Kent Palmkvist



TSTE12 Design of Digital Systems, Lecture 4

Agenda

- Practical issues
- Short tool overview
- Introduction to VHDL, continued
 - Timing
 - Testbench

2024-09-09

TSTE12 Design of Digital Systems, Lecture 4

TSTE12 Deadlines Y,D,ED

- First meeting with supervisor should happen no later than today!
 - Determine project manager (contact person)
 - Questions (short meeting)
- Lab 1 deadline Wednesday 11 September at 21.00
 - Require pass to continue project!
- Tuesday 10 September: First version of requirement specification
 - We use LIPS "light", want to capture expected behavior of final result in requirement specification

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 4

TSTE12 Deadlines MELE, erasmus

- Group definition Wednesday 11 September (afternoon)
 - On web, include supervisor assignment
- Friday 13 September: First meeting with supervisor
 - Determine project manager (contact person)
 - Question (short meeting)
- Tuesday 17 September: First version of requirement specification



2024-09-09

6

TSTE12 Design of Digital Systems, Lecture 4

MUX lab access

- LiU-card should now give access to MUX2 lab
 - Email me if you can not get into the lab
- Lab available 5-23 every day
 - Make sure to verify in schedule server if lab is available outside course schedule
 - MUX2 mostly used only for TSTE12
 - MUX1 also sometimes available (used more by other courses)
- Remote login: use thinlinc

TSTE12 Design of Digital Systems, Lecture 4

Project issues

- Expected project participation conduct
 - Do no be late to meetings
 - Inform the rest of the group if you have problem attending a meeting (in advance if possible)
 - Keep track of your project work, noting amount and type of task
- Documents should be discussed and approved by supervisor
- Possible to fail project even if design works
- Possible for individual to fail project even if rest of group get a pass!

2024-09-09

8

TSTE12 Design of Digital Systems, Lecture 4

Project hints

- Hints about Requirement specification
 - Possible subsystem: control, display, audio processing
 - Add plenty of features
 - Set priority (low, medium, high)
 - Avoid multiple requirements in one requirement statement
- Hints about design specification
 - Should indicate idea about general building blocks
 - Interfaces (signals/data to communicate)
 - Behavior

TSTE12 Design of Digital Systems, Lecture 4

Design flow and tools

- Three types of examined activities in the course
 - Handin
 - Lab
 - Project
- For handins (start next week): use simple text editor + modelsim
 - Start the TSTE12handin shell
 - Write code, compile, simulate, finally upload code
- Chapter 2 tutorial notes shows how to use modelsim http://www.isy.liu.se/edu/kurs/TSTE12/kursmaterial/

TSTE12 Design of Digital Systems, Lecture 4

HDL Designer tool

- Design entry tool, main entry tool to the project
 - Tutorial chapter 3 introduce this www.isy.liu.se/edu/kurs/TSTE12/kursmaterial
- Tools used to manage libraries, design, and other tools for use by larger designer groups
 - Graphic and text design entry
 - Tool startup configurations
 - Support many different languages and tools
 - Version control, team management....
- Highly configurable

TSTE12 Design of Digital Systems, Lecture 4

HDL Designer tool, cont.

- Top level: The project
 - Defined by xxx.hdp file
 - Contains list of libraries, (1 or more)
- Each library contains design units
 - Described as components (green and blue boxes)
 - Each unit have different view
 - Graphic and/or textual
 - Various forms of architectures (text, block, FSM, ...)
 - A default architecture view is indicated by a blue arrow
- Interfaces with simulation and synthesis tools



2024-09-09

HDL Designer tool, cont.

