Digital Systems Modeling and Verification

Design of Digital Systems

- **Design phases in general**
  - specification, modeling, synthesis, verification
- **Synthesis itself has many phases**
  - step-by-step refinement
  - different abstraction levels require different approaches
  - details define automation possibilities
- **Some good books**
  - Dirk Jansen et al. (editors), "The electronic design automation handbook."
  - Michael John Sebastian Smith, "Application-Specific Integrated Circuits."
  - More at EDACafé – https://edacafe.com/aws/book_list.php

Motivation

- Multi and many core systems on chip (SoC, NoC, etc.) require 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

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

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

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

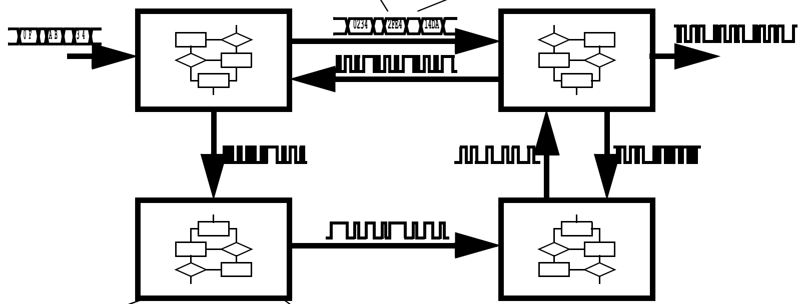
TTÜ1918

Abstraction levels

- **System level**
  - **modules / methods**
  - **channels / protocols**

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

```
message:
     header1 & payload1
   | header2 & payload2
   | header3 & payload1
   | illegal
```



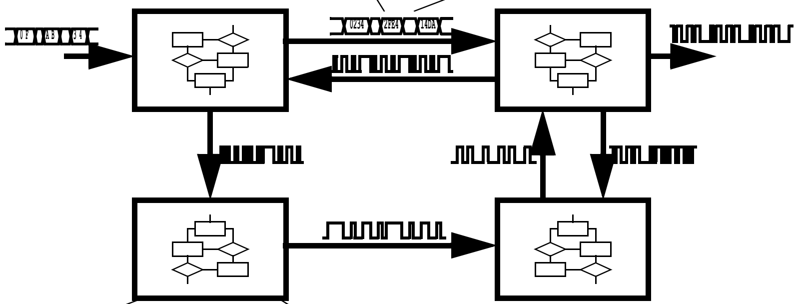
```
. . .

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

. . .
```

Abstraction levels

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



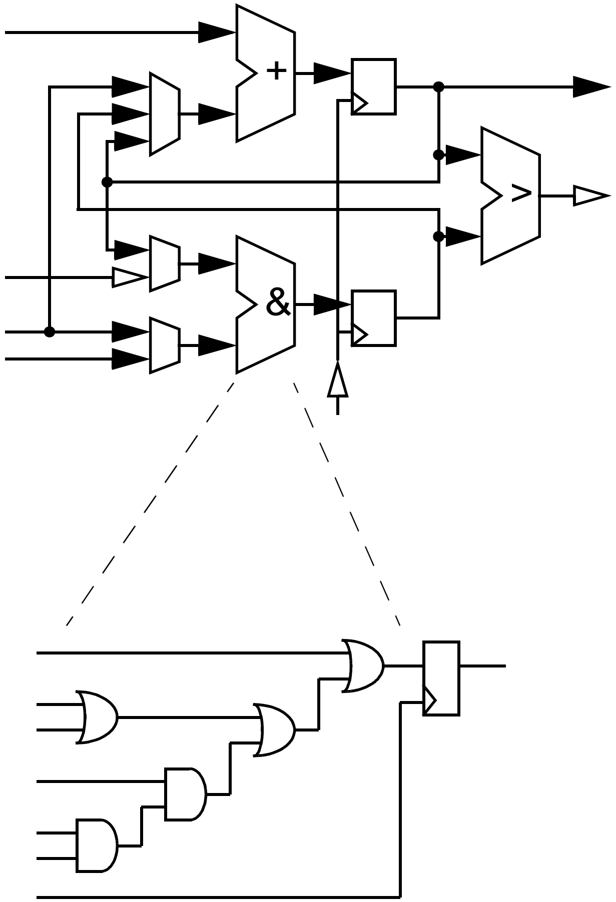
```
. . .
message=blocking_receive(channel_1);
append(list,message);
bubble_sort(list);
msg_pnt=first(list);
message= *msg_pnt;
nonblocking_send(message,channel_2);
remove(list,msg_pnt);
. . .
```

