TTÜ 1918

Digital Systems Modeling and Synthesis
Digitaalsüsteemide modelleerimine ja süntees
IAY0340 – 5.0 EAP 4 2-2-0 E S

http://www.ati.ttu.ee/IAY0340/

Peeter Ellervee

ICT-526   620 2258   511 3631   LRV@ati.ttu.ee

http://www.ati.ttu.ee/~lrv/   http://mini.li.ttu.ee/~lrv/

TTÜ 1918

Course plan

- **Introduction and simulation principles   [1 h]**
- **Hardware description language VHDL   [7 h]**
- **VHDL and RTL synthesis   [2 h]**
- **Testbenches, co-simulation and co-modeling of digital systems   [4 h]**
- **Hardware description language SystemVerilog   [4 h]**
- **Synthesis of digital systems at various abstraction levels   [4 h]**
- **Hardware description language SystemC   [2 h]**
- **Simulation of analog and digital systems – Spice, VHDL-AMS, etc.   [2 h]**
- **Code transformations and system level description languages   [2 h]**

- **Hands-on exercises   [15*2 h]**
  - **Digital simulation and synthesis – various modules in VHDL [14 h]**
  - **Elevator model – VHDL & Verilog [4 h]**
  - **PicoCPU – simple processor [6 h]**
  - **FIR filter design, SystemC and analog simulation [6 h]**

Textbooks

- **Dirk Jansen et al. (editors), "The electronic design automation handbook." [2003]**
- **Peter J. Ashenden, "The Designer's Guide to VHDL." [2008]**
- **Volnei A. Pedroni, "Circuit Design with VHDL." [2004]**
- **Stuart Sutherland, Simon Davidman, Peter Flake, "System Verilog for Design." [2010]**
- **Michael John Sebastian Smith, "Application-Specific Integrated Circuits." [1997]**
  - **http://www10.edacafe.com/book/ASIC/ASICs.php**
- **David C. Black, Jack Donovan, "SystemC: From the Ground Up." [2004]**

- **Plus some good older ones**
  - **Kalle Tammemäe, "Riistvara kirjeldamiskeel VHDL." [2002]**
  - **K.C. Chang, "Digital systems design with VHDL and synthesis: an integrated approach." [1999]**
  - **Ken Coffman, "Real world FPGA design with Verilog." [2000]**

- **Plus web-based source, incl. WikipediA**
  - **Bryan Mealy, Fabrizio Tappero, "Free Range VHDL." [2012] – http://freerangefactory.org**

Motivation

- **System-on-Chip (SoC) requires new design methodologies**
  - **to increase designers productivity**
  - **to get products faster into market**

- **There exists a demand for efficient design methodologies at higher abstraction levels**

- **A different thinking needed from the designers**

- **At higher abstraction levels**
  - **a designer has much wider selection of possible decisions**
  - **each of these decisions has also a stronger impact onto the quality of the final design**

TTÜ1918

Optimizations

- **Optimizations at logic level**
  - **thousands of nodes (gates) can exist**
  - **only few possible ways exist how to map an abstract gate onto physical gate from target library**
  - **optimization algorithms can take into account only few of the neighbors**

- **Optimizations at register transfer level (RTL)**
  - **hundreds of nodes exist (adders, registers, etc.)**
  - **there are tens of possibilities how to implement a single module**

- **At higher levels, e.g. at system level**
  - **there are only tens of nodes to handle (to optimize)**
  - **there may exist <u>hundreds</u> of ways how to implement a single node**
  - **every possible decision affects much stronger the constraints put onto neighboring nodes thus significantly affecting the quality of the whole design**

TTÜ1918

Decisions at higher abstraction levels

- **Two major groups of decisions**
  - **selection of the right algorithm to solve a subtask**
    - **making transformations inside the algorithm, e.g. parallel versus sequential execution**
    - **affect primarily the final architecture of the chip**
  - **decisions about the data representation**
    - **e.g. floating point versus fixed point arithmetic, bit-width, precision.**

- **Selection of a certain algorithm puts additional constraints also onto the data representation**

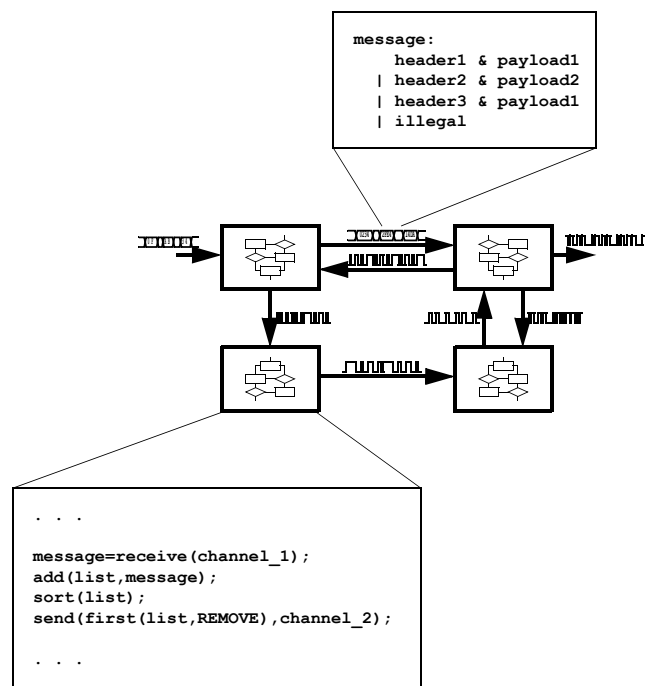- **Selecting a data representation narrows also the number of algorithms available**