- Green boxes (components)
 - Fixed interface (does not automatically update)
 - Possible to reuse in multiple designs
- Blue boxes (subsystems)
 - Updates interface when adding/removing inputs/outputs in block diagram (remember to save schematic to update VHDL)
- Tools can generate valid VHDL from graphical representation (schematics, state machines, etc.)
- State machine example in lab3 lab material



2024-09-09

11



2024-09-09

12

TSTE12 Design of Digital Systems, Lecture 4

File I/O

- Possible to read or write a file (1993 allow both on same file)
- Formatted IO
 - Not generally human readable (platform dependent)
- TEXT IO
 - Human readable
- Special package includes definitions
 - STD.TEXTIO
 - Functions for open file, read a complete line, and read individual data from the line

TSTE12 Design of Digital Systems, Lecture 4

Later revisions

- Mostly simplifications and additional function support
- 1993:
 - 8-bit ASCII, identifier restrictions relaxed, declarations simplifications
 - Shared variables (global variables outside processes).
 - Improved reporting in assert statements
- 2008:
 - Simplified sensitivity lists (keyword all to include all signals used)
 - Simplified conditions, allow bit and std_logic values as result of condition
 - Read of output ports on entity
- Tools does not always support latest revision!

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 14

VHDL timing and concurrency

- Simulation of concurrent events (hardware) on a sequential computer
- Must have the same result from simulation independent of execution order of individual event
- Delay is an important property of hardware that must be simulated

2024-09-09 16

TSTE12 Design of Digital Systems, Lecture 4

Signals vs Variables

- Electronic signals can not change values in 0 seconds
 Always slopes on voltages going from 0 to 1
- Common sequential code assumes variables are updated before next statement is executed
- Expect different result depending on if variables or signals are used
- Both variables and signals can be used in synthesized code

TSTE12 Design of Digital Systems, Lecture 4

Signal vs Variable example

• Inputs with changing value at different times

X: 1	4	5	5	3
Y: 2	2	2	3	2
Z: 0	3	2	2	2
nitial	t1	t1+2	t1+4	t1+6

• Result depends on if signals or variables as assigned

BS ·	$<= A^{-1}$	Z after	11s, 2 ns;			BV := AV +	Ż;		. –	-
AS:	2	2	8	10	15	AV: 2	8	10	15	6
BS:	2	2	5	10	12	BV: 2	11	12	17	8
	initial	t1	t1+2	t1+4	t1+6	initial	t1	t1+2	t1+4	t1+6

2024-09-09 17

2024-09-09 18

Signal assignment with delta delay

- Minimum delay is a delta delay
- Delta delay is > 0 s but much smaller than the minimum timestep of the simulator

X: 1	L	4	4		4	
Y: 2	2	2	2		2	
Z: ()	3	3		3	
initia	1	t1	t1	+delta	t1+2	2*delta
AS ·	<= X*	Υ;				
BS <	<= AS	5+Z;				
AS:	2	2	<u>)</u>	8		8
BS:	2	2	2	5		11

TSTE12 Design of Digital Systems, Lecture 4

Delta delay

- Can not be explicitly specified
- Delta delays will never add up to a simulation delay in seconds (standard time)
- Sometimes referred to as Macro (simulation time) and micro (delta delays) timing.
- Time may stand still in simulation by continuous signal updates
 - Example: process triggered by a signal that it is updating
 - Combinatorial loops without macro delay in assignments
 - Delta delay is increasing but not the simulation time

TSTE12 Design of Digital Systems, Lecture 4

Simulation models

- Delta delay only
 - Functional verification of models
- Standard time unit delay only
 - Validate system timing
- Mixed
 - Delta delay where delay is not important
 - Standard time unit delay where delay is significant
 - Study system timing

TSTE12 Design of Digital Systems, Lecture 4

VHDL timing

- Two types of time in VHDL
 - Variables: no delay in update
 - Signals: standard time delay and/or delta delay
- Delta delay
 - Never adds up to a standard time unit
 - Default delay when assigning signals unless delay is specified
- Known as macro and micro timing

2024-09-09 20

Timing implementation in simulation

- Simulator program flow
 - 1) If no entries in queue then stop, else increase time to next time entry in $% \left({{\left[{{{\rm{queue}}} \right]}_{\rm{c}}}} \right)$
 - 2) Start a new simulation cycle without advancing simulation time. Remove all entries scheduled for current simulation time, update all signals. Activate triggered processes
 - 3) Execute activated processes. Schedule new time queue entries.
 - 4) If there are new transactions on signals due to assignment with delta delay, then goto 2, otherwise goto 1
- Concurrent assignment can be seen as processes

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 22

Unexpected simulation results

- Time may stand still in simulation by continuous signal updates
 - Example: process triggered by a signal that it is updating
 - Combinatorial loops without macro delay in assignments



