

Agenda

- Code style
- Intellectual Property (IP)
- Alternatives to VHDL
 - System-C
 - SystemVerilog

2022-09-26

TSTE12 Deadlines Y,D,ED

- Weekly meetings should have started
 - Internal weekly meeting with transcript sent to supervisor
- Project completion
 - Friday 14 October
 - Presentation
 - Project report

TSTE12 Design of Digital Systems, Lecture 10

2022-09-26 4

TSTE12 Deadlines MELE, Erasmus

- Weekly meetings should have started
 - Internal weekly meeting with transcript sent to supervisor
- Project completion
 - Friday 28 October
 - Presentation
 - Project report

2022-09-26 5

2022-09-26

6

TSTE12 Presentation/demonstration

- 15-20 minutes/group
- All group members should participate
- Available times will be announced later
- At least two groups at the same time (2 groups audience)
- Projector, computer and DE2-115 board available
- See web page for guidelines of presentation
 - Want a selling presentation (but do not overdo this)
 - Point out what is different from everyone else designs
- Present both technical and administrative results

TSTE12 Design of Digital Systems, Lecture 10

TSTE12 Project Documents

- Project report
 - Use the general LIPS template document
- Afterstudy report
 - Use the special afterstudy report template
 - Fill in and submit individually
- Delivery
 - Clean out unrelated stuff from the project directories
 - Put a README.TXT at the top level of the project directory
 - Describe where, what name, how to use designs

<page-header>
2020-201
Canadian Set 2 available today 26 September
Deadline 3 October 23:30
Handin set 3
Start 10 October, deadline 17 October 23:30
Handin set 4
Start 24 October, deadline 31 October 23:30
Not necessary if you got atleast 9 correct theory and atleast 9 correct coding tasks
INDIVIDUAL work, no cooperation on handins

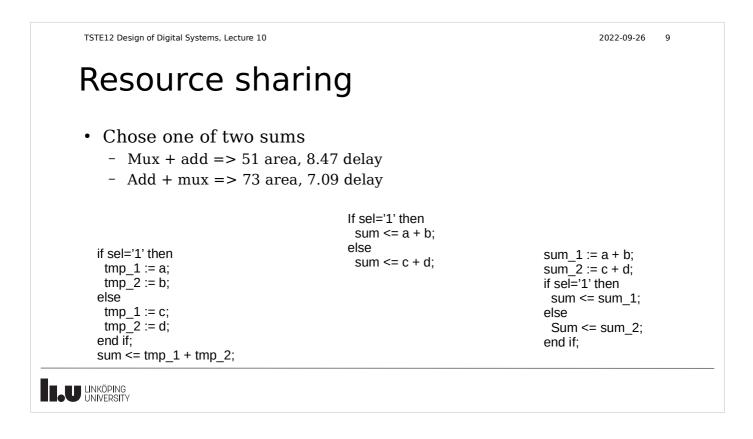
2022-09-26

8

TSTE12 Design of Digital Systems, Lecture 10

Last lecture today for Y/D/ED

- Lecture 11
 - Microprogramming
 - Lab 3
- Lecture 12
 - Low level programming,
 - Assembly language, C
 - Computer Peripheral (I/O)



```
TSTE12 Design of Digital Systems, Lecture 10
```

2022-09-26 10

Resource sharing

- Chose one of two sums. May add both or chose inputs first
 - Mux+add => 51 area, 8.47 delay
 - Add+mux => 73 area, 7.09 delay
- Flattening and structure. (logic level, not hierarchy)
- Logic can be flattened to e.g., two levels instead of three. Different results of area and logic



2022-09-26 11

How is timing requirements defined?

- · Often derived from a symbolic clock
- Signals are defined from edges of the clock
 Fix setup and hold time. Include clock skew
- · Usually defined as maximum delay
 - Expensive to guarantee minimum delay
 - Delay pin to flipflop, flipflop to pin
 - Time from flipflop to flipflop
- · Possible to specify multi cycle delay
- False paths

TSTE12 Design of Digital Systems, Lecture 10

Results

- Time reports
 - Generated by analysis of netlist/layout
 - Critical path reports
- Area reports
- Resource reports
 - Routing, flipflops, LUT, multipliers etc.
- VHDL simulation models
 - Post synthesis, post layout
- Layout possible to modify (edit at bit level)