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

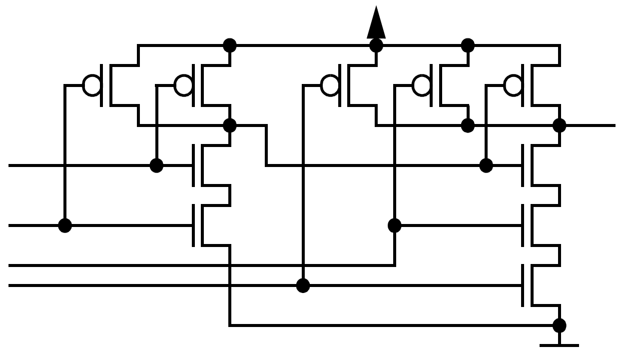
- **High-Level Synthesis**
  - **Resource or time constrained scheduling.**
  - **Resource allocation. Binding.**

Abstraction levels

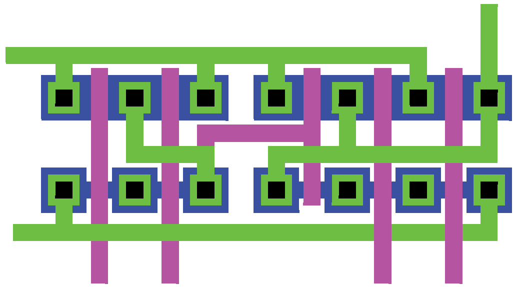
- Register transfer (RT) level
  - blocks / logic expressions
  - buses / words
- Level Synthesis
  - Data-path synthesis.
    Controller synthesis.
- Logic level
  - logic gates / logic expressions
  - nets / bits
- Logic Level Synthesis
  - Logic minimization.
    Optimization, overhead removal.



