Reconfigurable Systems

Jorge Semião

Instituto Superior de Engenharia

www.ualg.pt

Introduction HDL (Hardware Descrip. Language)

- Hardware description languages (HDL)
  - Language to describe hardware
  - Two popular languages
    - VHDL: **V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage
      - Developed by DOD (US Department Of Defence) from 1983
      - IEEE Standard 1076-1987/1993/200x
      - Based on the ADA language
    - Verilog
      - IEEE Standard 1364-1995/2001/2005
      - Based on the C language

Applications of HDL

- Model and document digital systems
  - Different levels of abstraction
    - Behavioral, structural, etc.
- Verify design
- Synthesize circuits
  - Convert from higher abstraction levels to lower abstraction levels

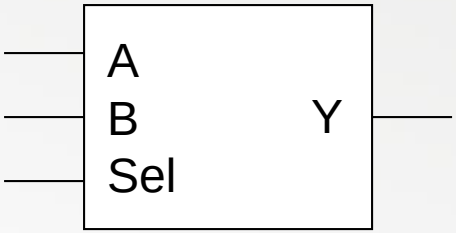
Introduction to VHDL

- VHSIC Hardware Discription Language
  - VHSIC Very High Speed Integrated Circuit
  - Industrial Standard (IEEE-1076)
  - portability of designs
  - hierarchy in design description
  - technology independence

VHDL as a language

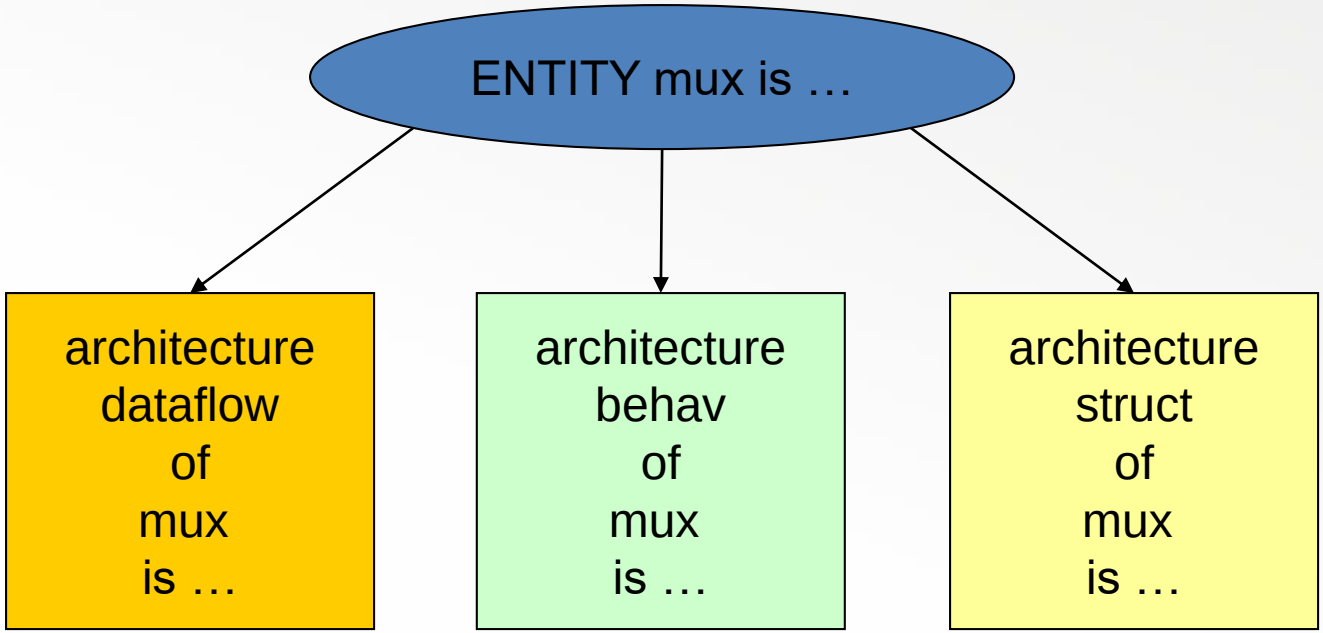
- strongly typed
  - type check at compile time
  - allows user defined types
  - case insensitive
  - two consecutive dashes (--) is used for comment

