Introduction to hardware design using VHDL

Tim Güneysu and Nele Mentens

ECC school

November 11, 2017, Nijmegen

Outline

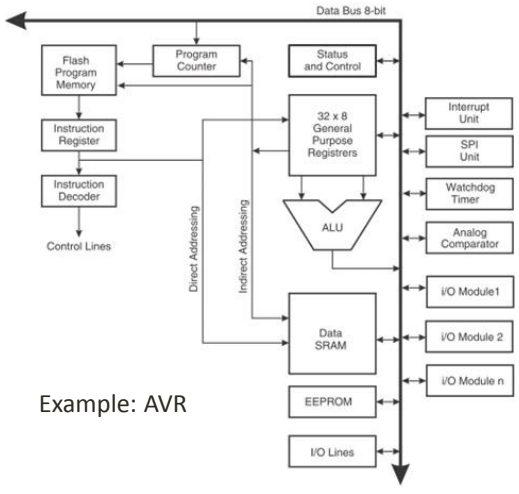
- Implementation platforms
- Introduction to VHDL
- Hardware tutorial

Implementation platforms

- Microprocessor
- FPGA = Field-Programmable Gate Array
- ASIC = Application-Specific Integrated Circuit

Implementation platforms
Microprocessor

architecture



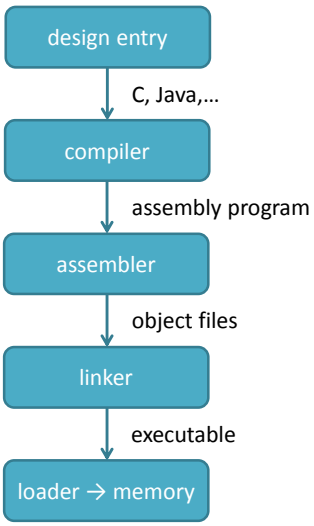
Example: AVR

The CPU is the heart of a microprocessor and contains a.o.:

- ALU (Arithmetic Logic Unit)
- register file
- program memory

Implementation platforms
Microprocessor

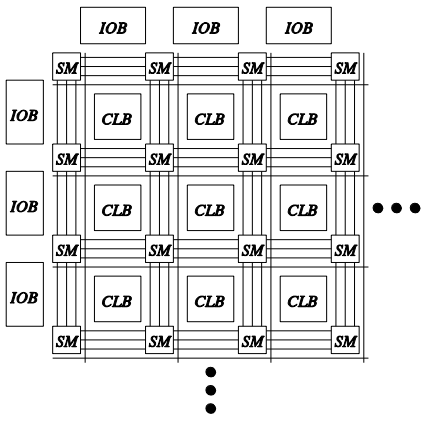
design flow



- The hardware architecture of a microprocessor is fixed
- The code describes what should be executed on the fixed hardware
- The instructions end up in the program memory

Implementation platforms
Field-Programmable Gate Arrary (FPGA)

architecture

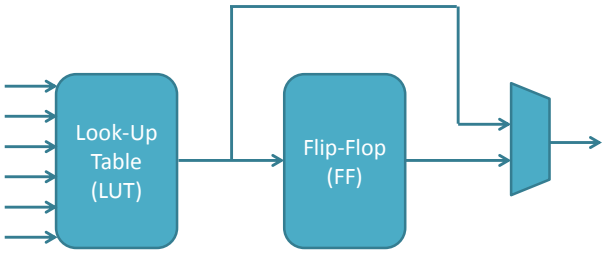


Basic components:

- CLB = Configurable Logic Block
  - CLBs consist of slices.
  - Slices consist of
    - Look-Up Tables (LUTs),
    - Multiplexers,
    - Flip-Flops (FFs),
    - Carry logic.
- SM = Switch Matrix
- IOB = Input/Output Block