Abstraction levels



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

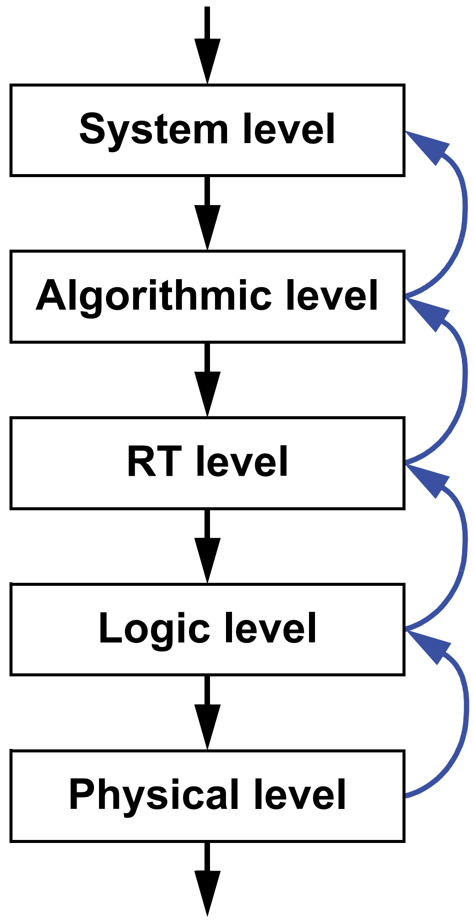


- **Physical Level Synthesis**
  - **Library mapping. Placement. Routing.**

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

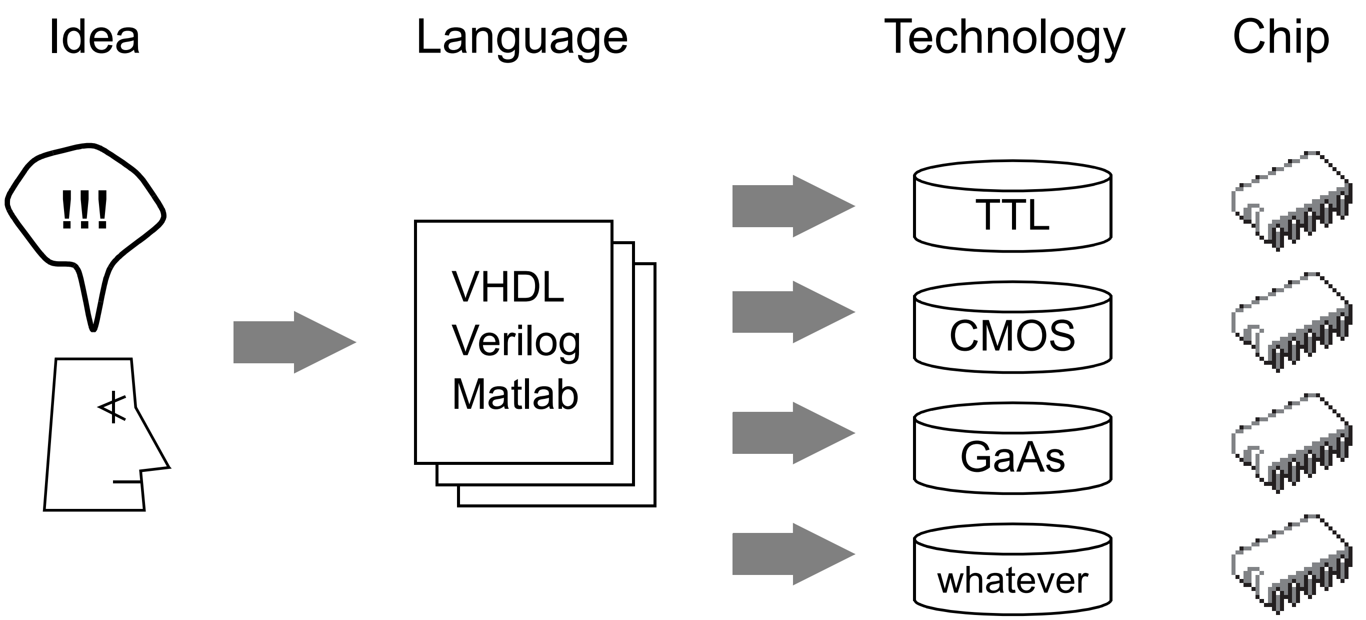
HW design flow



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

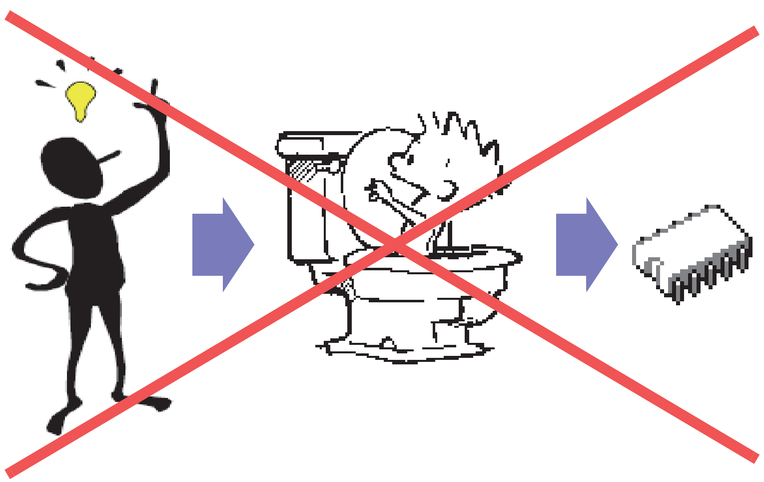
HDL – designing Systems-on-Chip (SoC) & Networks-on-Chip (NoC)



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

MYTH #1

- High level design is a single pass

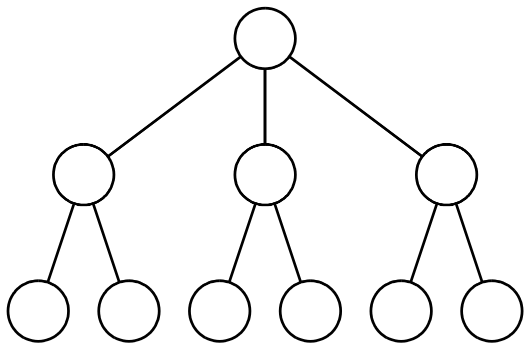


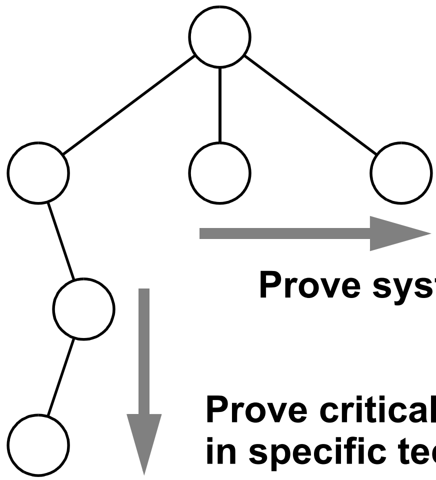
- Iterations needed
  - Functionality
  - Design goals