Level of Abstraction in VHDL

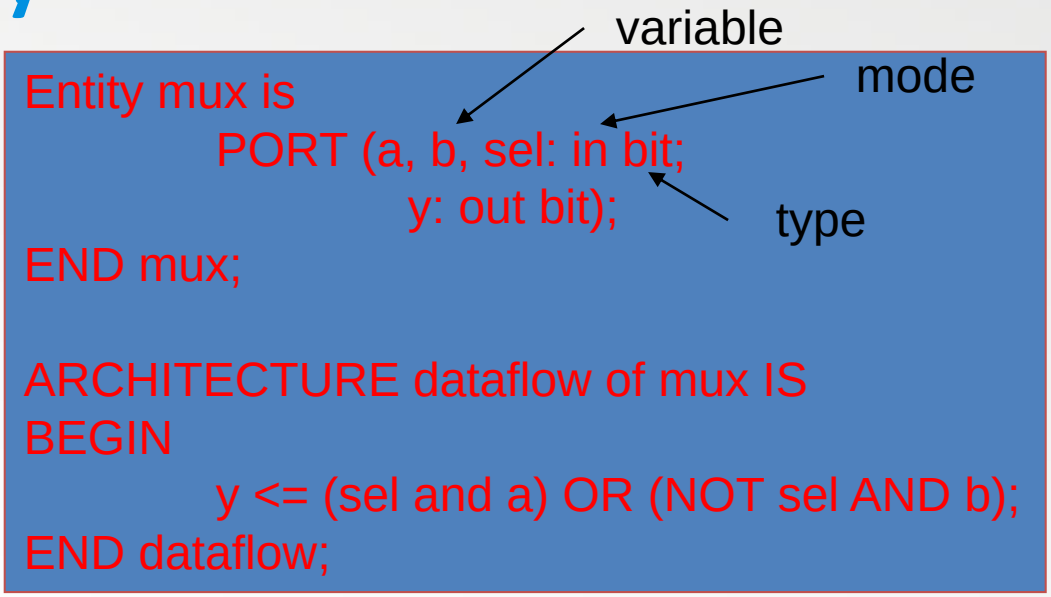



```
if Sel == 0
        Y <= A;
else
        Y <= B;
```

The block diagram of mux



Entity



set of concurrent assignments to represent dataflow

Architecture Body

- Four modeling styles
  - concurrent assignments (dataflow)
  - interconnected component (structure)
  - sequential assignment statements (behavior)
  - combination of the above three

```
Example:
ARCHITECTURE dataflow OF mux IS

-- signals, variables declaration

BEGIN
…
END dataflow;
```

Signals vs Variables

- Signals
  - Signals follow the notion of 'event scheduling'
  - An event is characterized by a (time,value) pair
  - Signal assignment example:

    `X <= Xtmp;` means

    Schedule the assignment of the value of signal `Xtmp` to signal `X` at (Current time + delta)

    where delta: infinitesimal time unit used by simulator for processing the signals

Signals vs Variables

- Variables
  - Variables do not have notion of 'events'
  - Variables can be defined and used only inside the **process** block and some other special blocks.
  - Variable declaration and assignment example:

```
process (…)
variable K : bit;
 begin
    …
  -- Assign the value of sign
immediately
    K := L;
  …
end process;
```

Variables can only be defined and used inside the **process** construct and can be defined only in this place

Dataflow

- concurrent signal assignment statements include:
  - simple signal assignment (<=)
  - select signal statement
  - conditional signal statement

Dataflow

- simple signal assignment

  y <= (sel AND a) OR (NOT sel AND b);

  -- y : target signal

  -- <= : signal assignment operator

  – six logical operators are
    - AND OR NAND NOR XOR NOT
  – relational operators
    - =, /=, <, <=, >, >=

Dataflow

- Selected Signal Assignment
  - concurrent signal

```
ENTITY mux_df IS
        PORT (a, b, sel: in bit;
                y: out bit);
END mux_df;

ARCHITECTURE dataflow of mux_df IS
BEGIN
        WITH sel SELECT
                y <= a when '1',
                     b when '0';
END dataflow;
```

Dataflow

```
Entity mux2_df is
        PORT (  data : in BIT_VECTOR(3 DOWNTO 0);
                sel: in INTEGER RANGE 0 to 3;
                f: out bit);
END mux2_df;

ARCHITECTURE dataflow of mux2_df IS
BEGIN
        WITH sel SELECT
                f <= data(0) when 0,
                     data(1) when 1,
                     data(2) when 2,
                     data(3) when 3;
END dataflow;
```