2022-09-26 13 TSTE12 Design of Digital Systems, Lecture 10 Synthesis operation Synthesis is based on different types of pattern matching - Support most constructs - Behavour may still be different - Often adds complicated patterns that are then simplified • E.g., full flipflop with asynchronous reset and set with fixed inputs. • Example: Generally generates a single flipflop, Process dflipflop(clk) but timing of Qinvers differs between simulation begin if rising_edge(clk) then of VHDL and synthesized design. Q <= D; end if: Qinvers \leq not Q; end process

TSTE12 Design of Digital Systems, Lecture 10

Recommended patterns

- Style guide exists (patterns)
 - Specific to the synthesis tools
- Specify patterns that are allowed and recommended
 - Important to produce efficient implementations
 - Example units: counters, memories, tristate buffers
- These manuals are available online



2022-09-26 15

Common rules/hints for synthesis

- Do not assign initial values to signals and variables in declarations
 - Not supported for synthesis
 - Use explicit reset instead
- Counter design
 - Use loadable down counters if not power of 2 counting
 - Avoids comparison operation, use carry result instead
- Always use limited number ranges
 May get 32 bit arithmetic if not limited
- Allow synthesis tool to select state coding

TSTE12 Design of Digital Systems, Lecture 10

Storage elements

- RAM block synthesis
 - Ordinary array implementation (if not recognized as RAM): flip-flops!
 - Altera Cyclone IV 2C115: 3.9 Mbit RAM
- Large number of available types
 - single/multi port
 - Synchronous/asynchronous
- RAM areas may be initialized (when FPGA is configured)
- ROM areas sometimes implemented as initialized RAM Areas
 - Described as case statements or array of constants

Single port memory	model
• Some hardware require clocking	architecture rtl of sync ram singleport is
library IEEE; use IEEE.std_logic_1164.all; use IEEE.numeric_std.all;	type mem_type is array (2**addr_width-1 downto 0) of std_logic_vector(data_width-1 downto 0); signal mem : mem_type; signal addr_reg : std_logic_vector(addr_width-1 downto
entity sync_ram_singleport is	begin singleport : process(clk)
generic (data_width : natural := 8; addr_width : natural := 8);	begin if rising_edge(clk) then
port(if (we = (1)) then
clk : in std_logic; we : in std_logic;	mem(conv_integer(addr)) <= data_in; end if;
addr : in std_logic_vector(addr_width-1 downto 0);	addr_reg <= addr;
<pre>data_in : in std_logic_vector(data_width-1 downto 0); data_out : out std_logic_vector(data_width-1 downto 0));</pre>	end if; end process singleport;
end entity;	data_out <= mem(conv_integer(addr_reg); end rtl;

2022-09-26 18

If synthesis does not achieve the optimization goals

- Rerun synthesis
 - Optimization often based on probabilistic algorithms
 - Different results in every run
- Try another optimization algorithm in the tool
 - Usually possible to optimize for area or speed
 - Combination of optimizations may give better results
- Change the state coding Select different state coding algorithms

cont.

TSTE12 Design of Digital Systems, Lecture 10 2022-09-26 19 Achieving the optimization goals, • Rewrite the VHDL code - Manually define the state coding - Rewrite with same functionality Known as retiming

- Register balancing
- If vs case
- Rewrite giving different functionality
 - Pipelining
- Rewrite operations manually
 - Supplied adder and multiplier structures may not be optimal in all situations

TSTE12 Design of Digital Systems, Lecture 10

2022-09-26 20 Achieving the optimization goals, cont.

- Increase the minimum power supply voltage
- Reduce the temperature range
- Change technology
- Reduce error coverage
 - Reduce ability to test manufactured chips
- Change to a better synthesis tool
- Optimize logic manually
- Change optimization goals

<page-header>
2022-021 2022
2022-022 2023
2022-024 2023
2022-024 2023
2022-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 2023
2023-024 202

TSTE12 Design of Digital Systems, Lecture 10

2022-09-26 22

How do they work

- May consist of a FA cell or gate up to a complete microprocessor or dedicated systems such as a modem.
 - Examples: microprocessors (ARM, powerpc etc.), memories, peripherals (usb, ethernet, etc.)
- Requires a high-level model that can be used for behavoural simulation (usually not possible to synthesize)
- Behavoural model is replaced with optimized netlist or layout at synthesis (sometimes not available to the user!)