TTÜ 1918

Design steps

- **System design**
  - *a.k.a.* **Architectural-level synthesis**
  - *a.k.a.* **High-level synthesis**
  - *a.k.a.* **Structural synthesis**

  - **description / specification –> block diagram**

  - **determining the macroscopic structure, *i.e.*, interconnection of the main modules (blocks) and their functionality**

- **Logic design**

  - **block diagram –> logic gates**

  - **determining the microscopic structure, *i.e.*, interconnection of logic gates**

- **Physical design**
  a.k.a. **Geometrical-level synthesis**

  - **logic gates –> transistors, wires**

© Peeter Ellervee

TTÜ 1918

Abstraction levels

```
message:
    header1 & payload1
    | header2 & payload2
    | header3 & payload1
    | illegal
```
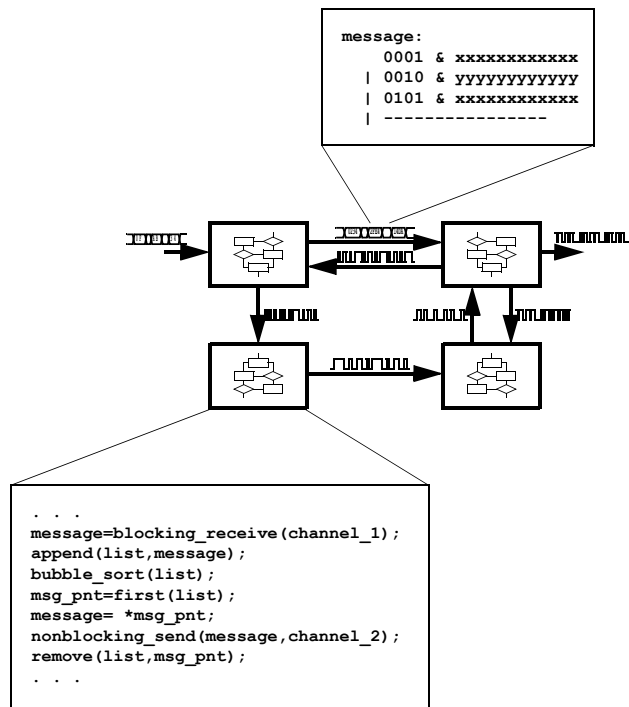
- **System level**
  - **modules / methods**
  - **channels / protocols**

- **System Level Synthesis**
  - **Clustering. Communication synthesis.**

```
. . .

message=receive(channel_1);
add(list,message);
sort(list);
send(first(list,REMOVE),channel_2);

. . .
```

© Peeter Ellervee

TTÜ1918

Abstraction levels

```
message:
    0001 & xxxxxxxxxxxx
  | 0010 & yyyyyyyyyyyy
  | 0101 & xxxxxxxxxxxx
  | ----------------
```

- **Algorithmic level**
  - **(sub)modules / algorithms**
  - **buses / protocols**

- **High-Level Synthesis**
  - **Resource or time constrained scheduling.**
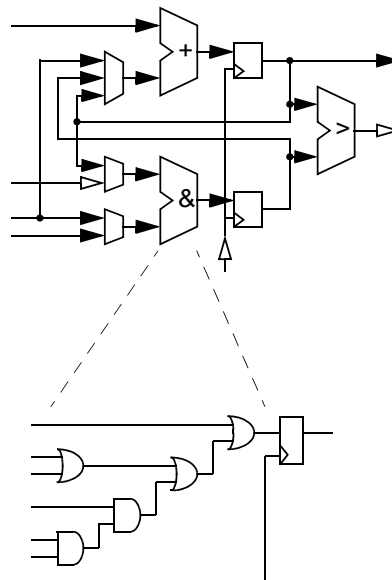 - **Resource allocation. Binding.**

```
. . .
message=blocking_receive(channel_1);
append(list,message);
bubble_sort(list);
msg_pnt=first(list);
message= *msg_pnt;
nonblocking_send(message,channel_2);
remove(list,msg_pnt);
. . .
```

TTÜ1918

Abstraction levels

- **Register transfer (RT) level**
  - **blocks / logic expressions**
  - **buses / words**
- **Level Synthesis**
  - **Data-path synthesis. Controller synthesis.**

