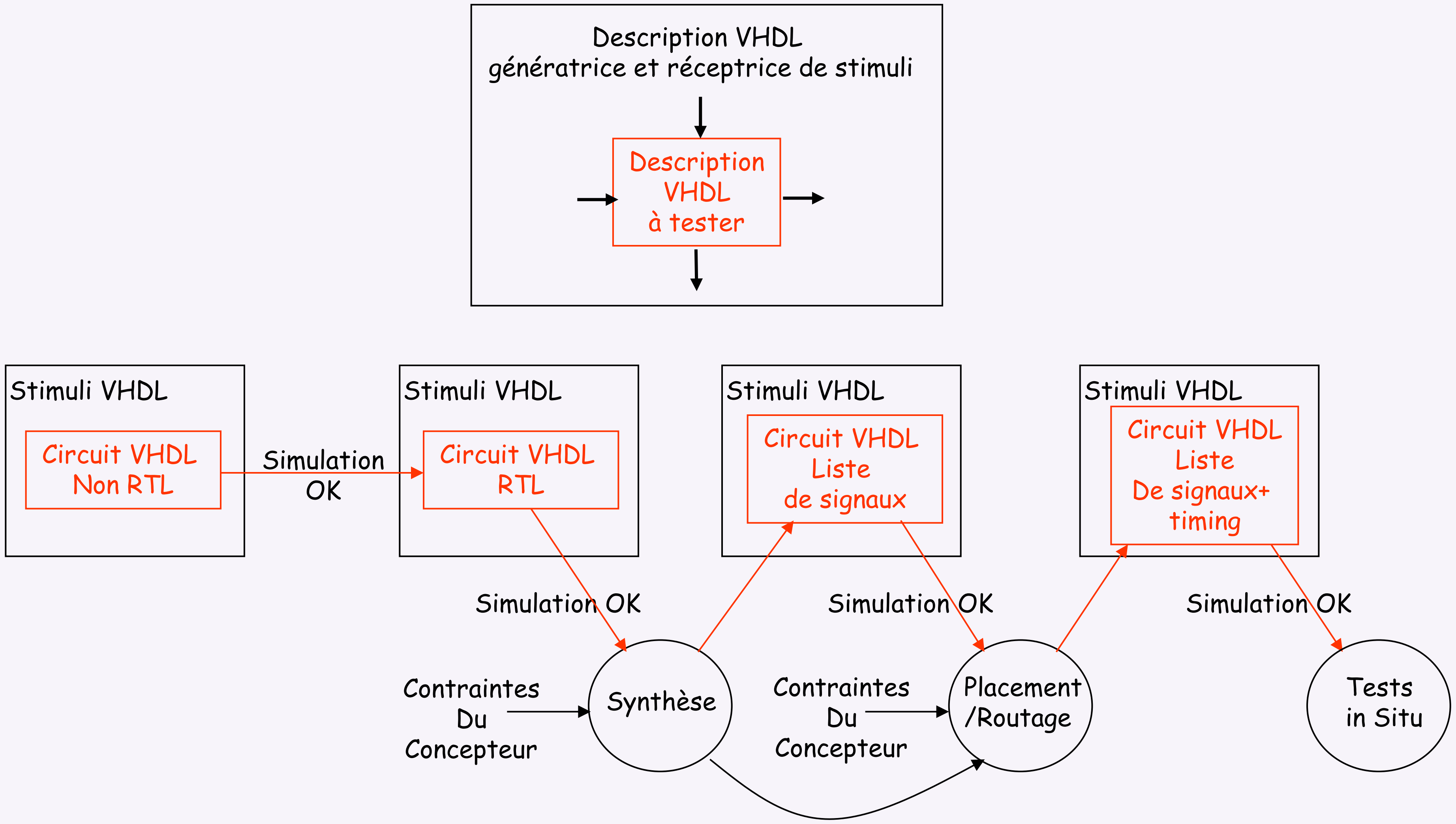
Non synthétisable

Génération entrée

Génération horloge

```
BEGIN

        -- Instantiate the Unit Under Test (UUT)
        c_bascule: bascule
        PORT MAP (nCl => nCl, clk => clk, d => d, q => q);

        tb : PROCESS
        BEGIN
                nCl     <= '0', '1' after 200 ns;
                wait; -- will wait forever
        END PROCESS;

        pGenD: process (clk)
        begin
                if (clk'event and clk='1') then
                        d <= not d after 5 ns;
                end if;
        end process pGenD;

        pClk: process
        constant periodClkNs : time       := 35 ns;
        begin
                clk  <= '0'; wait for periodClkNs/2;
                clk <= '1';wait for periodClkNs/2;
        end process pClk;
END;
```