PDSA - Plan, Do, Study, Act
PDCA - Plan, Do, Check, Adjust

MYTH #2

- Top down design, in its purest form, works





- 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

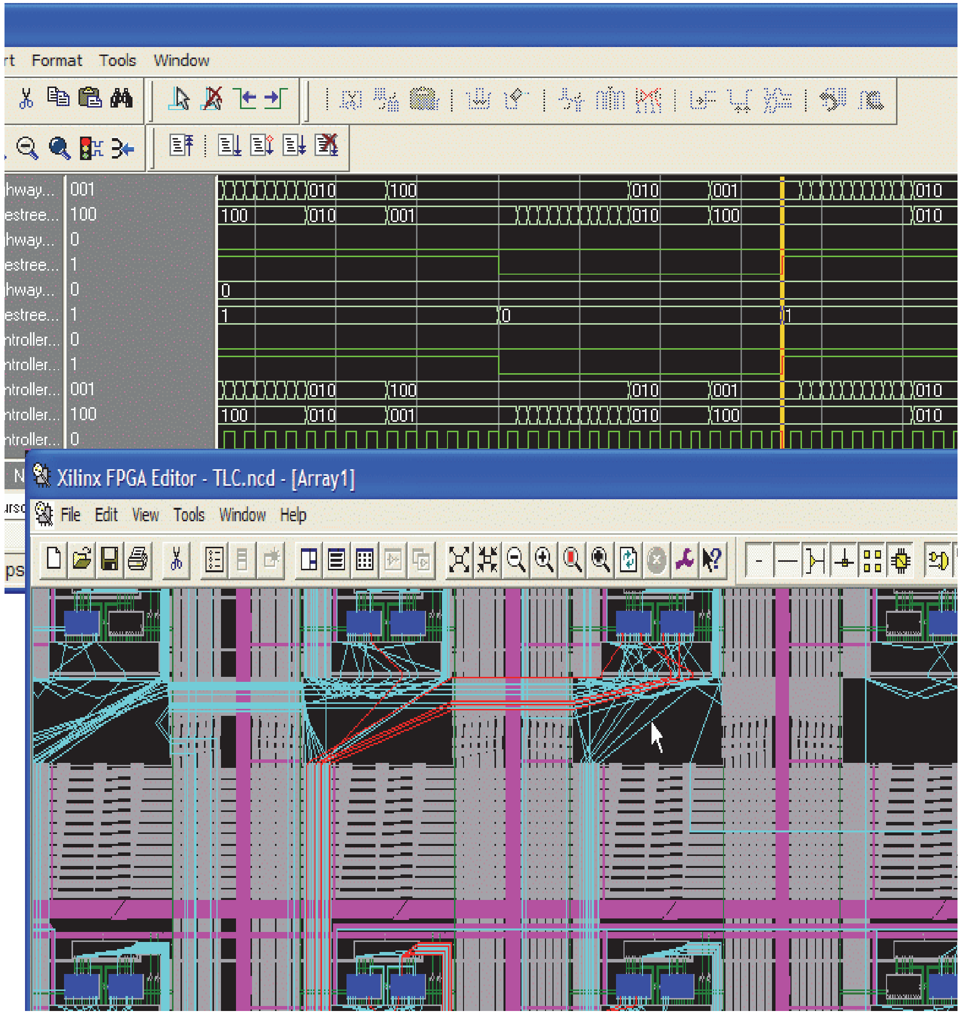
- Schematic capture
  - Design goals and operating conditions were implicit (in designer's mind)
  - Implementation was chosen (modified) to meet goals and conditions