- **Logic level**
  - **logic gates / logic expressions**
  - **nets / bits**
- **Logic Level Synthesis**
  - **Logic minimization. Optimization, overhead removal.**
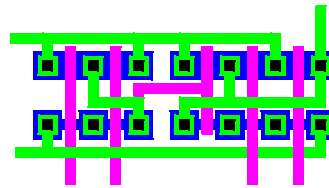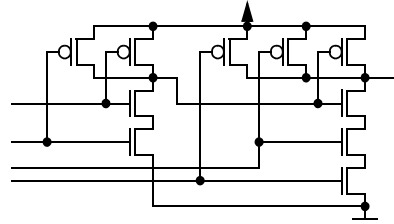
TTÜ1918

Abstraction levels

- **Physical level**
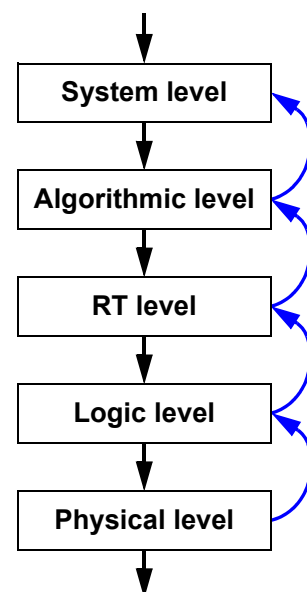
  - **transistors / wires**

