Real Time Embedded Systems

"System On Programmable Chip"

Programmable interface design

Specific Counter Design

René Beuchat

Laboratoire d'Architecture des Processeurs

rene.beuchat@epfl.ch

- We want to realize a specific counter with few features only:
  - Reset by command
  - Count on 32 bits
  - Start and stop counting by command

  - For an Avalon bus slave interface
  - This simple design could be used as starting point for more complex timer units

NIOS II – A specific Avalon Peripherals
Counter as exercise

- 32 bits counter
- Avalon Bus Slave Interface, 32 bits data
- Command, 3 separate Write access:
  - RzCount
  - StartCounter
  - StopCounter
- Read Counter Value on 32 bits
- Status:
  - Run
- Interrupt (option)
  - Enable Interrupt: Command
  - EOT (EndOfTime (Counter→0)): Status
  - ClearEOT: Status

NIOS II – A specific Avalon Peripherals
Counter as exercise

- **Command**, don't care of data
  just use the Write address :
  - RzCount
  - StartCounter
  - StopCounter
- **Command** register
  - Enable Interrupt Command, 1 bit

- **Status**:
  - Run
  - EOT (EndOfTime (Counter→0)): Status
  - To Clear EOT, write a '1' to Status bit EOT

- Address mapping:

Internal Reg. Add	Name	R/W	Size	Function
0	Counter	R	32	Counter value
1	Rz	W	0	Reset Counter
2	Start	W	0	Start Counting
3	Stop	W	0	Stop Counting
4	Command	W/R	8	General Command
5	Status	W/R	8	General Status

- Command register:

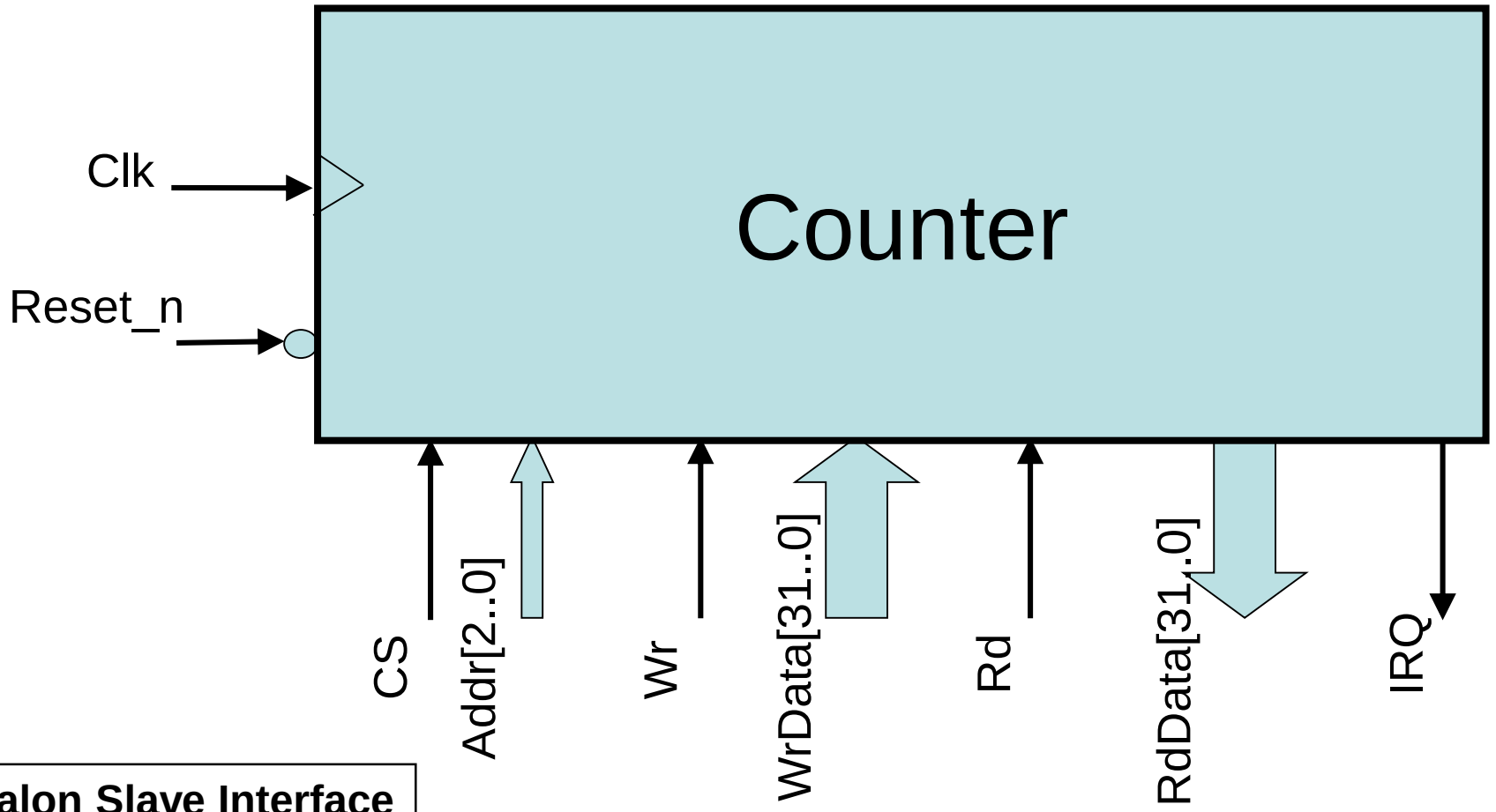
Command	7	6	5	4	3	2	1	0
Functions	-	-	-	-	-	-	-	**IRQEn**
@Reset	0	0	0	0	0	0	0	0