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

Design process today

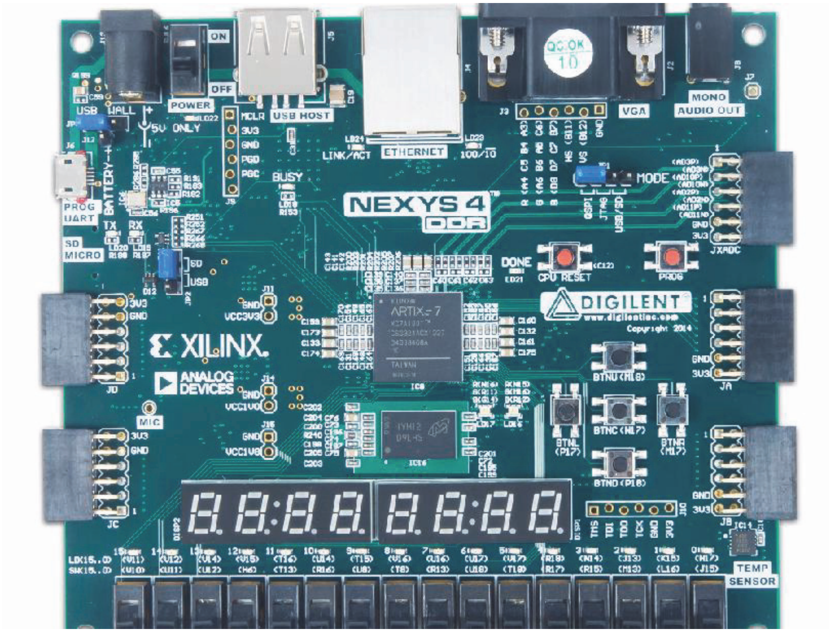
- **Hardware Description Language**

```
--
-- 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;
```