  - **polygons**

- **Physical Level Synthesis**
  - **Library mapping.
    Placement. Routing.**

TTÜ1918
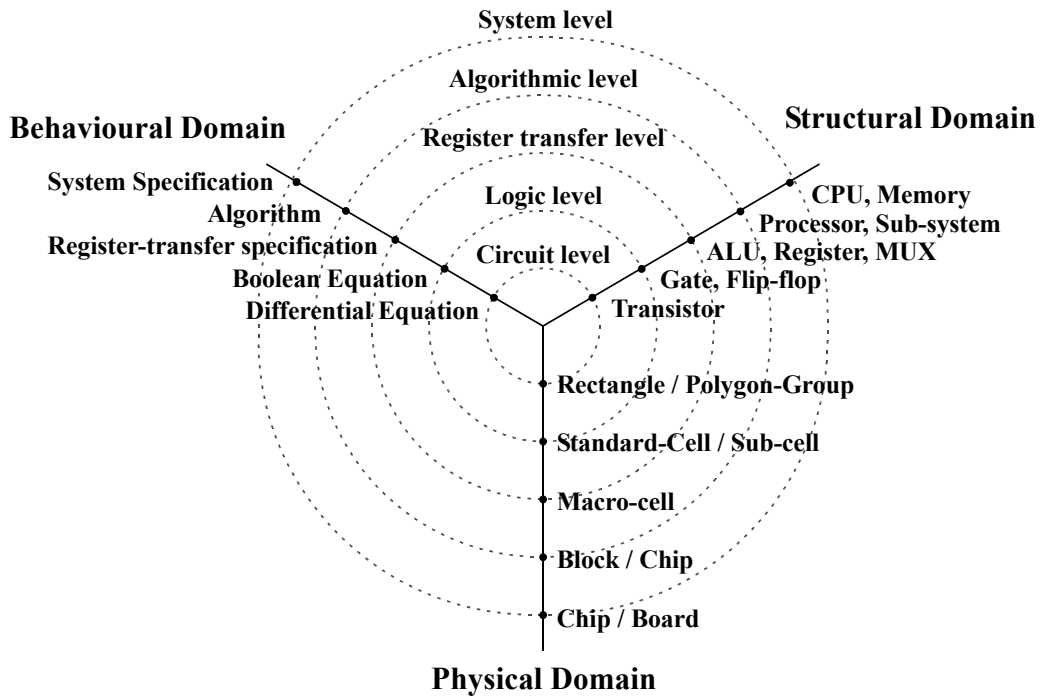
Design flow

HW design flow

- **Specification refinement**
  - **from higher to lower abstraction levels**
  - **refinement = transformations**
- **Algorithm selection**
  - **universal vs. specific**
  - **speed vs. memory consumption**
- **Partitioning**
  - **introducing structure**
  - **implementation environment -- HW vs. SW**
- **Technology mapping**
  - **converting algorithm into Boolean equations**
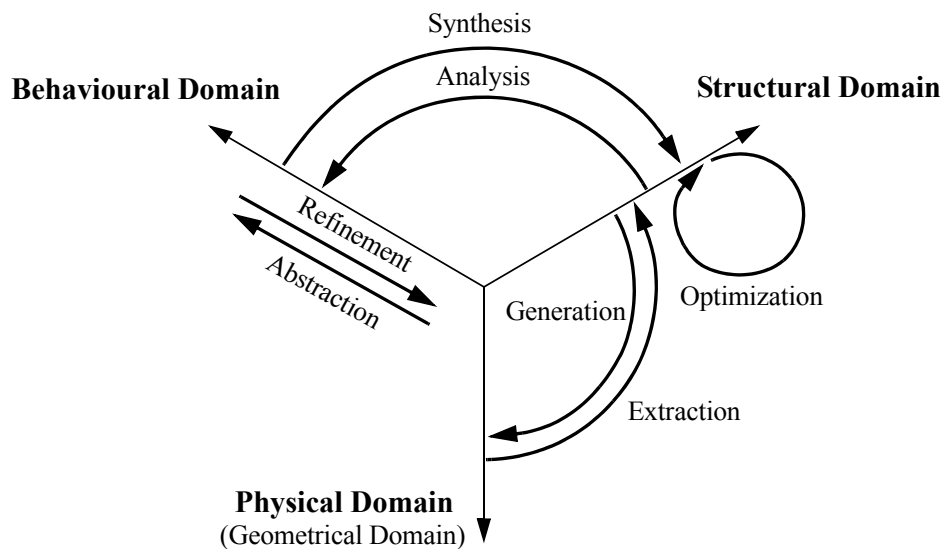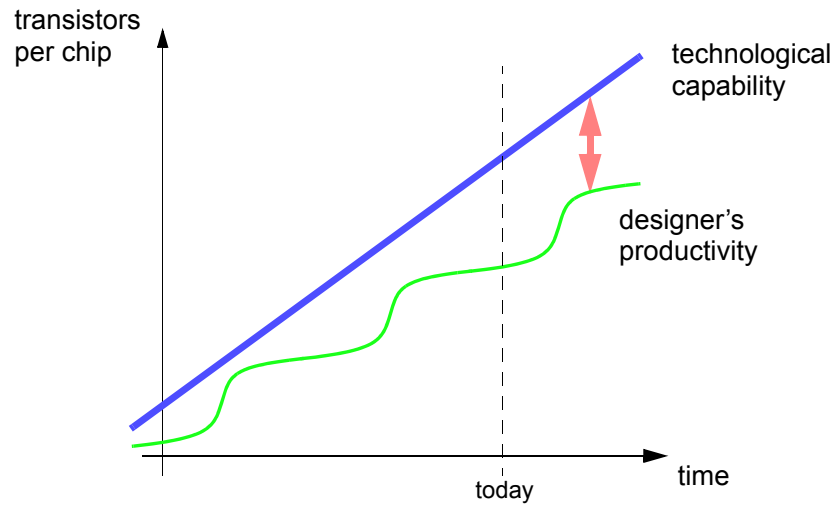 - **replacing Boolean equations with gates**

System level

Algorithmic level

RT level

Logic level

Physical level

Y-chart

System level

Algorithmic level

Behavioural Domain

Register transfer level

Structural Domain

System Specification

Logic level

Algorithm

CPU, Memory

Register-transfer specification

Circuit level

Processor, Sub-system

ALU, Register, MUX

Boolean Equation

Gate, Flip-flop

Differential Equation

Transistor

Rectangle / Polygon-Group

Standard-Cell / Sub-cell

Macro-cell

Block / Chip

Chip / Board

Physical Domain

Y-transformations

Synthesis

Analysis

Behavioural Domain

Structural Domain

Refinement

Abstraction

Generation

Optimization

Extraction

Physical Domain
(Geometrical Domain)

TTÜ 1918

Synthesis – design automation

TTÜ 1918

Design automation – a bit idealistic view

- **1990 – 4 Kgates / year / designer**

- **1993 – inhouse place and route – 5.6K**

- **1995 – engineer (RTL–>GDSII) – 9.1K**

- **1997 – small blocks reuse (2.5K-75K) – 40K**

- **1999 – large blocks reuse (75K-1M) – 56K**

- **2001 – synthesis (RTL–>GDSII) – 91K**

- **2003 – intelligent testbench – 125K**

- **2005 – behavioral and architectural levels, HW/SW (co)design – 200K**

- **2007 – very large blocks reuse (>1M, IP cores) – 600K**

- **2009 – homogeneous parallel processing (multi-core) – 1200K**

- *Future – hw/sw co-verification, executable specification, etc.*

Market = $$$

- **Design cost**
  - *design time* & chips *production* cost
  - huge *investments* (G$)
  - almost impossible to *correct*

- **High cost of modifications**
  - large *production volumes* are more cost effective
  - *zero-defect* is very important
  - following *market trends* is important

- **Price is inversely proportional to production volume**
  - common purpose processors - cheap but not always usable
  - ASIC - application specific tuning (e.g. telecommunication)
  - prototypes - flexibility is extremely important in the development phase
  - special purpose chips (e.g. satellites)

- **Reconfigurability**
  - flexible products, possibility to modify working circuits

---

Design criteria

- **Three dimensions - area, delay, power**
  - size, speed, energy consumption
  - four dimensions - plus testability (reliability)

- **Area**
  - gates, wires, buses, etc.

- **Delay**
  - inside a module, between modules, etc.