- Status register:

Status	7	6	5	4	3	2	1	0
Functions R W	-	-	-	-	-	-	Run	**EOT** **ClrEOT**
@Reset	0	0	0	0	0	0	0	0

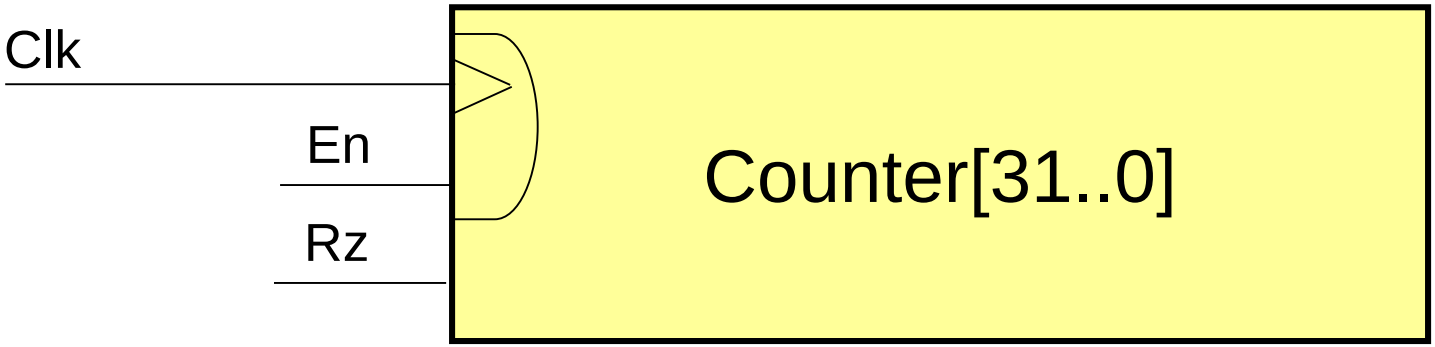
- More details on Avalon Bus Module:
  - Slave
  - 32 bits counter
  - Read and write Access on 32 bits bus
  - **0 wait Write** access
  - **1 wait Read** access
  - Synchronous to **Clk**
  - **Reset_n** to initialize the module
  - ***IRQ*** for interrupt Request