2022-09-26 23

2022-09-26 24

Advantages of IP

- Reuse of code and designs
- Tested (hopefully)
- Fast path to final design
- Do not need to be an expert on every subsystem
 - Example: fast multiplier structures
 - Still get high performance designs

TSTE12 Design of Digital Systems, Lecture 10

Drawbacks with IP

- Interface problems
 - Clock rates, bus protocols, number systems, wordlengths, etc.
- Risks at purchase
 - Functionality
 - Documentation
- Support
- Verification
 - Require lot of testbenches
 - Missing models

2022-09-26 25

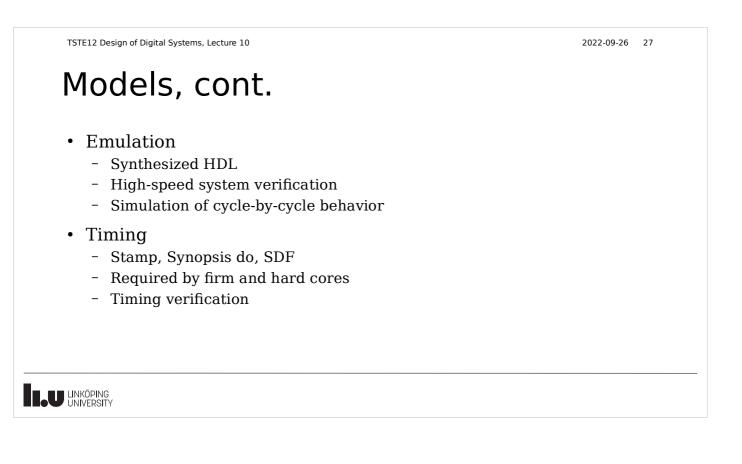
Models at different abstraction levels

- Model type, development environment, need, usage
- ISA
 - C,C++
 - Microprocessor based designs, HW/SW
 - High-speed simulation, application run
- Behavoural
 - C, C++, HDL
 - Non-microprocessor designs
 - High-speed simulation, application run

TSTE12 Design of Digital Systems, Lecture 10

Models, cont.

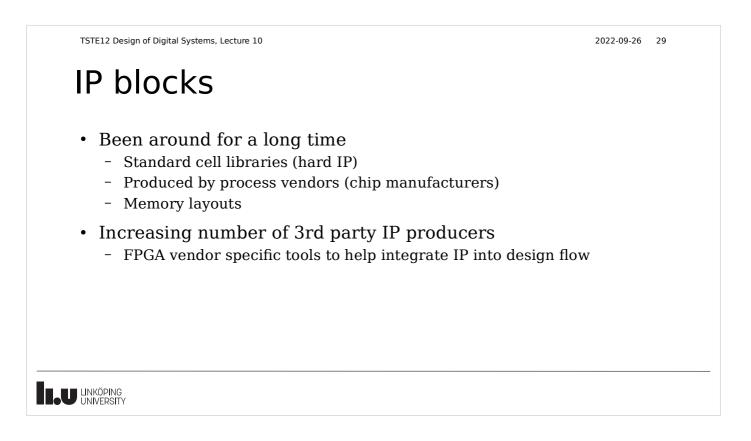
- Bus functional
 - C, C++, HDL
 - System simulation, internal behavior of the core
 - Simulation of bus protocols and transactions
- Fully functional
 - HDL
 - System verification
 - Simulation of cycle-by-cycle behavior



Models, cont.

- Floor plan/area
 - LEF-format
 - Required by hard cores only
 - SoC-level integration and physical design

2022-09-26 28



2022-09-26 30

TSTE12 Design of Digital Systems, Lecture 10

Design flow using IP

- Top-down design flow difficult
 Must get a match between subsystem and IP block
- Best to use a meet-in-the-middle approach
 - Identify functionality to be put in IP
 - Perform top-down partitioning until meet IP



2022-09-26 31

2022-09-26 32

Soft CPU

- Common trend to include soft CPU support
- CPU structure defined as IP
- Peripherals added using configuration files / GUI
- Software drivers automatically included
- GNU based development systems
- Custom instruction support

TSTE12 Design of Digital Systems, Lecture 10

Alternative HDL languages

- Verilog
 - IEEE standard 1995, revised 2001, merged into Systemverilog
- Systemverilog
 - IEEE standard 2005
- System-C
 - IEEE standard 2005
- Handel-C, Catapult C
- Mobius, JHDL (Java HDL)
- Chisel