- **Power consumption**
  - average, peak and total

- **Optimizations**
  - transferring from one dimension to another
  - design quality is measured by combined parameters, e.g., energy consumption per input sample

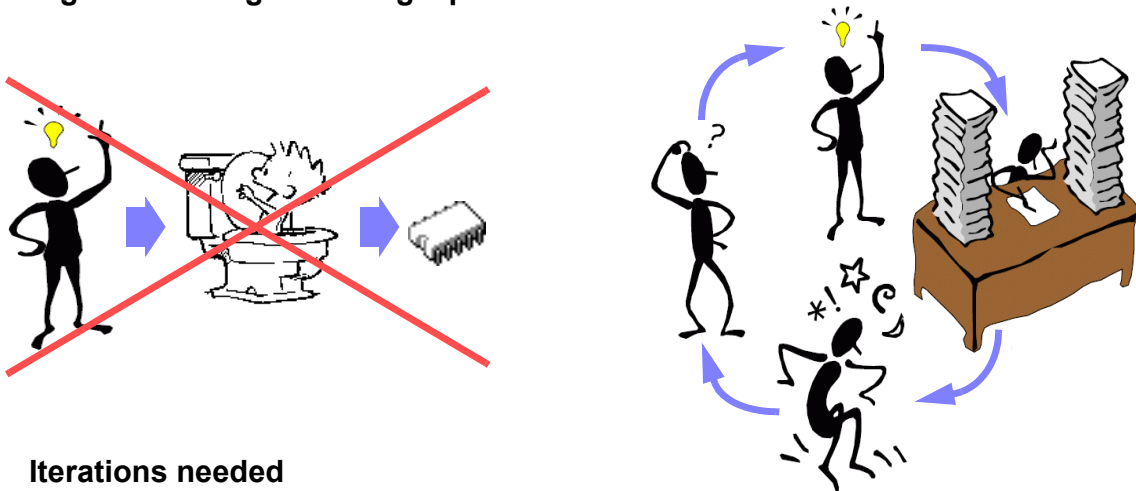# HDL – designing Systems-on-Chip (SoC) & Networks-on-Chip (NoC)

Idea	Language	Technology	Chip

!!!

VHDL
Verilog
Matlab

TTL

CMOS

GaAs

whatever

- **Fully automated flow from specification to implementation?**
  - **analysis, modeling, iterations etc. needed...**

MYTH #1

- **High level design is a single pass**

- **Iterations needed**
  - **Functionality**
  - **Design goals**

MYTH #2

- **Top down design, in its purest form, works**

Prove system function

Prove critical performance in specific technology

- **The pure breadth-first approach never actually works in practice**
  - **Bottom-up technology information must be considered early and often**
  - **Go depth-first for critical parts**
  - **Mix breadth-first with depth-first**

MYTH #3

- **You don't need to understand digital design anymore**

- **One must know hardware to get a good hardware**

- **Hope**
  - **Intimate knowledge of hardware is not necessary to design digital systems**

- **Fear**
  - **Using HDL based design methodology will turn them into software hackers**

- **Reality**
  - **High performance designs require a good deal of understanding about hardware**
  - **Designers must seed the synthesis tools with good starting points**
  - **Understanding the synthesis process is necessary to get good quality designs**

MYTH #4

- **Designer's job is just the functional specification now**

- **Specification   = Functionality +**
  Design goals +
  Operating conditions

- **Schematic capture**
  - **Design goals and operating conditions were implicit (in designer's mind)**
  - **Implementation was chosen (modified) to meet goals and conditions**

- **HDL**
  - **Design goals (area, speed, power, etc.) are explicitly specified**
  - **Operating conditions (variations, loads, drives) are also explicitly specified**

MYTH #5

- **Behavioral level is better than RTL**

- **Behavioral (a.k.a. algorithmic, high) level synthesis is not as mature as RTL (register-transfer level)**

- **If your specification includes enough timing information use RTL synthesis**

- **Behavioral constructs like *while*, *if-then-else* does not necessarily mean behavioral specification**
  - **... and/or can be misinterpreted**