2024-09-09 24

TSTE12 Design of Digital Systems, Lecture 4

Simulation models

- Delta delay only
 - Functional verification of models
- Standard time unit delay only
 - Validate system timing
- Mixed
 - Delta delay where delay is not important
 - Standard time unit delay where delay is significant
 - Study system timing

TSTE12 Design of Digital Systems, Lecture 4

Example of models

architecture TWO of BUF is Entity BUFF is port (X: in BIT; Z out BIT); signal Y: BIT; • A simple buffer examples end; begin process(X) - All buffer have Architecture ONE of BUFF is begin Y <= X; signal Y: BIT; different propagation end process begin delay Z <= Y: process(X) end TWO; variable Y : BIT; - Difference in delta begin architecture THREE of BUF is delays are difficult to Ž <= X; signal Y1, Y2: BIT; end process; see in waveform begin end ONE; windows Y1 <= X; Y3 <= Y2; - Possible to create Y2 <= Y1; multiple delta delay Z <= Y3; end THREE;

2024-09-09 25 TSTE12 Design of Digital Systems, Lecture 4 Example models, cont. Architecture FIVE of BUFF is signal Y5: BIT; begin process(X) Two almost identical buffers begin Y5 <= X; - Have very different simulation behavour Z <= Y5; - Both probably generate same hardware in end process; end FIVE; synthesis architecture FIVE A of BUF is Lacking entries in sensitivity list signal Y5: BIT; begin - Solution: Always add all input signals to the process(X,Y5) sensitivity list begin Y5 <= X; Drawback: unnecessary process triggering may give Z <= Y5; slower simulation end process end FIVE A;

TSTE12 Design of Digital Systems, Lecture 4

Inertial and Transport delay

- Delay can be of two types (3 in VHDL93)
 - Inertial Z <= I after 10 ns;
 - If input change again before end of delay then do not update output

2024-09-09 26

- Filter out short glitches (RC delay)
- Transport Z <= transport I after 10 ns;
 - "True" delay of signal (like transmission lines)
- Reject (VHDL93) Q <= reject 4 ns inertial a after 10 ns;
 - Q_tmp <= A after 4 ns; Q <= Q_tmp after 6 ns;

TSTE12 Design of Digital Systems, Lecture 4

Implementation of Inertial and Transport delay in simulator

- Important to understand why a signal change may not reach the assigned signal
- Transaction
 - Pair of value and time. What value when
- Waveform
 - A series of transactions (sorted by time value)
- Current value of driver
- Value of transaction whose time is not greater than current simulation time. Removed when simulation time is updated if next transaction time is reached

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 28

Waveform update algorithm

1. All old transactions with time at or after earliest new transaction are deleted. Add new transactions to the waveform

If inertial then

- 2. Mark all new transactions
- 3. Mark old transaction if it immediately precedes a marked transition and its value is the same as the marked transaction
- 4. Mark the current value transaction

5. All unmarked transactions are removed



2024-09-09 29

Waveform update example

- Z <= I after 10 ns; (I is a 5 ns pulse starting at t=0)
- First change Z updated to '1' at t=0, (10,'1') transaction added
 Both current and transaction marked and kept
- Second change, Z updated to '0' at t=5, (15,'0') transaction added
- If inertial: (10,'1') not marked, removed
- End result: the pulse on I is not visible on Z (filtered out)

TSTE12 Design of Digital Systems, Lecture 4

Inertial delay side effects

- Process for generating reset signal Res
 - Only executed once at start
 - First assignment is eliminated by second assignment
- Use transport or combined assignment to get pulse

Res <= transport '1' after 50 ns;

Res <= transport '0' after 100 ns;

• Generate complete waveform instead

Res <= '1' after 50 ns, '0' after 100 ns;

Process begin Res <= '1' after 50 ns; Res <= '0' after 100 ns; wait; end process;

2024-09-09 30

TSTE12 Design of Digital Systems, Lecture 4

Modeling of combinational and sequential logic

- Simple approach.
 - Process sensitivity list = circuit inputs
 - Compute new value using variables
 - Assign output signal with delay
 - Possible to synthesize (ignoring delay)
- Models uses generic in the port
 - Adds parameters to components without need of a signal
 - May have default values in entity declaration