2022-09-26 33

2022-09-26 34

Verilog language background

- First designed 1985
- Designed for logic simulation (not synthesis)
- Owned by Cadence until 1990
 Released to public 1990
- IEEE standard 1995
- Inspired by C
- Possible to mix simulation of blocks in Verilog and VHDL

TSTE12 Design of Digital Systems, Lecture 10

Verilog, general aspects

- C-like syntax
 - Operators
 - bitfields
- Case sensitive identifiers
- Include files (.h)
 - Corresponds to VHDL packages
 - Share common definitions
- Not strongly typed
- No pointers and access types

TSTE12 Design of Digital Systems, Lecture 10
General structure
 Module Corresponds to VHDL entity Specifies interface and function
module small_block (a, b, c, o1, o2); input a, b, c; output o1, o2;
wire s; assign o1 = s c ; assign s = a && b ; assign o2 = s \hat{c} ; endmodule

TSTE12 Design of Digital Systems, Lecture 10

Data types

- Signal values [0 1 x z]
 - Predefined
- Values specifies bitwidth
 - 334 32 bits wide decimal number
 - 3 bits wide binary number (ie, 011) - 3'b11
 - 20'h'f ffff 20 bits wide hexadecimal number
 - 10'bZ 10 bits wide all tri-state



2022-09-26 35

<page-header><page-header><page-header><section-header><list-item><list-item><list-item><list-item><list-item><list-item><list-item><list-item>

TSTE12 Design of Digital Systems, Lecture 10

2022-09-26 38

Process equivelent construct

- Initial block
 - Only run once
- Always block
 - corresponds to process statement
 - Sensitivity list
 - Sequential and/or parallel code inside





Generic template for clocked circuit with asynchronous and synchronous

40

always @(<edge of clock> or <edge_of_asynchronous_signals>)

if (<asynchronous_signal>)

<asynchronous signal_assignments>

else if (<asynchronous_signal>)

<asynchronous signal_assignments>

... else

<synchronous signal_assignments>

Timing model

- Specify a minimum time resolution
 - Not completely separated from "standard time"
 - Possible to define delay of assignment
 - #1 test = a; // assign test after 1 time unit
- May have a data slip
 - Always @(posedge clk) Q1 = D;
 - Always @(posedge clk) Q2 = Q1; // Data slip
- Data slip solved by added delay
 - Always @(posedge clk) Q1 = #1 D;
 - Always @(posedge clk) Q2 = #1 Q1;

TSTE12 Design of Digital Systems, Lecture 10

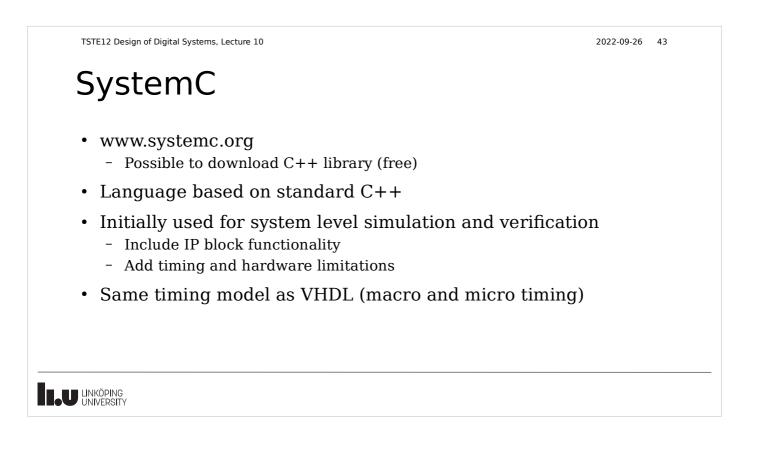
Dynamic processes

- Support named events
 - Declare an event: event event7;
 - Trigger event: -> event7;
 - Code triggered by event:

@ (event7) begin
<Some procedural code>
end

- Support fork/join
 - Difficult to translate to hardware

2022-09-26 42



SystemVerilog

- The following presentation found at
 - http://compas.cs.stonybrook.edu/~nhonarmand/courses/sp15/cse502/ res/date04_systemverilog.pdf

2022-09-26 44

- Presented at Design Automation Conference (DATE) in 2004



<section-header><table-cell><section-header><list-item><list-item><list-item><list-item><list-item><list-item>