TTÜ1918

Design process today

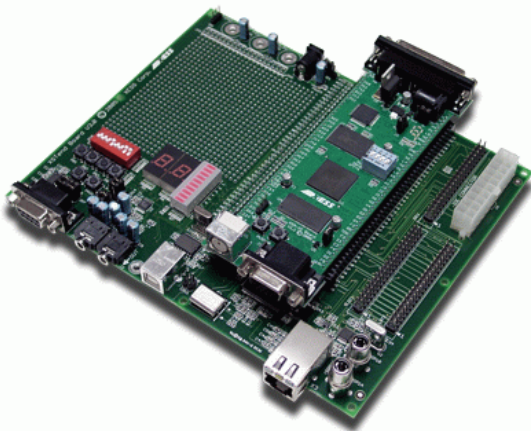- **Hardware Description Language**

```
--
-- Highway is green, sidestreet is red.
--
if  sidestreet_car = NoCar  then
  wait until  sidestreet_car = Car;
end if;
-- Waiting for no more than 25 seconds ...
if  highway_car = Car  then
  wait until  highway_car = NoCar  for 25 sec;
end if;
-- ... and changing lights
highway_light <= GreenBlink;
wait for 3 sec;
highway_light <= Yellow;
sidestreet_light <= Yellow;
wait for 2 sec;
highway_light <= Red;
sidestreet_light <= Green;
```
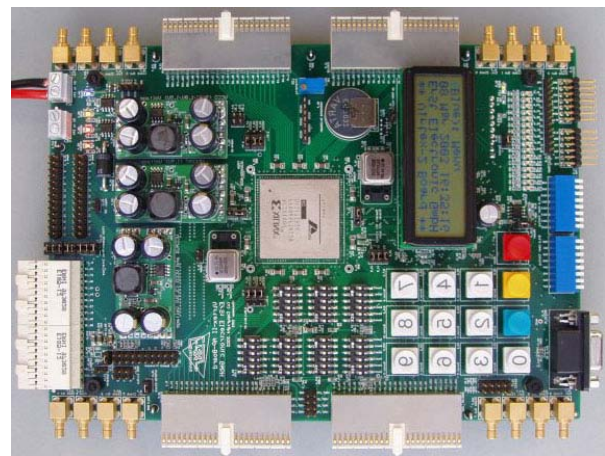

TTÜ1918

Prototyping

- **Possibility to check how a system works at conditions very close to the operating environment without the need to create expensive chips**
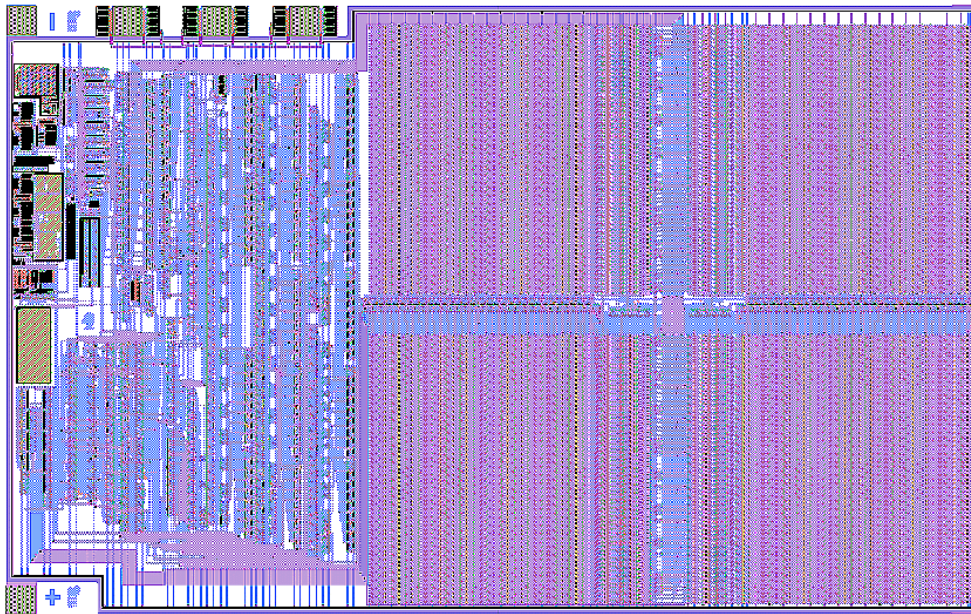

XESS Corp., www.xess.com, $400
[Xilinx Inc., FPGA XC3S1000]

ErST Electronics, www.erst.ch, 2340€
[Xilinx Inc., FPGA X2V2000]

Chip - the final result

Reality

- **Found On First Spin ICs/ASICs**
  - **Functional Logic Error -** **43%**
  - **Analog Tuning Issue -** **20%**
  - **Signal Integrity Issue -** **17%**
  - **Clock Scheme Error -** **14%**
  - **Reliability Issue -** **12%**
  - **Mixed Signal Problem -** **11%**
  - **Uses Too Much Power -** **11%**
  - **Has Path(s) Too Slow -** **10%**
  - **Has Path(s) Too Fast -** **10%**
  - **IR Drop Issues -** **7%**
  - **Firmware Error -** **4%**
  - **Other Problem -** **3%**
- **Overall 61% of New ICs/ASICs Require At Least One Re-Spin.**

- **Aart de Geus, Chairman & CEO of Synopsys, Boston SNUG keynote address, 9.09.2003**

Trends

	2000	2010	2020
memory size	2 Gbit	256 Gbit	1024 Gbit
transistors per cm^2	$8 \cdot 10^6$	$160 \cdot 10^6$	$480 \cdot 10^6$
internal clock frequency	1.5 GHz	10 GHz	40 GHz
external / bus clock frequency	0.5 GHz	1.5 GHz	2.5 GHz
pin count	2000	6000	10000
chip area	800 mm^2	1300 mm^2	1800 mm^2
wire width	140 nm	40 nm	10 nm
supply voltage	1.5 V	0.8 V	0.5 V
power consumption	100 W	170 W	300 W
power consumption (batteries)	0.5 W	1.5 W	2.5 W

NB! These are rough estimates only!