BIT_VECTOR is an array of bits which is of type BIT

Dataflow

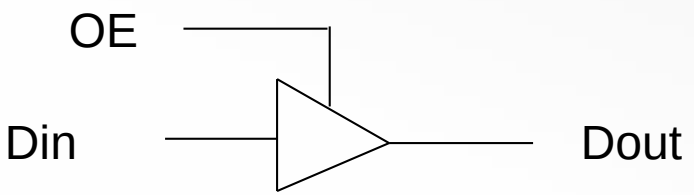
```
library ieee;
use ieee.std_logic_1164.all;

entity dcd_3_8 is
        port (      a: in std_logic_vector(2 downto 0);
                    s: out std_logic_vector(7 downto 0)
        );
end dcd_3_8;

architecture dataflow of dcd_3_8 is
begin
        with a SELECT
                s <= "00000001" when "000",
                "00000010" when "001" | "00Z",
                "00000100" when "010" | "0Z0",
                "00001000" when "011" | "0ZZ",
                "00010000" when "100" | "Z00",
                "00100000" when "101" | "Z0Z",
                "01000000" when "110" | "ZZ0",
                "10000000" when "111" | "ZZZ",
                "xxxxxxxx" when OTHERS;
end dataflow;
```

Dataflow

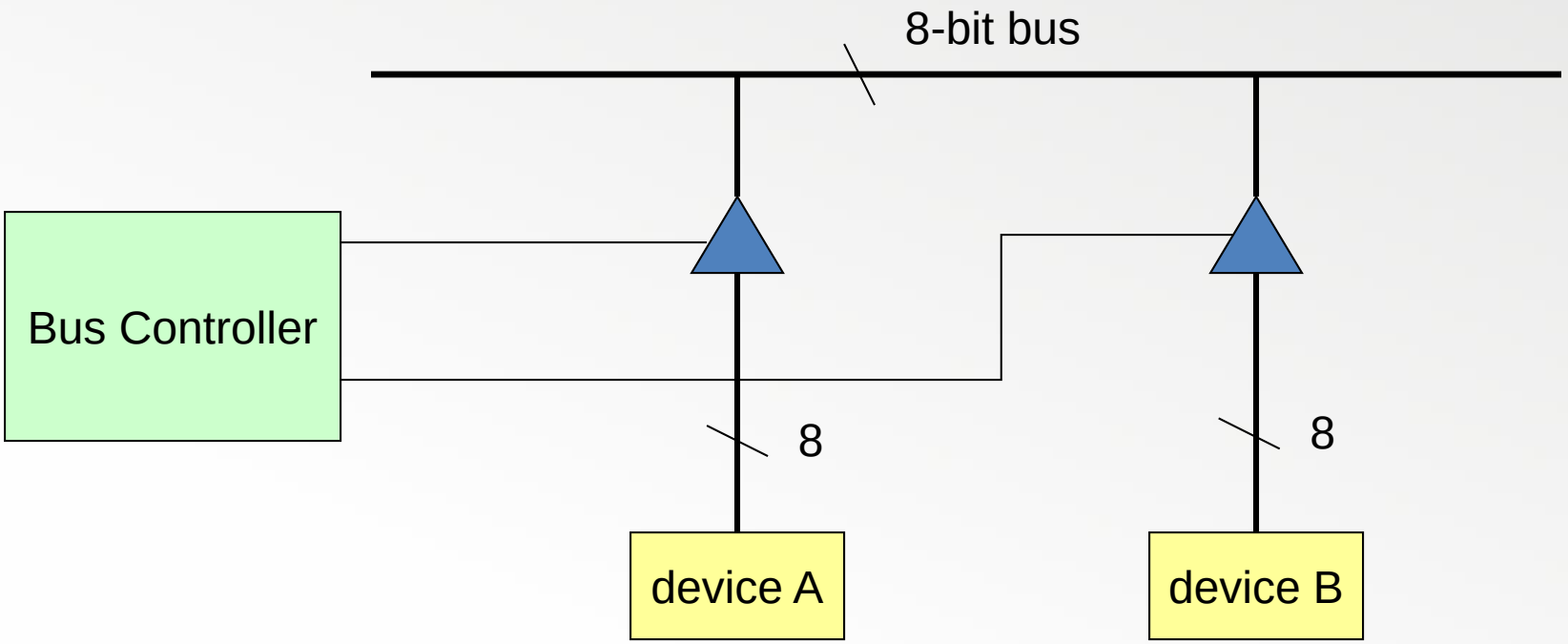
type STD_LOGIC is (
‘U’ – uninitialized
‘X’ – Forcing unknown
‘0’ – Forcing low
‘1’ – Forcing high
‘Z’ – High Impedance
‘W’ – Weak Unknown
‘L’ – Weak Low
‘H’ – Weak High
‘-’ – Don’t Care
);