Bloc diagram of the interface



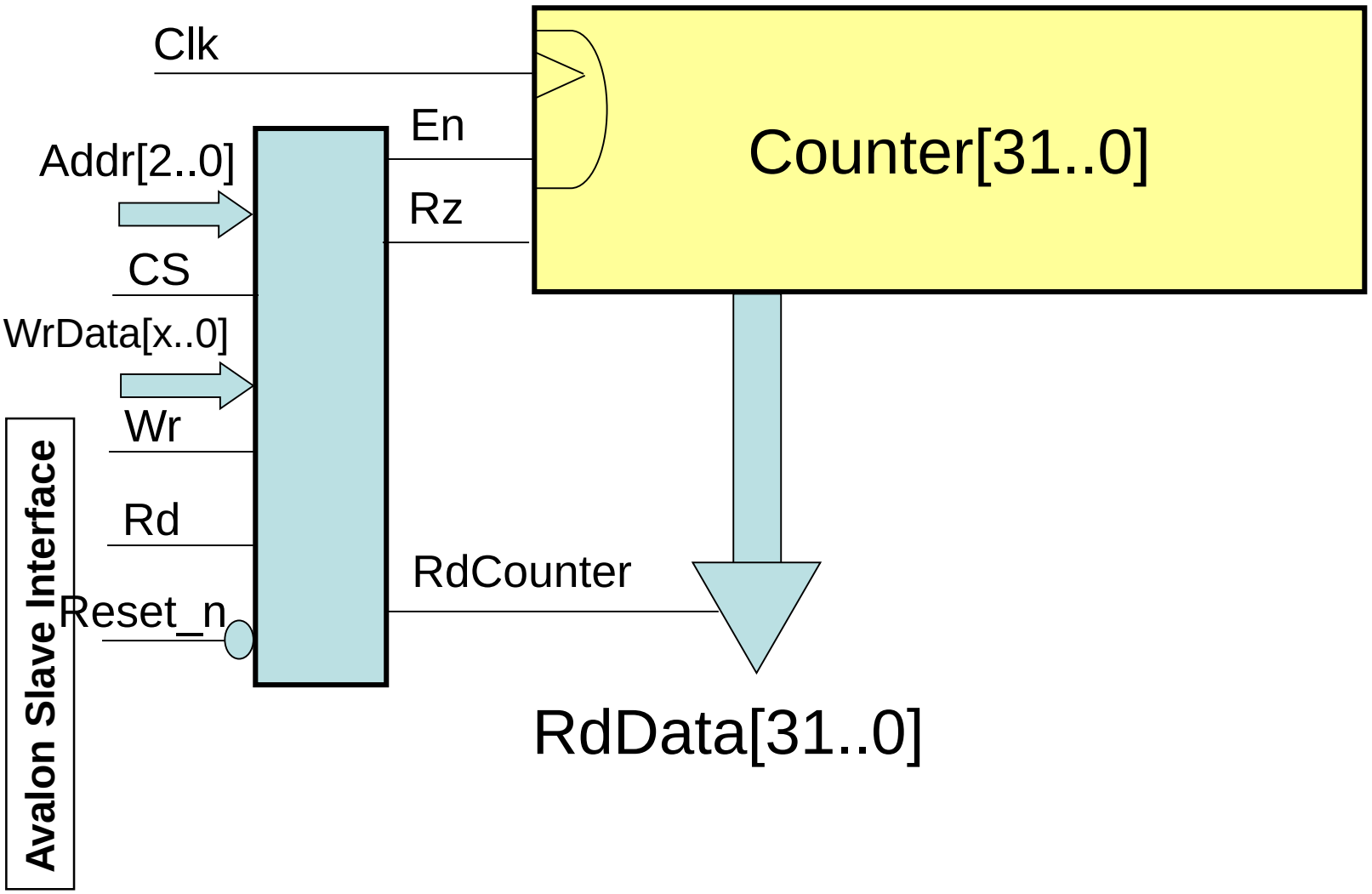
Avalon Slave Interface

NIOS II – A specific Avalon Peripherals
Counter as exercise: Base, a 32 bits counter