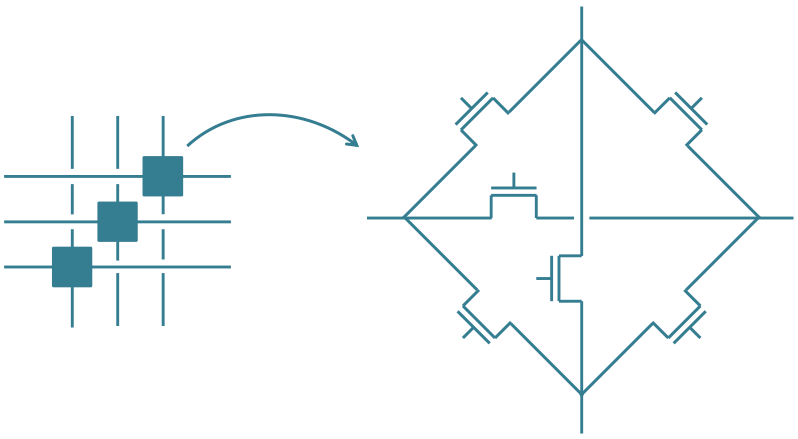
Implementation platforms
Field-Programmable Gate Arrary (FPGA)

basic content of a slice



Implementation platforms
Field-Programmable Gate Arrary (FPGA)

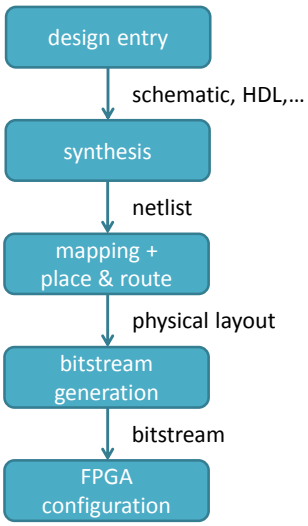
basic principle of a switch matrix



ECC school, November 11, 2017, Nijmegen

Implementation platforms
Field-Programmable Gate Arrary (FPGA)

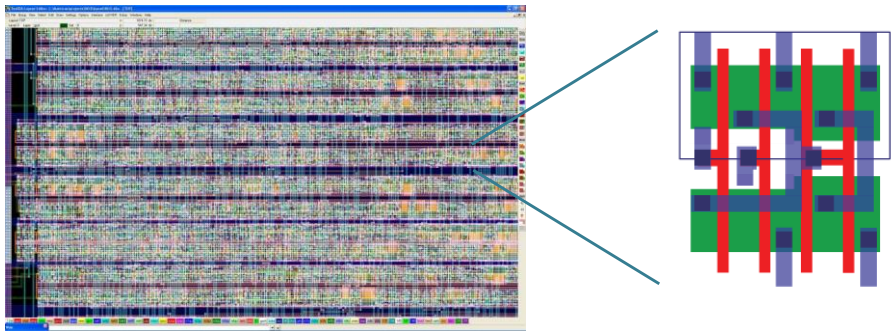
design flow



- The hardware architecture of an FPGA is configurable
- The code describes the hardware that we need
- The bitstream ends up in the configuration memory
- The area is measured in terms of occupied LUTs, flip-flops, dedicated hardware blocks

Implementation platforms
Application-Specific Integrated Circuit (ASIC)

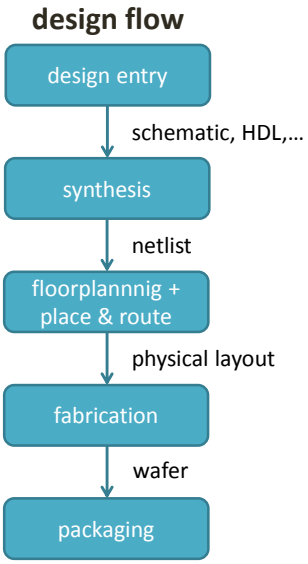
architecture



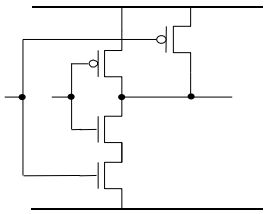
Basic components:

- Standard cells from a standard cell library
  - Logic cells and sequential cells

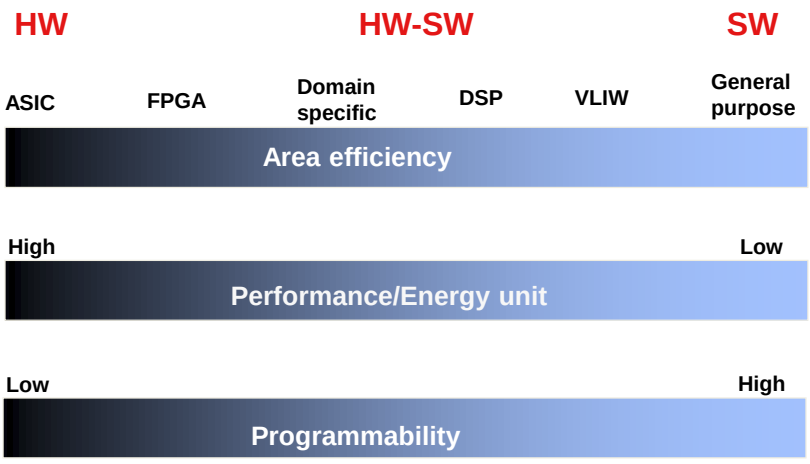
Implementation platforms
Application-Specific Integrated Circuit (ASIC)



- The hardware architecture of an ASIC is fixed
- The code describes the hardware that we need
- The GDS file contains the physical information that goes to the foundry
- The area is measured in terms of the number of equivalent NAND gates (Gate Equivalent = GE)



Implementation platforms
Comparison



Introduction to VHDL
Standard

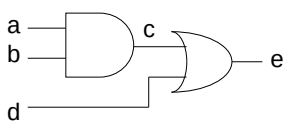
- VHDL (VHSIC Hardware Description Language)
  - VHSIC = Very High Speed Integrated Circuit
- International standard
  - First standard: IEEE 1076-1987
  - Most recent update: IEEE 1076-2008

Introduction to VHDL
Hardware vs. software

- Description language for hardware ≠ programming language
- Programming language (e.g. C):
  - hardware = processor
  - hardware is already designed, implemented and fabricated
  - code: describes how the hardware will be used
  - code is compiled for a specific processor
- Hardware description language (e.g. VHDL)
  - hardware = FPGA or ASIC design
  - hardware is designed
  - code: describes which hardware will be designed
  - code is synthesized for a specific FPGA or ASIC technology
  - example:

```
c <= a and b;
e <= c or d;
```

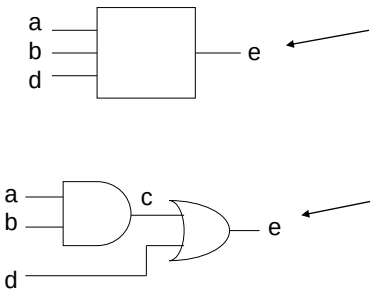
```
e <= c or d;
c <= a and b;
```



2x the same implementation

Introduction to VHDL
Entities and architectures

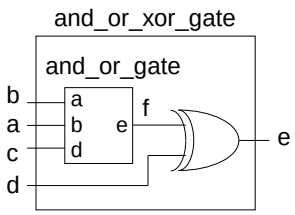
- The VHDL code of each component consists of
  - an interface description: entity,
  - a behavioral description: architecture.
- Example:




```
entity and_or_gate is
      port(      a, b, d: in bit;
                 e: out bit);
end and_or_gate;

architecture arch of and_or_gate is
      signal c: bit;
begin
      c <= a and b;
      e <= c or d;
end arch;
```


