OUTLINE

- Top-down design
- VHDL history
- Main elements of VHDL
  - Entities and architectures
  - Signals and processes
  - Data types
  - Configurations
- Simulator basics
- The testbench concept

GAJSKI’S Y-CHART (1983)

BEHAVIORAL DOMAIN
- Systems
- Algorithms
- Register transfers
- Logic
- Transfer functions

STRUCTURAL DOMAIN
- Processors
- ALUs, RAMs, etc.
- Gates, flipflops, etc.
- Transistors

PHYSICAL DOMAIN
- Transistor layout
- Cell layout
- Module layout
- Floorplan

TOP-DOWN DESIGN (1)

BEHAVIORAL DOMAIN

STRUCTURAL DOMAIN

Top-down structural decomposition + bottom-up layout construction

PHYSICAL DOMAIN
TOP-DOWN DESIGN (2)

Communication does not change!

Structural decomposition

EXAMPLE HARDWARE: sis08

• Serial in, serial out
• 8-bit data
• Environment provides new data_in when req is high
• Environment should collect data_out when ready is high.
• req, ready, and data_out are updated on rising edge of clk.
• data_in is consumed (and should be stable) on rising edge of clk.

ENTITY AND ARCHITECTURE

• VHDL directly supports the concept of one communication interface for a block with multiple descriptions of its contents.
• Entity: declaration of communication
• Architecture: declaration of content: behavior, structure or mix
• One entity can have multiple architectures.
• Check files:
  - sis08_ent.vhd
  - sis08_copy_arch.vhd
  - sis08_gcd_arch.vhd
DESCRIBING REGISTERS

- All registers: positive edge-triggered flipflops (copying data from input to output on rising clock edge) with asynchronous reset:

```
seq: process(clk, reset)
begin
    if (reset = '1')
    then
        data_out <= (others => '0');
        ready <= '0';
    elsif rising_edge(clk)
    then
        data_out <= data_in;
        ready <= '1';
    end if;
    end process seq;
```

LIBRARIES, PACKAGES, SIGNALS AND PROCESSES

- Points of attention:
  - Library: precompiled design information, grouped by a library name.
  - Package: design information of general nature to be used in more specific designs. Example: a collection of data types and functions that operate on them.
  - Signal: a carrier of values that change in time, normally associated to a wire or wire bundle in hardware.
  - Process: a description of how signals change their values. Multiple processes model parallelism.

COMBINATIONAL AND SEQUENTIAL LOGIC

- Points of attention:
  - Processes can model logic with or without internal memory, sequential resp. combinational logic.
  - Sensitivity list
  - Coding style for synchronous sequential logic with asynchronous reset.

SIMULATOR BASICS

- The simulation engine of VHDL is a so-called event-driven simulator.
- Events are signal changes. An event triggers new events after some delay.
- Events take place at some time. The simulator handles events in the order of their activation time.
- In VHDL processes are activated for events on signals in their sensitivity lists. Process activation generally creates new events.
- Even when no delay is specified in the VHDL code, the simulator will introduce an infinitesimal delay, a so-called delta delay.
**SIGNALS AND VARIABLES (1)**

- A **signal** in VHDL is the equivalent of a physical wire.
- Processes communicate via signal changes.
- Signals are always declared outside a process.
- Signal assignments are indicated by `<=` and are never immediate, take at least a *delta delay*.
- Signal assignments can occur outside processes: concurrent assignment, a process on its own.

**SIGNALS AND VARIABLES (2)**

- A signal assignment corresponds to hardware: the combination of signals at the right-hand side produces the signal at the left-hand side.
- Multiple assignments to the same signals is like shorting two outputs: it is not allowed unless the signal is “resolved”.

**DATA TYPES IN VHDL**

- Few built-in data types: integers, characters, strings, time.
- Mechanism to define new data types by enumeration, arrays, records.
- Very common non-native data type: **std_logic**, defined in standardized package **std_logic_1164**, compiled in library **ieee** (including functions such as Boolean operators).
  - Associated vector type **std_logic_vector**, with constants written as strings, e.g. ”00101”.
  - **std_logic** is a resolved type, i.e. a *resolution function* computes a signal value in the case of multiple drivers.

- Processes can also have local **variables**:
  - Declared inside a process
  - Not visible to other processes
- Variable assignment is indicated by `:=` and is immediate.
- It is therefore allowed to perform multiple assignments to a variable without advancing simulation time:
  - *This does not have a direct hardware equivalent.*
- Avoid the use of variables in the hardware that you design!
NUMERIC DATA TYPES

- A sequence of bits can be interpreted as a number.
- Multiple interpretations exist.
- The package `numeric_std`, standardized by the IEEE, available in library `ieee` defines two interpretations for `std_logic` vectors:
  - `signed`: signed number encoded as 2's complement
  - `unsigned`: positive binary numbers.
- VHDL has strong typing: assignments from `std_logic_vector` to `signed/unsigned` require type casting.

TESTBENCH (1)

- Simulation is the most common method to verify the correctness of a design, the design under verification (DUV).
- In order to simulate a design, specific signal patterns need to be supplied to the inputs.
- It is convenient to use a universal HDL to describe this signal generation, in this case VHDL.
- It is also convenient to monitor the outputs of the DUV; sometimes input signals can be generated depending on the received outputs.
- This results in a test-vector controller (TVC) with opposite I/O patterns to the DUV.

TESTBENCH (2)

- DUV + TVC = testbench (TB)
- The testbench does not have any I/O signals!
- Check files:
  - `tb_siso8.vhd`
- Note: TB has a structural architecture.

CONFIGURATIONS

- A VHDL simulator is supposed to store compiled versions of all architectures of an entity simultaneously.
- Question: how does one indicate which of the architectures should be simulated?
- Answer: VHDL configurations define which architectures to use for each entity all the way down the hierarchy.
- Check files:
  - `conf_tb_siso8_copy.vhd`
  - `conf_tb_siso8_gcd.vhd`
REPORT GUIDELINES (1)

- Upload report to Canvas.
  - Report text in PDF.
  - VHDL source files converted to PDF (all files created by you including configurations; in principle, you do not modify any of the provided files).
- Be to the point: too much and too little text are penalized.
  - Page limits are provided per project, per exercise.
  - Do not copy the entire project text into your report.
  - Do not report on results of training exercises.
- When answering questions, motivate where possible, especially motivate design choices.

REPORT GUIDELINES (2)

- Provide block diagrams to illustrate your design’s structure.
- Include waveforms appropriate to what you want to show:
  - Make a motivated selection of the signals in the design (just dumping all signals may be penalized).
  - Choose an appropriate format: binary, decimal, hexadecimal …
  - Choose an appropriate time span.
  - Make sure the waveforms are readable, also after printing on paper.

NEXT STEPS

- Go to the lab (ZI West).
- Students already registered in Canvas should have access to the Linux server. Contact the assistant if you don’t have access to Osiris/Canvas.
- Do not rush in doing the exercises. Take your time to study the handouts.
- For questions, try to contact the teaching assistant (with “Horusapp” on Thursday mornings, see slides on organization).