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 32

Combinational logic examples

 Gates 	,
---------------------------	---

- Generic states delay
- May have default delay defined

```
entity NAND2 is
 generic(DEL: TIME);
 port(I1,I2: in BIT; 0: out BIT);
end NAND2;
architecture DF of NAND2 is
begin
 0 <= I1 nand I2 after DEL;
end DF;
```

```
component XOR_GATE
port (X,Y : in bit; 0 : out bit);
end component;
component NAND_GATE
Generic (DEL: TIME := 3 ns);
port (X,Y : in bit; 0 : out bit);
end component
```

architecture STRUCTURAL of ONES_COUNT is

signal I1, I2, I3 : bit;

begin

```
end STRUCTURAL;
```

2024-09-09 33

Combinational logic examples, cont.

- Two-to-4 decoder
 - Set one of the four outputs to '1' based on the I input value

entity TW0_T0_4_DEC is generic(DEL: TIME); port(I: in BIT_VECTOR(1 downto 0); 0: out BIT_VECTOR(3 downto 0)); end TW0_T0_4_DEC; architecture ALG of TW0_T0_4_DEC is begin process(I) begin case I is when "00" => 0<= "0001" after DEL; when "01" => 0<= "0010" after DEL; when "10" => 0<= "0100" after DEL; when "11" => 0<= "1000" after DEL; end case; end process; end ALG;

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 34

Sequential logic process template

process(clk, ...) begin Must check both event and if <async expressions> then level to detect clock edge async behavior elsif clk'event and clk='1' then sync behavior - Alternative functions endif available in the std_l,ogic end process; libraries rising_edge, falling_edge process(c1k) • Do NOT do the following: process(clk) begin begin if clk='1' then if clk='1' then • This is acting as a flip-flop Q <= D; sync behavior endif endif based design, but is synthesized end process; end process; to a latch based one! LINKÖPING UNIVERSITY

TSTE12 Design of Digital Systems, Lecture 4

Sequential logic, cont.

- Latch
 - Latches missing the edge detection
 - Bad design style
 - Synthesis result not working
- Flipflop would only copy D when a positive edge on Clk



```
entity LATCH is
generic(LATCH_DEL:TIME);
port(D: in BIT_VECTOR(7 downto 0);
        CLK: in BIT;
        LOUT: out BIT_VECTOR(7 downto 0));
end LATCH;
architecture DFLOW of LATCH is
begin
    LATCH: process(clk,D)
    begin
    If (clk='1') then
        LOUT <= D after LATCH_DEL;
    end if;
end process;
end DFLOW;
```

TSTE12 Design of Digital Systems, Lecture 4 Sequential logic	2024-09-09 36 architecture ALG of JKFF is begin process(CLK,S,R)
 JK flipflop with asynchronous set/reset entity JKFF is generic(SRDEL,CLKDEL: TIME) port(S,R,J,K,CLK: in BIT; Q,QN: inout BIT); end JKFF; 	begin if S = '1' and R = '0' then Q <= '1' after SRDEL; QN <= '0' after SRDEL; elsif S = '0' and R = '1' then 0 <= '0' after SRDEL:
 Edge trigged using 'event 	QN <= '1' after SRDEL; elsif CLK'EVENT and CLK = '1' and
 Asynchronous update 	S='0' and $R='0'$ then if $I = '1'$ and $K = '0'$ then
Higher priority than clocked circuit function	Q <= '1' after CLKDEL; QN <= '0' after CLKDEL; elsif J = '0' and K ='1' then Q <= '0' after CLKDEL;
 Synchronous update 	QN <= '1' after CLKDEL; elsif J= '1' and K= '1' then
 Note use of elsif (must be used) 	Q <= not Q after CLKDEL; QN <= not QN after CLKDEL; end if;
 Edge trigged using 'event 	end process; end ALG;

2024-09-09 37

2024-09-09

38

Sequential logic, cont.