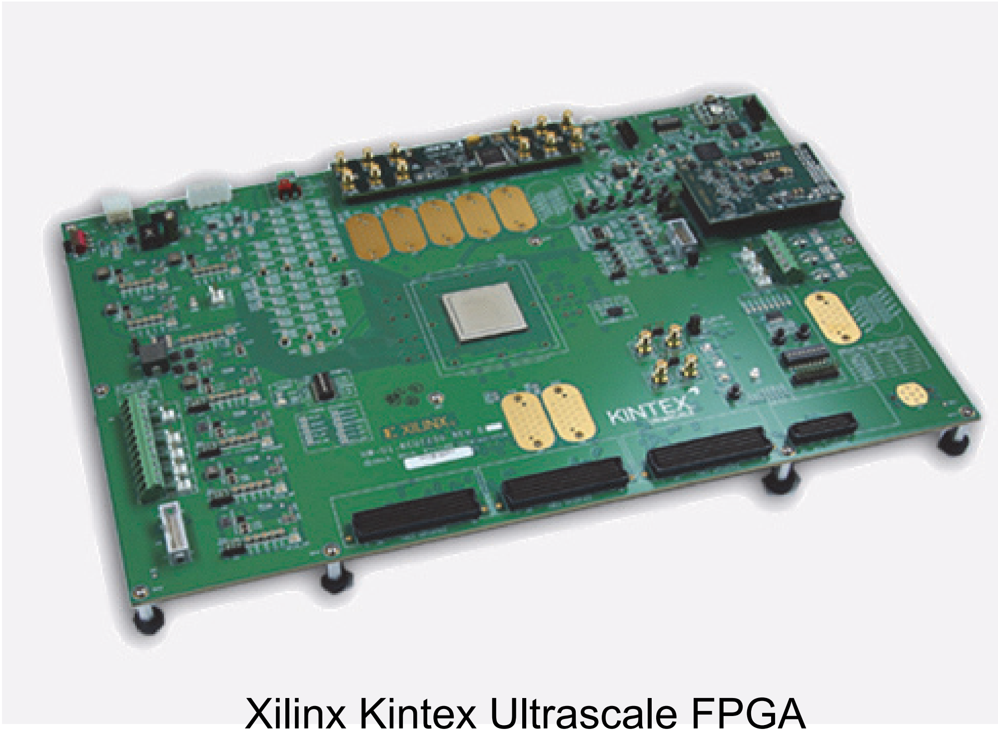


Prototyping

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

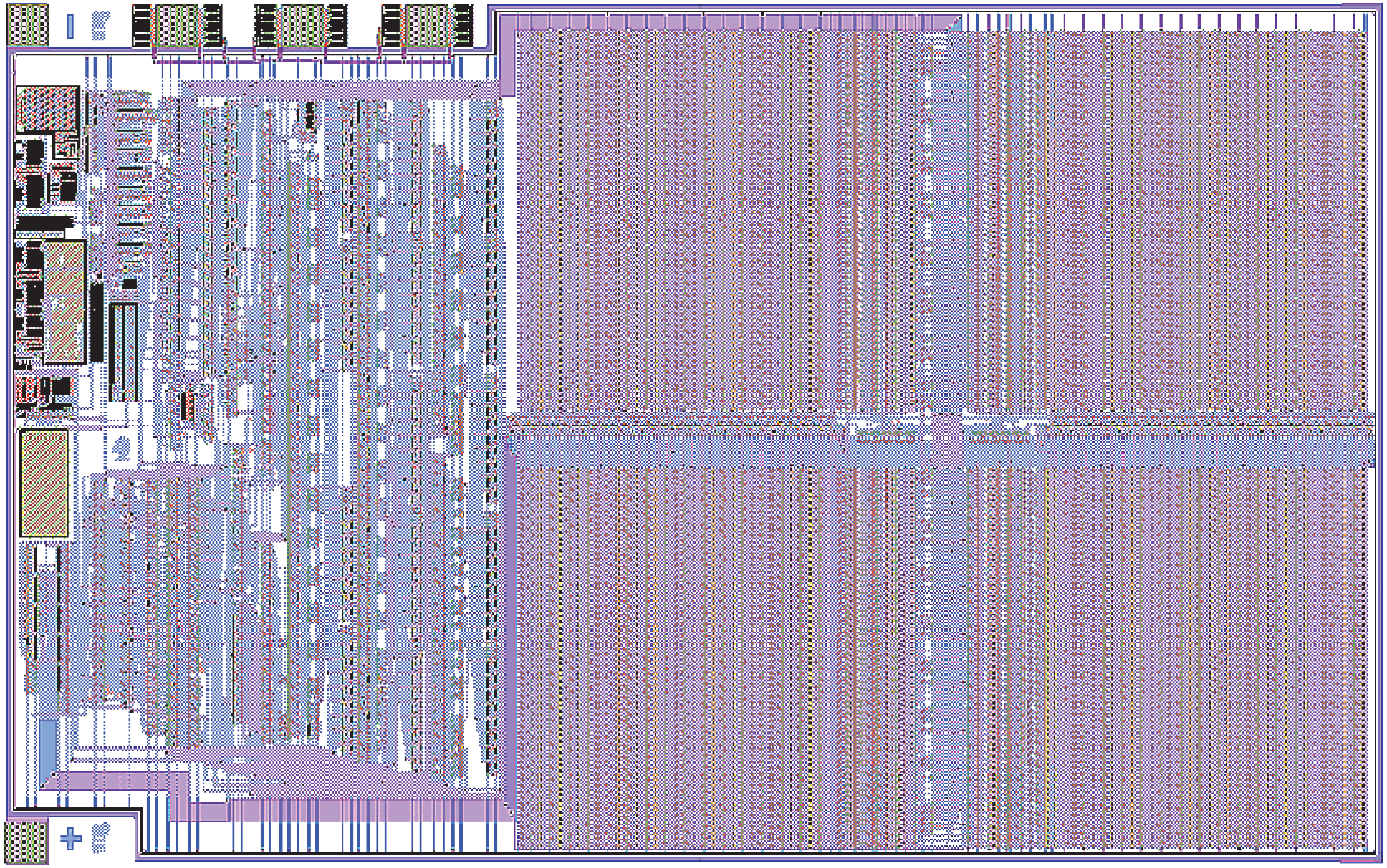


Digilent Nexys 4 DDR Artix-7 FPGA
[XC7A100T-1CSG324C; $320]



Xilinx Kintex Ultrascale FPGA
KCU1250 Characterization Kit
[XCKU040-2FFVA1156E; $7,495]

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 so far

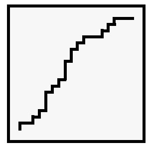
	2000	2010	2020
memory size	2 Gbit	256 Gbit	1024 Gbit
transistors per cm^2	$8 \cdot 10^6$	$160 \cdot 10^6$	$480 \cdot 10^6$
internal clock frequency	1.5 GHz	10 GHz	40 GHz
external / bus clock frequency	0.5 GHz	1.5 GHz	2.5 GHz
pin count	2000	6000	10000
chip area	800 mm^2	1300 mm^2	1800 mm^2
wire width	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 approximations only!