Le flot de développement VHDL



Synthèse VHDL RTL

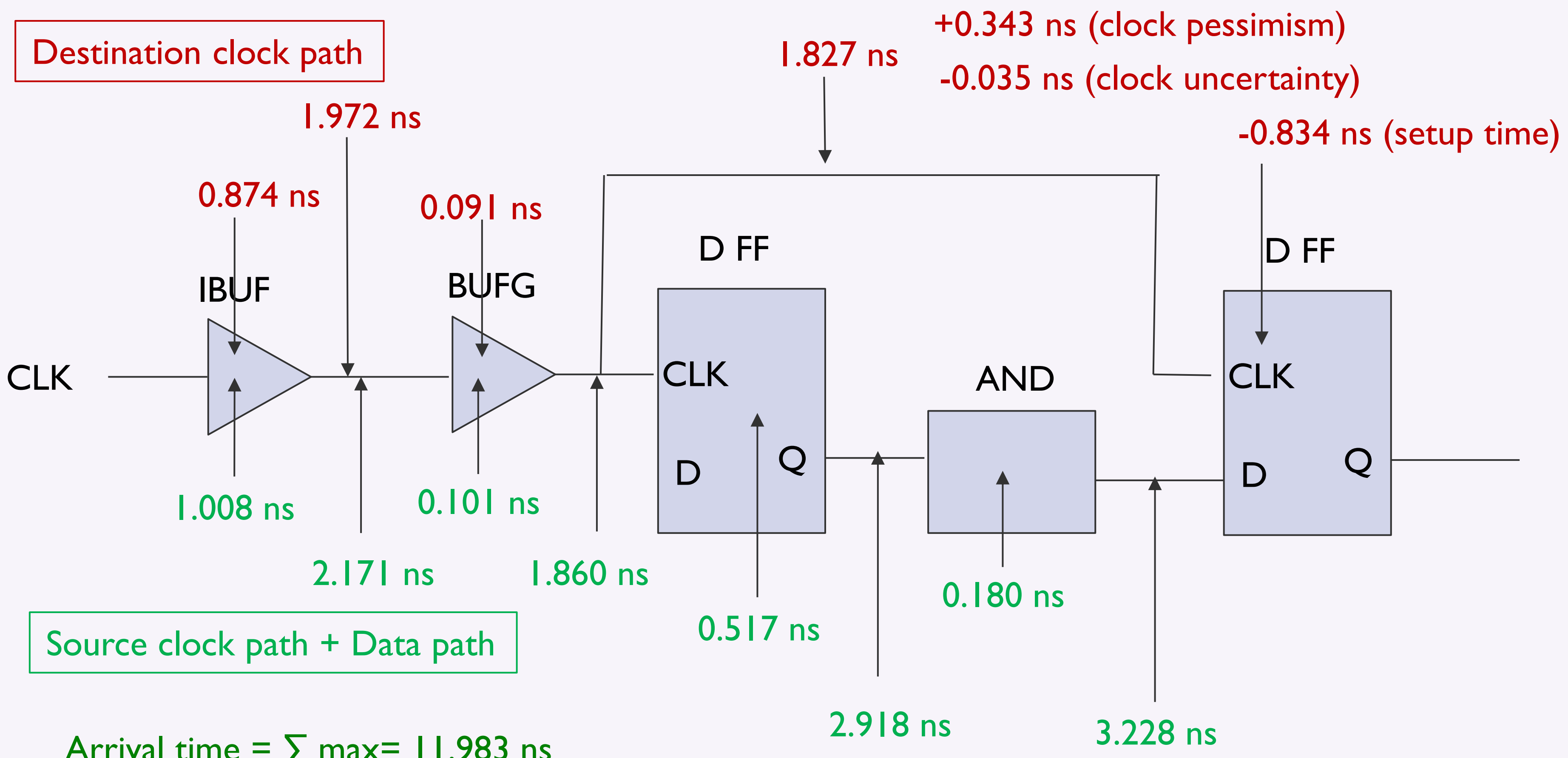
Que fait ce code VHDL ?

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

entity procSignal is
   port(
      nCl    : in   std_logic;
      clk    : in   std_logic;
      d      : in   std_logic;
      q      : out  std_logic
      );
end procSignal;