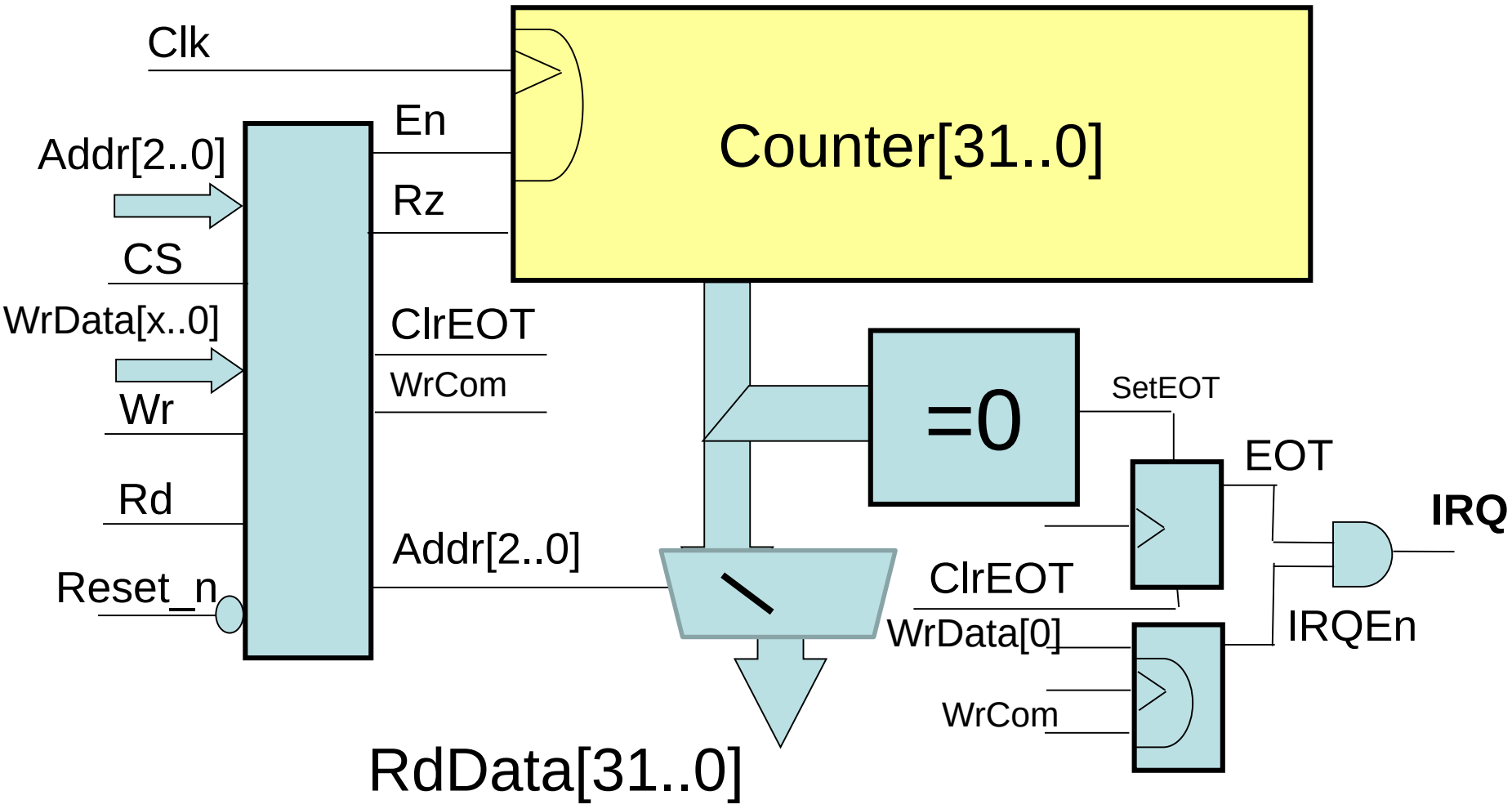


NIOS II – A specific Avalon Peripherals

Counter as exercise: Avalon interface for Read and basic commands



NIOS II – A specific Avalon Peripherals
Counter as exercise: IRQ complement



Note: Read Status not represented

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

Or (but obsolete !!!)
  use ieee.std_logic_arith.all;
  use ieee.std_logic_unsigned.all;
```

NIOS II – A specific Avalon Peripherals
Counter as exercise: Entity