- **International Roadmap for Devices and Systems™ – https://irds.ieee.org/**

Problems

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



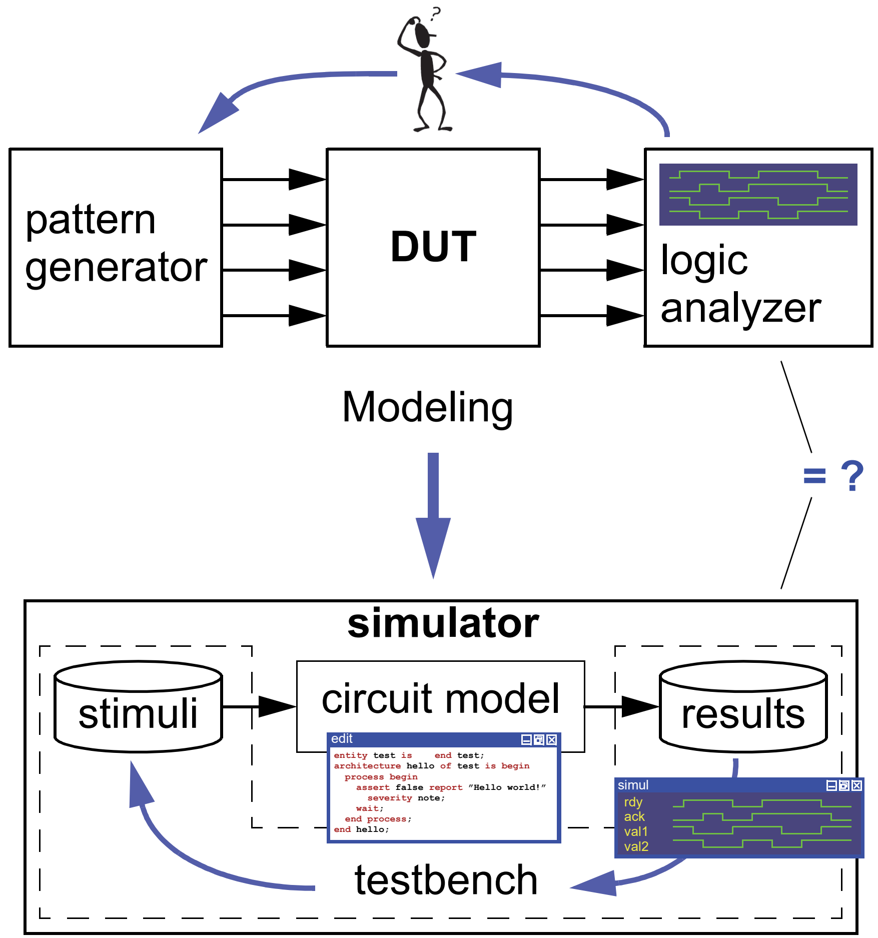
10 mm –> (10^8 m/s) –>
10^{-10} s –> (10%) –>
1 GHz !?

- GALS – globally asynchronous locally synchronous
- mixed signal – digital and analog circuits on the same chip

- **Modeling is getting more and more important...**

Simulation = Modeling + Analysis

- An important phase in the design flow
- 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!



Use of HDL –> Simulation

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

concurrent / parallel modules

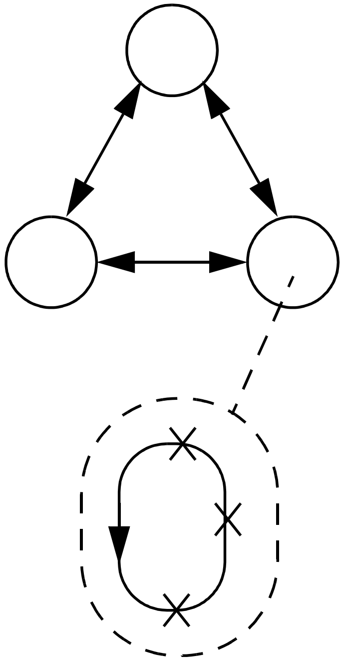
connected via signal / channels

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?