Problems

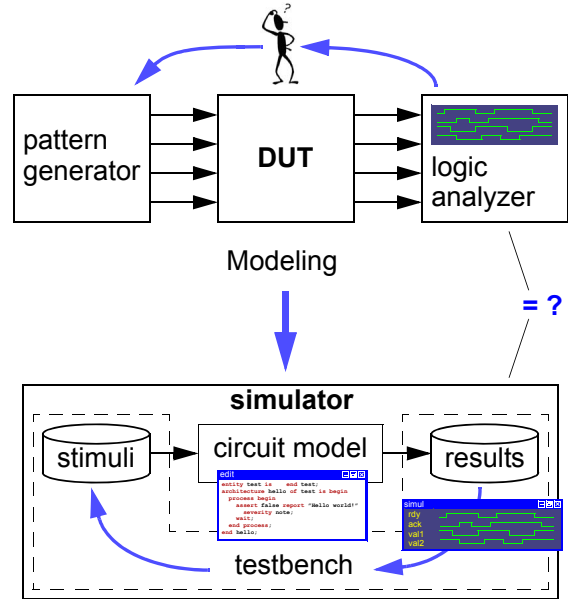| physical level	quantum effects
	noise
logic level	crosstalk
	speed of light
system level	# of transistors

10 mm –> (10^8 m/s) –>
10^{-10} s –> (10%) –>
1 GHz !?

- • **GALS – globally asynchronous locally synchronous**
- • **mixed signal – digital and analog circuits on the same chip**
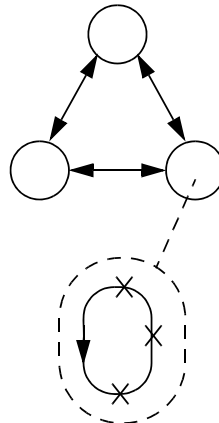- • **Modelling is getting more and more important...**

Simulation

- **Simulation = modeling + analysis**

- **Logic level simulation**

- **RT-level simulation**

- **Functional level simulation**
  - **Behavioral level simulation**

- **System level simulation**

- **Test environment == stimuli generator + DUT + results analyzer**

- **NB! Detailed simulation is slow!**

Modeling

= ?

Use of HDL –> Simulation

- **Simulation = modelling + analysis**
  - **Logic / register-transfer / functional (behavioral) / system level simulation**

concurrent / parallel modules

connected via signal / channels

sequential vs. concurrent execution?

execution order?!

current / new values to avoid non-determinism

event queue history+future

module / unit / process

continuous execution is slow

only when needed?

time / event triggered
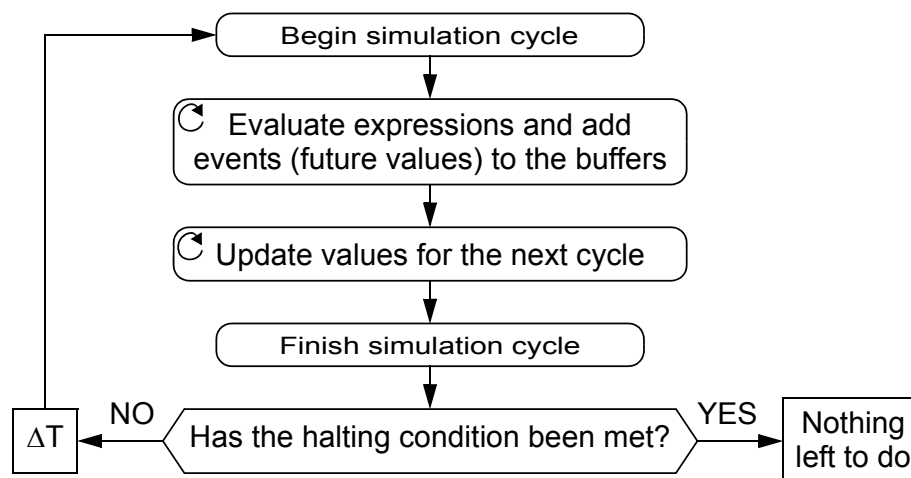
different simulation engines

Simulators

- *Time-driven*:
  all components of the digital logic system are evaluated at every time step

- *Event-driven*:
  system input events are kept in an time-ordered event queue

Simulation delay models

- **Unit-delay (RTL simulator)**
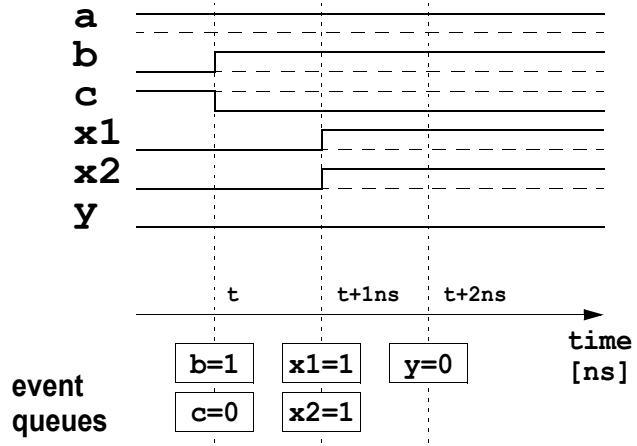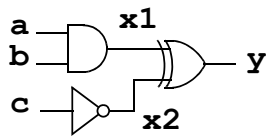
- **Zero-delay**

