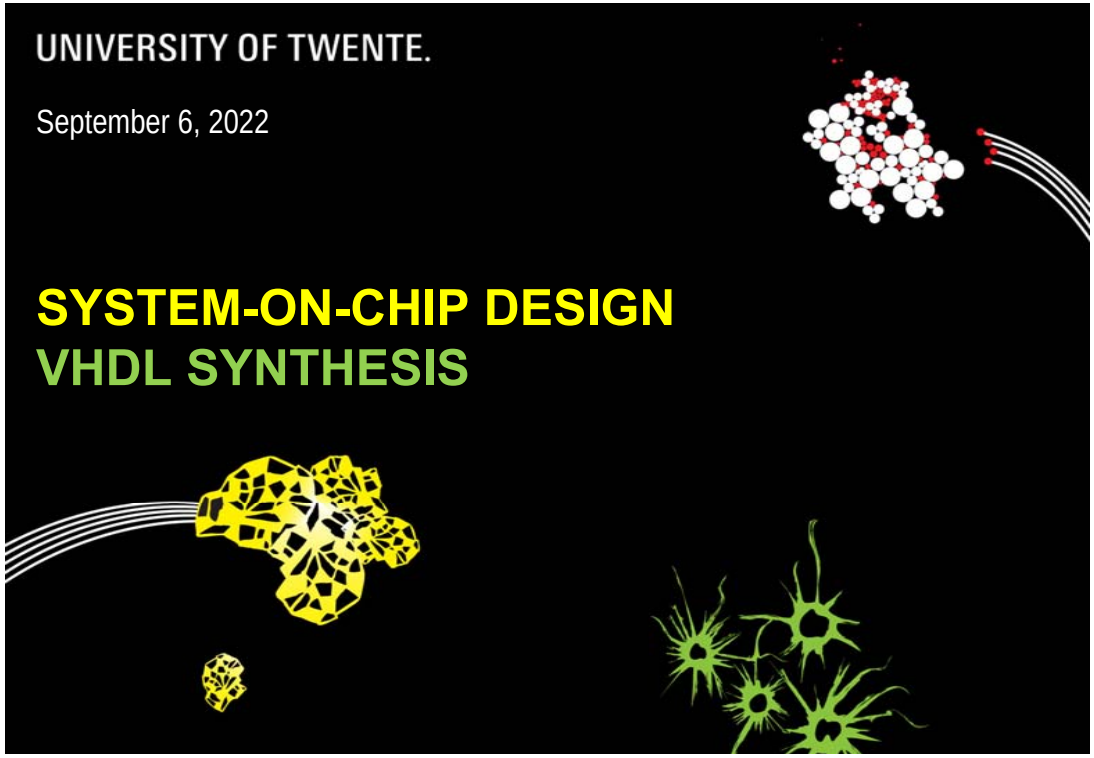
UNIVERSITY OF TWENTE.

September 6, 2022

SYSTEM-ON-CHIP DESIGN
VHDL SYNTHESIS



VHDL SYNTHESIS OUTLINE

- Synthesis vs. simulation
- Parameterizable IP blocks
- Examples of synthesizable code
- Design rules
- Tooling
- Design flow

SIMULATION VS. SYNTHESIS

- VHDL was designed with only simulation in mind.
- Mature synthesis technology was not yet available in the 80s.
- Synthesis of hardware requires the use of the VHDL *synthesizable language subset* (standardized):
  - Synthesizable data types: `std_logic`, `signed` and `unsigned` of `numeric_std` package.
  - Agreements on how to describe combinational logic.
  - And how to describe sequential logic.
- All hardware descriptions that you have seen until now (siso8) are synthesizable; the testbench code is not!

PARAMETERIZABLE IP BLOCKS

- Reusability of a design (*intellectual property*, IP) is improved by introducing parameters for hardware properties.
- In VHDL such parameters are called generics.
- Check files:
  - **`siso_gen_ent.vhd`**
  - **`siso_gen_copy_arch.vhd`**
- Note also the extra signals for test.
- Generics at the top level require an extra level of hierarchy in the testbench. Check:
  - **`tb_siso_gen.vhd`**