Introduction to VHDL
Hierarchy

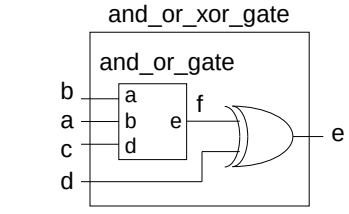
- Hierarchy can be built in.
- There is hierarchy when a component contains an instantiation of another component.




```
entity and_or_xor_gate is
      port(a, b, c, d: in bit;
           e: out bit);
end and_or_xor_gate;

architecture arch of and_or_xor_gate is
      component and_or_gate is
            port(a, b, d: in bit;
                 e: out bit);
      end component;
      signal f: bit;
begin
      inst_and_or_gate: and_or_gate
            port map(a => b,
                     b => a,
                     d => c,
                     e => f);
      e <= d xor f;
end arch;
```

Introduction to VHDL
Hierarchy

- Hierarchy can be built in.
- There is hierarchy when a component contains an instantiation of another component.




```
inst_and_or_gate: and_or_gate
      port map(b, a, c, f);
```

order must be correct

identical

```
entity and_or_xor_gate is
      port( a, b, c, d: in bit;
            e: out bit);
end and_or_xor_gate;

architecture arch of and_or_xor_gate is
      component and_or_gate is
            port( a, b, d: in bit;
                  e: out bit);
      end component;
      signal f: bit;
begin
      inst_and_or_gate: and_or_gate
            port map(a => b,
                     b => a,
                     d => c,
                     e => f);
      e <= d xor f;
end arch;
```

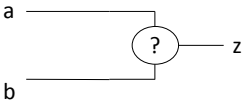

Introduction to VHDL
bit vs. std_logic

- The package “std_logic_1164” in library “ieee” contains a.o. the types “std_ulogic” en “std_logic”, consisting of 9 values (instead of 2 for “bit”)

```
type std_ulogic is (    'U',   -- Uninitialized
                        'X',   -- Forcing Unknown
                        '0',   -- Forcing 0
                        '1',   -- Forcing 1
                        'Z',   -- High Impedance
                        'W',   -- Weak Unknown
                        'L',   -- Weak 0
                        'H',   -- Weak 1
                        '-',   -- Don't Care);
subtype std_logic is resolved std_ulogic;

type std_ulogic_vector
      is array (NATURAL range <>) of std_ulogic;
type std_logic_vector
      is array (NATURAL range <>) of std_logic;
```

example

```
signal a, b, z: std_logic;
...
z <= a;
z <= b;
```



- It is advised to always use “std_logic” instead of “bit”

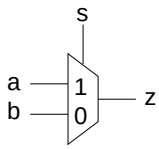
Introduction to VHDL
Concurrent and sequential statements

- Concurrent statements: are implement in parallel and executed at the same time
- Sequential statements: can only occur in a process

example:

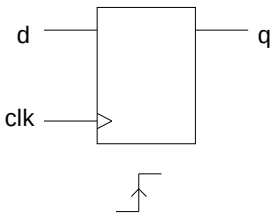
```
entity mux is
        port( a, b, s: in std_logic;
              z: out std_logic);
end mux;

architecture arch of mux is
begin
        p1: process(a, b, s)
        begin
                if s = '1' then
                        z <= a;
                else
                        z <= b;
                end if;
        end process;
end arch;
```

sensitivity list



Introduction to VHDL
Storage elements

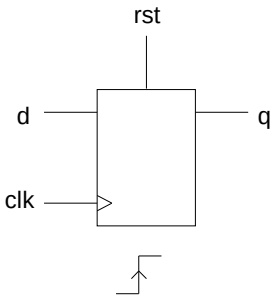
- D-flipflop:




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

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

architecture arch of dff is
begin
        store: process(clk)
        begin
                if clk'event and clk = '1' then
                        q <= d;
                end if;
        end process;
end arch;
```

Introduction to VHDL
Storage elements

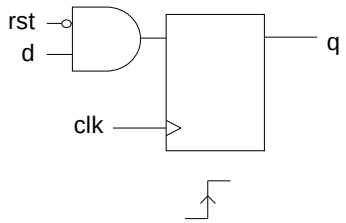
- D-flipflop with asynchronous reset:




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

entity dff is
        port( d, clk, rst: in std_logic;
                q: out std_logic);
end dff;

architecture arch of dff is
begin
        store: process(rst, clk)
        begin
                if rst = '1' then
                        q <= '0';
                elsif clk'event and clk = '1' then
                        q <= d;
                end if;
        end process;
end arch;
```


Introduction to VHDL
Storage elements

- D-flipflop with synchronous reset:

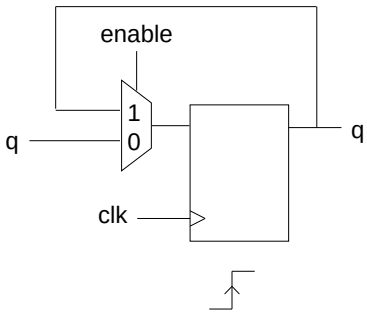



```
library ieee;
use ieee.std_logic_1164.all;

entity dff is
        port( d, clk, rst: in std_logic;
                q: out std_logic);
end dff;

architecture arch of dff is
begin
        store: process(clk)
        begin
                if clk'event and clk = '1' then
                        if rst = '1' then
                                q <= '0';
                        else
                                q <= d;
                        end if;
                end if;
        end process;
end arch;
```

ECC school, November 11, 2017, Nijmegen

Introduction to VHDL
Storage elements

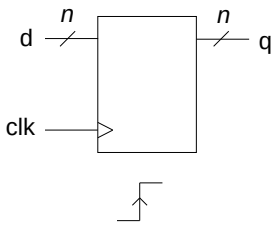
- D-flipflop with enable:




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

entity dff is
        port( d, clk, enable: in std_logic;
                q: out std_logic);
end dff;

architecture arch of dff is
begin
        store: process(clk)
        begin
                if clk'event and clk = '1' then
                        if enable = '1' then
                                q <= d;
                        end if;
                end if;
        end process;
end arch;
```


Introduction to VHDL
Modules with parameters

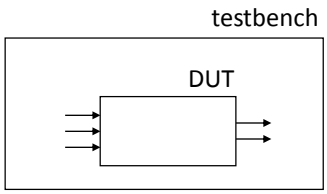
- Register with a parameterizable width:




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

entity ffn is
        generic(size: integer:=4);
        port( clk: in std_logic;
                d: in std_logic_vector(size-1 downto 0);
                q: out std_logic_vector(size-1 downto 0));
end ffn;

architecture arch of ffn is
begin
        p: process(clk)
        begin
                if clk'event and clk = '1' then
                        q <= d;
                end if;
        end process;
end arch;
```

Simulation

- A VHDL module can be simulated with a testbench:
  - Also written in VHDL
  - No ports in the entity
  - Containing an instantiation of the device under test (DUT)
- Input signals are applied internally in the testbench
- Output signals are evaluated
  - Through waveforms in a simulation window
  - In a text file



Hardware tutorial

- 4-bit adder
- n-bit adder
- 4-bit modular adder
- EXERCISE: n-bit modular adder
- n-bit modular adder/subtracter
- n-bit modular constant multiplier (multiplication by 5)
- EXERCISE: n-bit modular multiplier
  - through consecutive additions
- EXERCISE: n-bit modular multiplier
  - through left-to-right modular double-and-add
- 4xn-bit register file
- EXERCISE: elliptic curve point doubling

Hardware tutorial

- For each module, the VHDL code for the module and the VHDL code for the testbench are given
- Where it says EXERCISE, the VHDL code for the module needs to be completed
- The tutorial will cover synthesis and post-synthesis (behavioral) simulation