

UE VLSI

cours 5: Description des registres, détail du banc de registres

Jean-Lou Desbarbieux
UPMC 2017

Sommaire

- 1 Registres
- 2 Fifos
- 3 REG

Description VHDL d'un registre

Version simple : une entrée une sortie, une horloge.

```
process(ck)
begin
  if rising_edge(ck) then
    dout <= din;
  end if;
end process;
```

Description VHDL d'un registre avec reset

Version avec un reset synchrone : une entrée, une sortie, une horloge et un signal de reset.

```
process(ck)
begin
  if rising_edge(ck) then
    if reset_n = '0' then
      dout <= '0';
    else
      dout <= din;
    end if;
  end if;
end process;
```

Description VHDL d'un registre avec reset asynchrone

Version avec un reset asynchrone : une entrée, une sortie, une horloge et un signal de reset.

```
process(ck, reset_n)
begin
  if reset_n = '0' then
    dout <= '0';
  elsif rising_edge(ck) then
    dout <= din;
  end if;
end process;
```

Interface

```
ENTITY fifo_32b IS
  PORT(
    din      : in std_logic_vector(31 downto 0);
    dout     : out std_logic_vector(31 downto 0);
    — commands
    push     : in std_logic;
    pop      : in std_logic;
    — flags
    full     : out std_logic;
    empty    : out std_logic;

    reset_n  : in std_logic;
    ck       : in std_logic;
    vdd      : in bit;
    vss      : in bit
  );
```

Interface 1 Ecriture

```
wdata1 : in Std_Logic_Vector(31 downto 0);
wadr1  : in Std_Logic_Vector(3  downto 0);
wen1   : in Std_Logic;

wdata2 : in Std_Logic_Vector(31 downto 0);
wadr2  : in Std_Logic_Vector(3  downto 0);
wen2   : in Std_Logic;

wcry   : in Std_Logic;
wzero  : in Std_Logic;
wneg   : in Std_Logic;
wovr   : in Std_Logic;
cspr_wb : in Std_Logic;
```

Interface 2 Lecture

— *Read Port 1 32 bits*

```
reg_rd1 : out Std_Logic_Vector(31 downto 0);
radr1   : in Std_Logic_Vector(3  downto 0);
reg_v1  : out Std_Logic;
```

— *Read Port 2 32 bits*

```
reg_rd2 : out Std_Logic_Vector(31 downto 0);
radr2   : in Std_Logic_Vector(3  downto 0);
reg_v2  : out Std_Logic;
```

— *Read Port 3 32 bits*

```
reg_rd3 : out Std_Logic_Vector(31 downto 0);
radr3   : in Std_Logic_Vector(3  downto 0);
reg_v3  : out Std_Logic;
```

Interface 3 CSPR

```
-- read CSPR Port  
reg_cry : out Std_Logic;  
reg_zero : out Std_Logic;  
reg_neg : out Std_Logic;  
reg_cznv : out Std_Logic;  
reg_ovr : out Std_Logic;  
reg_vv : out Std_Logic;
```

Interface 4 invalidation

```
-- Invalidate Port  
inval_adr1 : in Std_Logic_Vector(3 downto 0);  
inval1 : in Std_Logic;  
  
inval_adr2 : in Std_Logic_Vector(3 downto 0);  
inval2 : in Std_Logic;  
  
inval_czn : in Std_Logic;  
inval_ovr : in Std_Logic;
```

Interface 5 PC etc...

— *PC*

```
reg_pc : out Std_Logic_Vector(31 downto 0);  
reg_pcv : out Std_Logic;  
inc_pc : in Std_Logic;
```

— *global interface*

```
ck : in Std_Logic;  
reset_n : in Std_Logic;  
vdd : in bit;  
vss : in bit);
```