```
ENTITY Counter IS
   PORT(
   -- Avalon interfaces signals
      Clk           : IN     std_logic;
      nReset        : IN     std_logic;

      Address       : IN     std_logic_vector (2 DOWNTO 0);
      ChipSelect    : IN     std_logic;

      Read          : IN     std_logic;
      Write         : IN     std_logic;

      ReadData      : OUT    std_logic_vector (31 DOWNTO 0);
      WriteData     : IN     std_logic_vector (31 DOWNTO 0);

   -- Interruptions
      IRQ           : OUT    std_logic
   );
End Counter;
```

Counter as exercise: Architecture: Internal signals

```
-- signals for register access

  iCounter  : unsigned(31 DOWNTO 0);

  iEn       : std_logic;
  iRz       : std_logic;
  iEOT      : std_logic;
  iClrEOT   : std_logic;
  iIRQEn    : std_logic;
```

NIOS II – A specific Avalon Peripherals
Counter as exercise: Counter Architecture, Counter

```
ARCHITECTURE comp OF Counter IS
... SIGNAL ...
BEGIN
 -- Counter process, synchronous Reset by command and count if
    enabled
 pCounter:
    process(Clk)
    begin
      if rising_edge(Clk) then
         if iRz= '1' then
            iCounter <= (others => '0');      -- Reset Counter → 0
         elsif iEn = '1' then
            iCounter <= iCounter+1;           -- increment
         end if;
      end if;
    end process pCounter;
END comp;
```

NIOS II – A specific Avalon Peripherals

Counter as exercise: Counter Architecture, registers access

```
pRegWr:                                 -- Process Write to registers
    process(Clk, nReset)
    begin
        if nReset = '0' then            -- asynchronous Reset
            iEn <= '0';                 -- No Count by default
            iRz <= '0';
            iIRQEn <= '0';              -- No IRQ Enable by default
        elsif rising_edge(Clk) then
            iRz <= '0';                 -- No Reset, as it's just a command
            iClrEOT <= '0' ;
            if ChipSelect = '1' and Write = '1' then        -- Write cycle
                case Address(2 downto 0) is
                    when "000" => null ;
                    when "001" => iRz <= '1';       -- Reset Counter (pulse)
                    when "010" => iEn <= '1';       -- Start Run
                    when "011" => iEn <= '0';       -- Stop Run
                    when "100" => iIRQEn <= WriteData(0);
                    when "101" => iClrEOT <= WriteData(0);
                    when others => null;
                end case;
            end if;
        end if;
    end process pRegWr;
```

NIOS II – A specific Avalon Peripherals
Counter as exercise: Counter Architecture, registers access

```
-- Read registers Process nead 1 WaitCycle on Avalon
pRegRd:
    process(Clk)
    begin
        if rising_edge(Clk) then
           ReadData <= (others => '0');                -- default value
           if ChipSelect = '1' and Read = '1' then     -- Read cycle
                case Address(2 downto 0) is
                  when "000" =>  ReadData <= std_logic_vector(iCounter);
                                                       -- cast and Read Counter
                  when "100" =>  ReadData(0) <= iIRQEn;
                  when "101" =>  ReadData(0) <= iEOT;          -- EOT
                                 ReadData(1) <= iEn; -- Run
                  when others => null;
                end case;
             end if;
        end if;
    end process pRegRd;
```

Counter as exercise: Counter Architecture, Interrupt

```
-- Interrupt Process , End Of Time activated when counter is @0, reste by a
   command from pRegWr
pInterrupt:
    process(Clk)
    begin
        if rising_edge(Clk) then
             If iCounter = X"0000_0000" then
               iEOT <= '1';          -- Flag End Of Time → '1' when counter =0
             Elsif iClrEOT = '1' then
               iEOT <= '0';          -- Cleared on access by WrStatus(0)
             End if;
        end if;
    end process pInterrupt;

-- IRQ activation
IRQ <= '1' when iEOT ='1' and iIRQEn = '1' and iEn = '1'else
           '0';
```

- From this description and code, it is very easy to add others features as:
- Output Compare function
- Interrupt at specific time
- Reload counter
- Etc…