- Register with alternative design
 - Use a guarded statement
 - Use 'STABLE instead of 'EVENT

```
entity REG is
generic(DEL: TIME);
port(RESET,LOAD,CLK: in BIT;
DATA_IN: in BIT_VECTOR(3 downto 0);
Q: inout BIT_VECTOR(3 downto 0));
end REG;
architecture DF of REG is
begin
REG: block(not CLK'STABLE and CLK ='1')
begin
Q <= guarded "0000" after DEL when RESET ='1' else
DATA_IN after DEL when LOAD ='1' else
Q;
end block REG;
end DF;
```

TSTE12 Design of Digital Systems, Lecture 4

Output feedback problems

- Entity output can NOT be read in the architecture
- Three solutions
 - Use INOUT
 - Does not match OUT, enables output values to influence internal signal values
 - Use BUFFER
 - Does not match OUT, complicates building testbenches etc.
 - Use OUT with a temporary signal
 - use temporary signal everywhere needed (read and assign), assign entity out signal at the end of the architecture
- LINKÖPING UNIVERSITY

2024-09-09 39

Sequential logic, oscillator architecture ALG of CLOCK_GENERATOR is signal CLOCK: BIT; begin process (RUN, CLOCK) • Run signal indicate when variable CLKE: BIT := '0'; to start generating clock entity CLOCK_GENERATOR beain if RUN='1' and not RUN'STABLE then CLKE := '1'; generic(PER: TIME); pulses. port(RUN: in BIT; CLK: out BIT); CLOCK <= transport '0' after PER/2; end CLOCK_GENERATOR; CLOCK <= transport '1' after PER; Feedback example end if; if RUN='0' and not RUN'STABLE then Need extra variable CLKE := '0'; end if; to guarantee complete if CLOCK='1' and not CLOCK'STABLE and CLKE = '1'then CLOCK <= transport '0' after PER/2; clock cycles CLOCK <= transport '1' after PER; - Simulation use only, end if; will not synthesize CLK <= CLOCK; end process; end ALG;

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 40

Sequential logic, oscillator

end loop; end process; end ALG;	 Wait statement based Can not have both wait and sensitivity list in process 	<pre>entity COSC is generic(HI_TIME,LO_TIME: TIME); port(RUN: in BIT; CLOCK: out BIT := '0'); end COSC; architecture ALG of COSC is begin process begin wait until RUN ='1'; while RUN = '1' loop CLOCK <= '1'; wait for HI_TIME; CLOCK <= '0'; wait for LO TIME:</pre>
end Loop; end process; end ALG;		CLOCK <= '0'; wait for LO_TIME;
end process; end ALG;		end Loop;
end ALG;		end process;
		end ALG;

TSTE12 Design of Digital Systems, Lecture 4

Numeric calculations

- Bit-vectors (and std_logic_vectors) does not correspond to a numeric value
 - "1011" could mean 11 in decimal (unsigned), or -5 in decimal (2's complement), or even -3 if sign-magnitude would be used
- Additional definitions are included in supporting packages to enable arithmetic on bit-vectors and std_logic_vectors
 - ieee.numeric_bit.all
 - ieee.numeric_std.all
- Must use defined types signed or unsigned to allow calculations
 - Same definitions as bit_vector and std_logic_vector
 - Can copy values between types due to same element type



TSTE12 Design of Digital Systems, Lecture 4

Numeric calculations example

- Counter incrementing 3-bit count value each clock cycle
 - Asynchronous reset example
 - Clock edge detection using 'event

library ieee; use ieee.numeric_bit.all;

entity INL3_KB is port (C : in bit; R : in bit; Q : out bit_vector(1 to 3)); end entity; architecture KB of INL3_KB is begin

process(C,R)
variable count : unsigned(1 to 3);
begin
if R = '1' then
count := (others => '0');
elsif C'event and (C='1') then
count := count + 1;
end if;
Q <= bit_vector(count);
end process;</pre>

end architecture;

Numeric calculations de	elalis	
Addition does not increment wordlengths	101 +011	0101 +0011
 May get overflow Must signextend to detect carry 	000	1000
 Adding different length vectors will sign extend the May still get overflow 	e shortest one	
• Multiplication always generates an output number of input bits	of bits equal to the	e total number
 Multiplying a 3-bit input with a 4-bit input gene 	erates a 7-bit outpu	ut result