architecture rtl of procSignal is
begin
   pBasc : process (clk, nCl)
   begin
      if (nCl = '0') then
        q <= '0';
      elsif (clk = '1') then
         q  <= d;
      end if;
   end process pBasc;
end rtl;
```

flip-flop inferred: `elsif (clk'event and clk = '1') then`

`latch inferred even if missing d in the sensitivity list`

Fréquence maximale



Arrival time = ∑ max= 11.983 ns

Required time = ∑ min= 5 ns + 4.764 ns + 0.343 ns – 0.035 ns – 0.834 ns = 9.238 ns

@200 MHz

0,343 ns = (1.008+2.171+0.101) ns - (0.874+1.972+0.091) ns

Slack = Required – Arrival time
= 9,238 ns – 11,983 ns = - 2,745 ns
OK si > 0

Synthèse et “Wait on”

Reference : UG901 `(v2015.2) June 24, 2015` *Vivado Synthesis User Guide, p157-158*

IMPORTANT: *The same sequential process cannot have both a sensitivity list and a wait statement, and only one wait statement is allowed.*

D Flip-Flop:

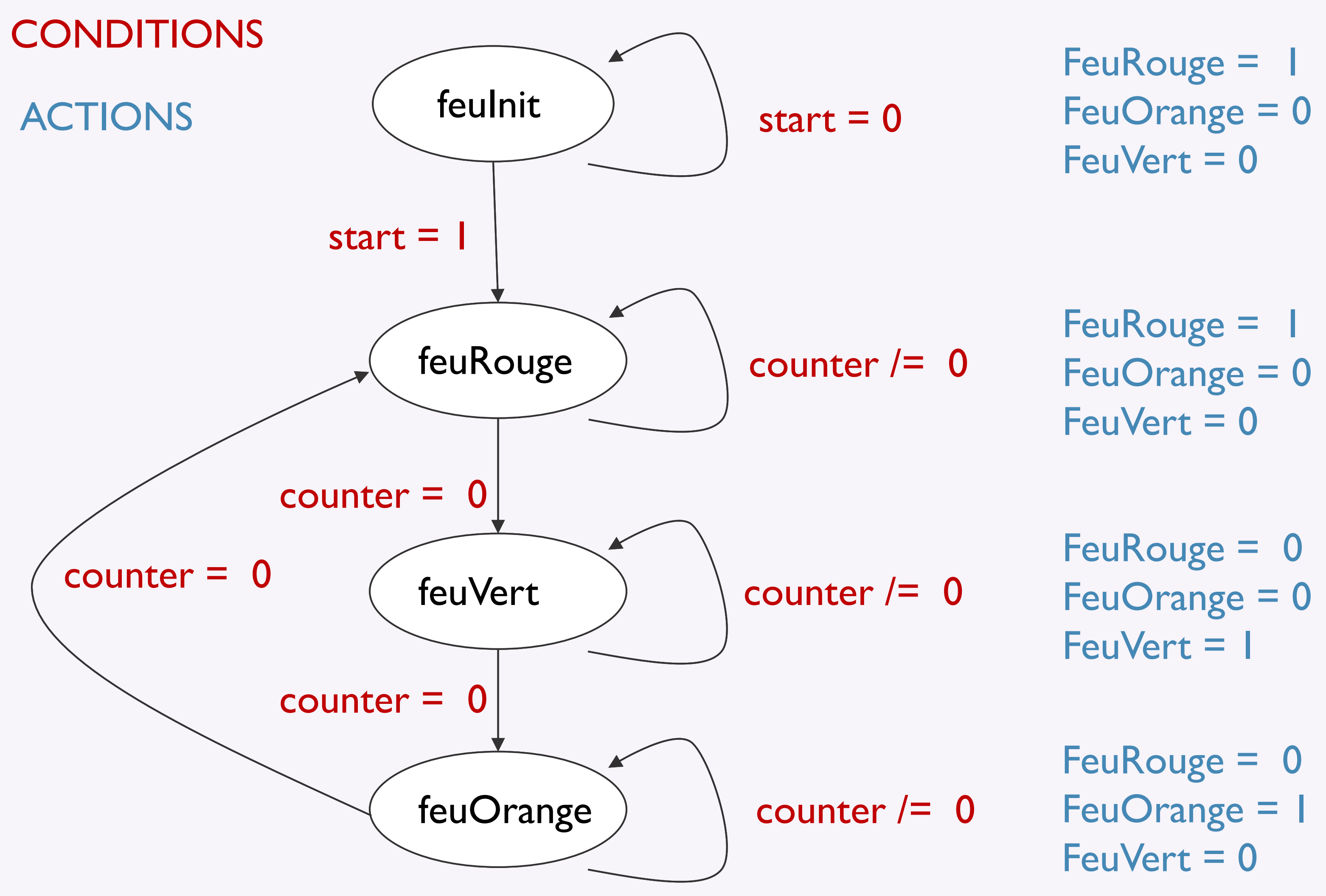
```
Process
begin
        wait until rising_edge (clk);
        q <= d;
end process;
```

IMPORTANT: *You cannot describe a sequential element with asynchronous control logic using a process without a sensitivity list. Only a process with a sensitivity list allows such functionality. Vivado synthesis does not allow the description of a Latch based on a wait statement. For greater flexibility, describe synchronous logic using a process with a sensitivity list.*

Si “clk” ajouté à la liste de sensibilité, `process (clk)`:

```
[Synth 8-2578] process cannot have both a wait statement and a sensitivity list
```

Machine à états I



Machine à états 2

```
controlefeu : process (clk, reset)
          begin
          if (reset = '1') then
               feuState <= feuInit;
               counter <= (others=>'0');
                cmdVert <= '0';
               cmdOrange <= '0';
                cmdRouge <= '1';
          elsif (clk'event and clk='1') then
               case feuState is
                    when feuInit =>
                         cmdVert <= '0';
                         cmdOrange <= '0';
                         cmdRouge <= '1';
                         if (start = '1') then
                            counter <= dureeRouge;
                            feuState <= feuRouge;
                       end if;
                   when feuRouge =>
…
```