Tri-state Buffer

Din	OE	Dout
X	0	Z
0	1	0
1	1	1

Dataflow



Dataflow

Conditional Signal Assignment

```
ENTITY mux_df2 IS
        port ( a, b, sel : in bit;
                    y : out bit );
END mux_df2;

ARCHITECTURE dataflow of mux_df2 is
BEGIN
        y <= a WHEN sel = '1' ELSE b;
END dataflow;
```

Dataflow

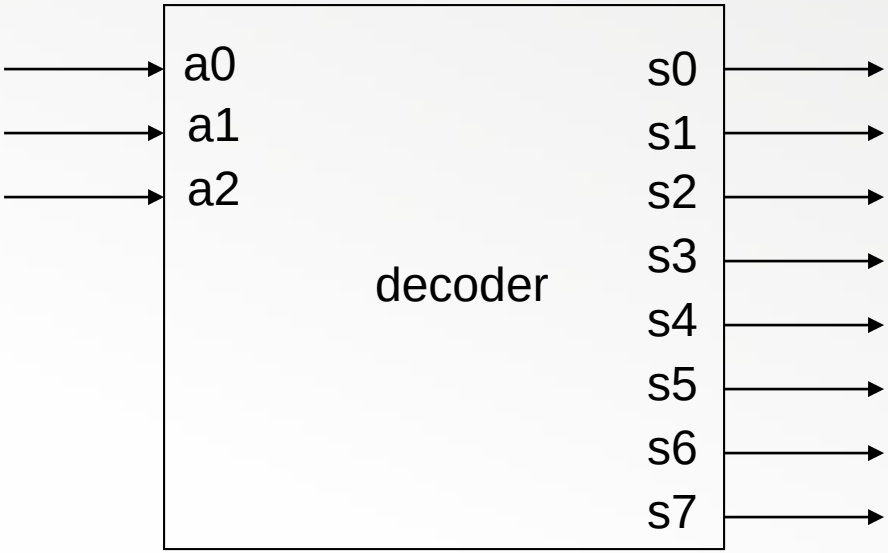
```
Entity mux2_df is
        PORT (  data : in BIT_VECTOR(3 DOWNTO 0);
                sel: in INTEGER RANGE 0 to 3;
                f: out bit);
END mux2_df;

ARCHITECTURE dataflow of mux2_df IS
BEGIN
        f <= data(0) when sel = 0 ELSE,
                data(1) when sel = 1 ELSE,
                data(2) when sel = 2 ELSE,
                data(3);
END dataflow;
```

When a conditional signal assignment statement is simulated, each Boolean expression is tested in the order that it was written.

Dataflow

- Decoder is another example of selected signal assignment construct



Behavioral Descriptions

- contain “PROCESS” statement.
  - statements appearing inside a process are simulated sequentially. However, a process statement is a concurrent statement.
  - also contain a sensitivity list or WAIT statement.
  - variables can be used locally (keyword VARIABLE).
    - := is used for a variable assignment.
- can contain multiple processes.
  - signals are used for communication
  - information can only be transferred between processes through signals.

Behavioral

```
ENTITY mux_proc IS port (a, b, sel: in bit;
                         y: out bit);
END mux_proc;
architecture behv of mux_proc is
begin
PROCESS (a, b, sel)
   BEGIN
       IF(sel = '0') THEN
            y <= b;
       ELSIF(sel = '1') THEN
            y <= a;
       END IF;
    END PROCESS;
END behv;
```