- *delta*-delay (VHDL) – δ-delay, Δ-delay

Unit-delay simulation model

TTÜ1918

Unit-delay simulation model (example)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```


event queues

TTÜ1918

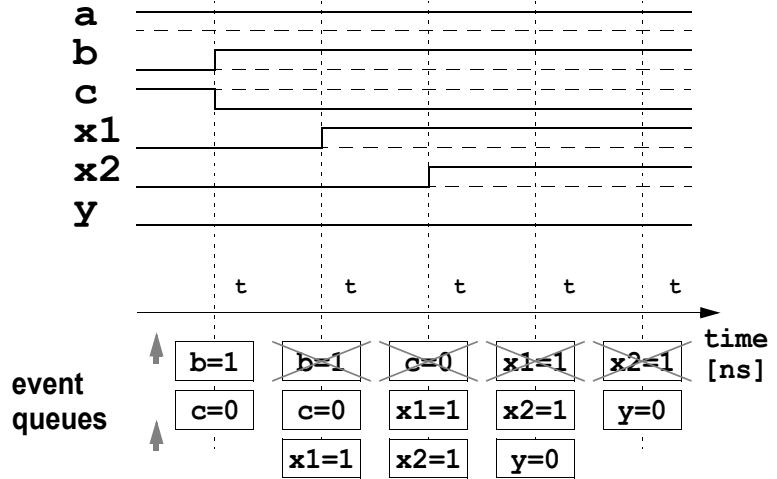# Zero-delay simulation model (Verilog)

Begin simulation cycle

Immediately evaluate and update
Add new event to the queue

Are there more events
scheduled for this moment?

YES

NO

Finish simulation cycle

NO Has the halting condition been met? YES

ΔT

Nothing
left to do

Zero-delay simulation model (example #1)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```

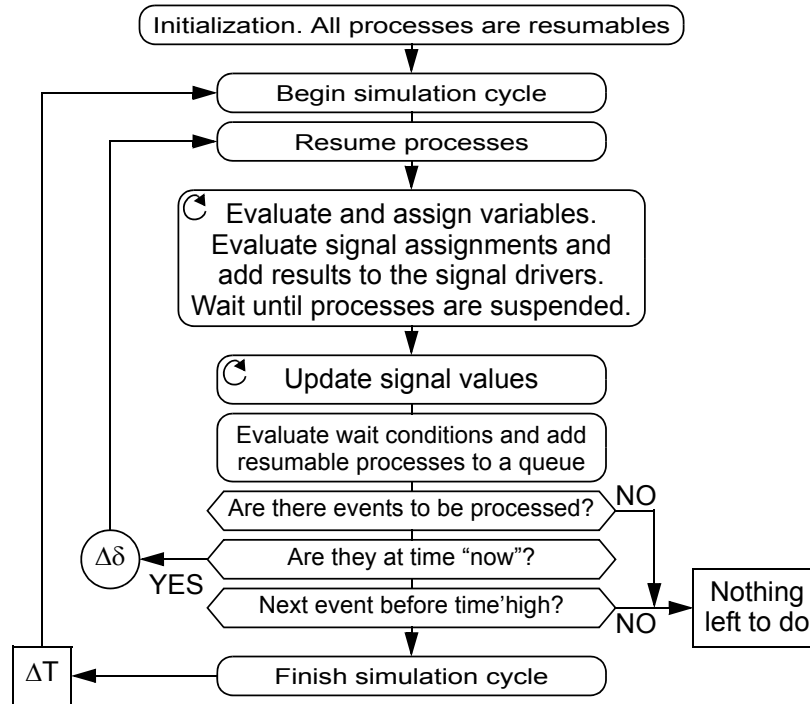## Zero-delay simulation model (example #2)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```

Delta-delay (VHDL) simulation model

Initialization. All processes are resumables

Begin simulation cycle

Resume processes

↻ Evaluate and assign variables.
Evaluate signal assignments and
add results to the signal drivers.
Wait until processes are suspended.

↻ Update signal values

Evaluate wait conditions and add
resumable processes to a queue

Are there events to be processed?　NO

Are they at time "now"?

Δδ

YES

Next event before time'high?　NO

Nothing
left to do

ΔT　Finish simulation cycle

Delta-delay (VHDL) simulation model (example)

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```

a
b
c
x1
x2
y

a
b
c
x1
x2
y

t　　t+Δ　　t+2Δ

**time
[ns]**

**event
queues**

| b=1 | x1=1 | y=0 |
| c=0 | x2=1 | |