EXAMPLES OF SYNTHESIZABLE CODE

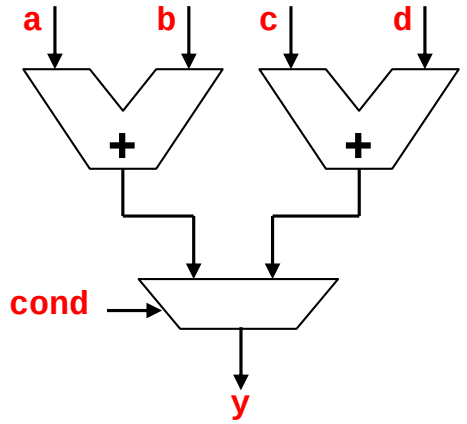
- Truth-table based bit-level logic, handout Figure 23.
- If-then-else based bit-level logic, handout Figure 24.
- Word-level logic, handout Figure 25.
- Memory (to be realized by flipflops), handout Figure 8.

ADDER-MULTIPLEXER EXAMPLE (1)

- Consider code:

```
if cond = '1'
then y <= a+b;
else y <= c+d;
end if;
```

- A 1-to-1 implementation will result in implementation with 2 adders and one multiplexer.

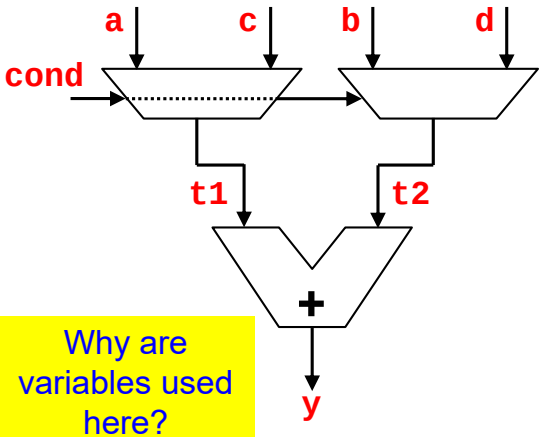


ADDER-MULTIPLEXER EXAMPLE (2)

- Optimized code:

```
if cond = '1'
then t1:=a; t2:=b;
else t1:=c; t2:=d;
end if;
y <= t1 + t2;
```

- Solution with 1 adder and two multiplexers.



Why are variables used here?

ADDER-MULTIPLEXER EXAMPLE (3)

- Guideline:
  - Modern synthesis tools can usually perform the above type of optimization.
  - Optimization may depend on tool-configuration settings.
  - If you want to have better control on the result, you can better write optimal code.

IMPORTANT DESIGN RULES

- Separate sequential and combinational logic by using separate processes.
- Sequential logic should always use an asynchronous reset and the rising edge of the clock.
- Combinational logic descriptions require that signal values are always defined:
  - If a signal receives a value in the “then” branch of an “if” statement, then it should also receive a value in the “else” branch.
  - Otherwise, the synthesis software will generate *latches*, as you are describing memory.

DEALING WITH DELAY (1)

- Obviously, a correct modeling of delays is key for sound digital IC design.
- Delay specification can be done in HDL:
  - Think of keyword `after` in VHDL.
- In practice, delay is specified outside HDL, in a separate file.
- The same circuit has different delay behavior for so-called PVT variations:
  - P: process, e.g. dopants in silicon may not always come out equal, wires may vary in width
  - V: voltage
  - T: temperature.

DEALING WITH DELAY (2)

- Extreme cases are called *corners*, e.g. max. operating temperature and lowest possible voltage is one corner.
- Keeping delays outside HDL allows dealing with different corners without needing to change HDL code.
- A popular delay format is SDF (*standard delay format*).
- The simulator combines HDL and SDF for simulations with accurate delay (delay numbers in SDF overrule default values in HDL).
- An even more important use of SDF is *static timing analysis* (STA, computing longest and shortest paths without simulation). This is not used in the SoC Design course.

TOOLING

- For synthesis we use Synopsys Design Compiler.
- It is used in batch mode and controlled by a configuration script
- Its result is the standard-cell netlist in VHDL, an SDF file with delay information, and a log file.
- Always check log file for errors and warnings.

- In this course, we only deal with SDF for *typical* delays (no extreme corners). In real life, a chip should pass STA for all operating corners before tape-out.

DESIGN FLOW

1. Draw your block diagram.
2. Write the corresponding VHDL code.
3. Simulate with Questasim.
4. If correct, synthesize with Design Compiler.
5. Perform a post-synthesis simulation with Questasim including delay information in SDF.

Go back to Step 1 (sometimes Step 2) if something is not correct at some point.