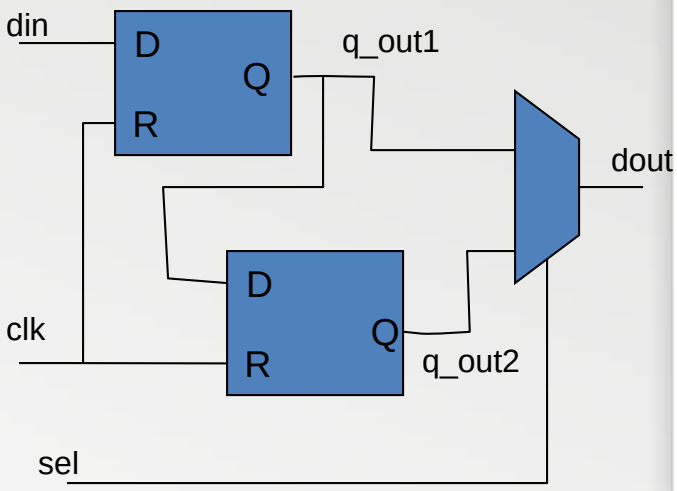
Behavioral

```
ENTITY unknown IS port (a, b: in bit;
                        y, z: out bit);
END unknown;
architecture behv of unknown is
begin
PROCESS (a, b)
   BEGIN
       IF((a or b) = '0') THEN
            y <= '1';
       ELSE
            z <= a or b;
       END IF;
    END PROCESS;
END behv;
```

Look at differences between simulation and synthesis.

Behavioral

```
ENTITY dff_mux2 IS
        port (reset, clock, din, sel: in std_logic;
                dout : out std_logic);
END dff_mux2;
architecture inference of dff_mux2 is
  signal q_out1, q_out2: std_logic;
  begin
    PROCESS -- no sensitivity list here
      BEGIN
        WAIT_UNTIL (clock'EVENT and clock = '1');
        q_out1 <= din;
        q_out2 <= q_out1;
      END PROCESS;
    dout <= q_out1 when sel = '1' else q_out2;
END inference;
```



Built-in Datatypes

- Scalar (single valued) signal types:
  - `bit`
  - `boolean`
  - `integer`
  - Examples:
    - `A: in bit;`
    - `G: out boolean;`
    - `K: out integer range -2**4 to 2**4-1;`
- Aggregate (collection) signal types:
  - `bit_vector`: array of bits representing binary numbers
  - `signed`: array of bits representing signed binary numbers
  - Examples:
    - `D: in bit_vector(0 to 7);`
    - `E: in bit_vector(7 downto 0);`
    - `M: in signed (4 downto 0);`
      `--signed 5 bit_vector binary number`

User-defined datatype

- Construct datatypes arbitrarily or using built-in datatypes
- Examples:
  - `type temperature is (high, medium, low);`
  - `type byte is array(0 to 7) of bit;`

VHDL Architecture

- VHDL description (sequential behavior):

```
architecture arch_name of my_ckt is
begin
 p1: process (A,B,S)
 begin
  if (S='0') then
   X <= A;
  else
   X <= B;
  end if;

  if ((X = '0') and (S = '0')) then
   Y <= '1';
  else
   Y <= '0';
  end if;

 end process p1;
end;
```

Error: Signals defined as output ports can only be driven and not read

VHDL Architecture

```
architecture behav_seq of my_ckt is

    signal Xtmp: bit;

    begin
     p1: process (A,B,S,Xtmp)
       begin
      if (S='0') then
       Xtmp <= A;
      else
       Xtmp <= B;
      end if;

      if ((Xtmp = '0') and (S = '0')) then
       Y <= '1';
      else
       Y <= '0';
      end if;

      X <= Xtmp;
     end process p1;
end;
```

Signals can only be defined in this place before the **begin** keyword

General rule: Include all signals in the sensitivity list of the process which either appear in relational comparisons or on the right side of the assignment operator inside the `process` construct.

In our example:
`Xtmp` and `S` occur in relational comparisons
`A`, `B` and `Xtmp` occur on the right side of the assignment operators

VHDL Architecture

- VHDL description (concurrent behavior):

```
architecture behav_conc of my_ckt is

   signal Xtmp: bit;

begin

   Xtmp <= A when (S='0') else
           B;
   Y <= '1' when ((Xtmp = '0') and (S = '0')) else
        '0';

   X <= Xtmp;

end ;
```

Structural level netlist

```
architecture behav_conc of my_ckt is

   -- component declarations

   signal Sbar, Xbar, W1, W2: bit;

begin

   G1: not port map(Sbar,S);
   G2: and port map(W1,A,Sbar);
   G3: and port map(W2,B,S);
   G4: or  port map(X,W1,W2);

   G5: not port map(Xbar,X);
   G6: and port map(Y,Xbar,Sbar);

end ;
```

- Gate level VHDL descriptions (`and, or,` etc) are described separately
- Design in which other design descriptions are included is called a "hierarchical design"
- A VHDL design is included in current design using port map statement