2024-09-09 45

2024-09-09 46

Avoid old packages

- Before the introduction of numeric_std and numeric_bit there where other libraries
 - std_logic_unsigned, std_logic_signed
 - std_logic_arith
- Do NOT use these, they are obsolete
 - Made it difficult/impossible to mix signed and unsigned

TSTE12 Design of Digital Systems, Lecture 4

Including integers

- Integers can be used for synthesis
 - If synthesis tool cannot figure out the limits, the result is 32-bit arithmetic
 - Subtypes (limiting range) help to reduce hardware and catch unexpected use
- Integers will be implemented as bitvectors
 - Either unsigned or signed (2's complement)
 - Translation between integer and bitvectors exist
 - x_signed := to_signed(y_int,x_signed'size);
 - Translation other way around (unsigned to integer value)
 - y_int = to_integer(x_signed);

Another aspect of signal assignment

- One signal can be assigned from different parts of the code
 - Support multiple entities driving the same wire
 - Example: Databus in a computer connecting multiple memories and CPU
- Modelling must be strict and clear
 - Same result independant of simulator tool
 - Should not be able to detect the order the processes where calculated
- Not all data types support multiple sources for the value

TSTE12 Design of Digital Systems, Lecture 4

2024-09-09 48

Multiple assignment on one signal

- Each process containing a signal assignment will have a driver in the simulator generating a contribution to the final signal value
 - Concurrent signal assignments will have one driver each
 - Processes only have one driver for each signal (even with multiple assignment)
 - The signal update seen before is done individually on each driver
 - One driver does not know anything about other drivers
- When the value of a signal is fetched, the contributions from the different drivers current values are collected.
 - The resulting signal value depends on the definition of how to combine the values from the different drivers, using a resolution function

TSTE12 Design of Digital Systems, Lecture 4

Example of data types supporting multiple drivers

- Signals driven by multiple drivers must be resolved
 - Use a special function that resolves multiple drivers
- Resolution function
 - Example: Wired-OR
 - signal X1 : WIRED_OR Bit;
 - subtype STD_LOGIC is RESOLVED STD_ULOGIC;
 - signal Y2 : STD_LOGIC;
 - RESOLVED is the resolution function name
 - Called every time the value of the signal is calculated
 - Gets all driver values as input

50 Multivalued logic • Not enough with 0 and 1 to model "real" logic • Example: Bus • Requires bus release • Signal assignment driver can not drop its value • Use a value to indicate not driven, and indicate nondriven signals (Z) • Need to indicate conflicting driver (X)

Multivalued logic in VHDL

- Alternative to data type BIT
 - Type MVL4 is ('X', '0', '1', 'Z');
 Type MVL4_VECTOR is array(NATURAL
 range <>) of MVL4;
- X leftmost to make it the initial value unless explicitly initialized in the code

Multidriver signals

- Requires a resolution signal
- Different combinations possible
 - X always overrides others
 - 0 and 1 at the same time gives X
 - Z and Z gives Z

Resolution function definition

Subtype DotX is wiredX MVL4;

- WiredX is the name of the resolution function Function WiredX (V:MVL4_VECTOR) return MVL4;
- Where V is a vector containing all values of all drivers of a signal

Resolution function implementation

```
• Implement as a loop and lookup table
```

```
Function wiredX (V: MVL4_VECTOR) return MVL4 is
    Variable result: MVL4:= 'Z';
Begin
    For i in V'RANGE loop -- range not known in advance
        Result = table_WIREDX(result,V(i));
        Exit when result = 'X';
    End loop;
    Return result;
End wiredX;
```

Resolution function impl., cont.

- Check of X in loop is not necessary, but speed up simulation
- Table should then look like:

```
Type MVL4_TABLE is array (MVL4, MVL4) of MVL4;

Constant table_WIREDX : MVL4_TABLE :=

--

--

X 0 1 Z

--

(('X', 'X', 'X', 'X', 'X'), -- X

('X', '0', 'X', '0'), -- 0

('X', 'X', '1', '1'), -- 1

('X', '0', '1', 'Z')); -- Z
```