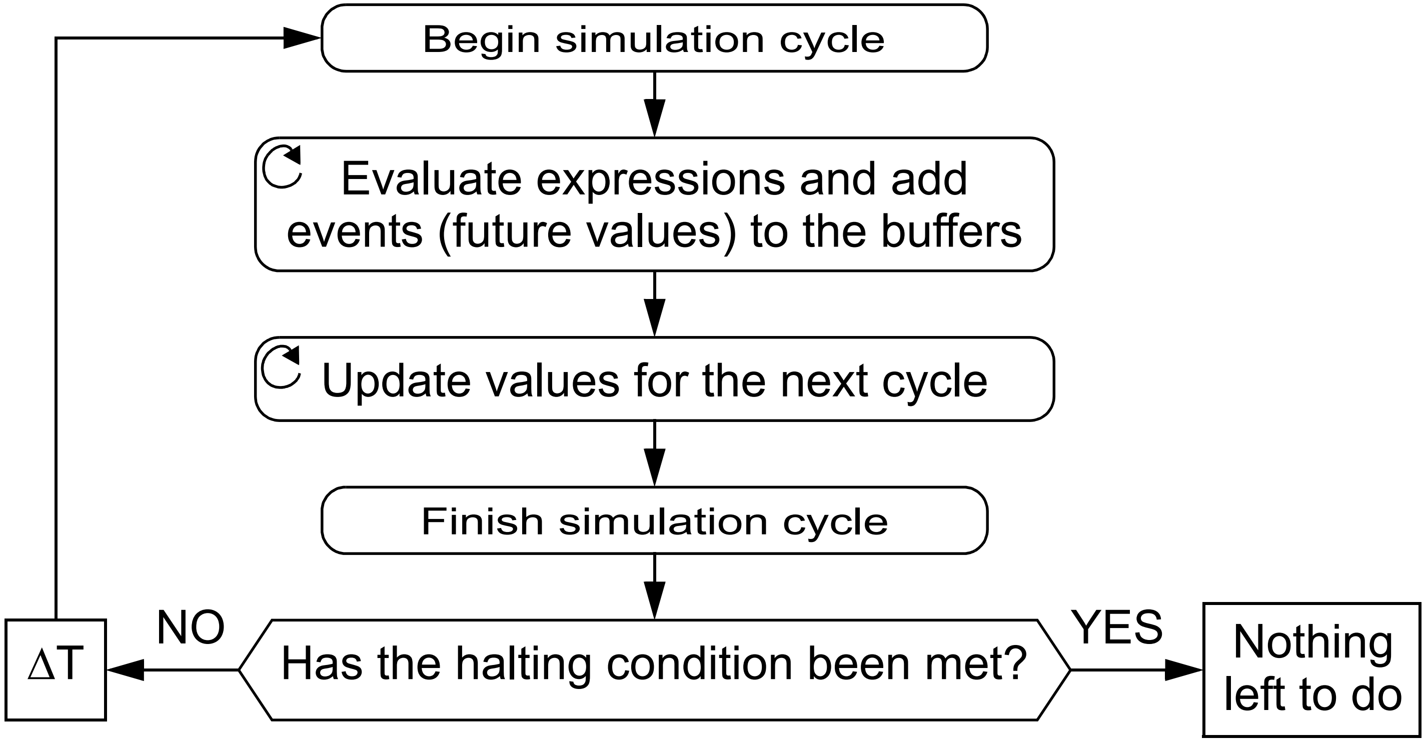
time / event triggered

different simulation engines

Simulators & timing/delay models

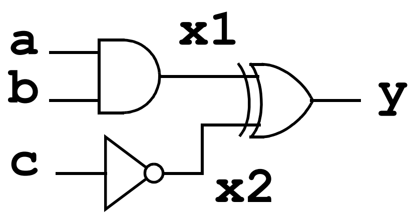
- Time & events
  - *Time-driven*:
    all components of the digital logic system are evaluated at every time step
  - *Event-driven*:
    system input events are kept in an time-ordered event queue
- Delay models
  - unit-delay (RTL simulator)
  - zero-delay (Verilog)
  - *delta*-delay (VHDL) – δ-delay, Δ-delay
- Simulation engines
  - all make use of the three following steps but details differ...
    - (1) calculate (and remember) new values for signals
    - (2) update signal values
    - (3) update time

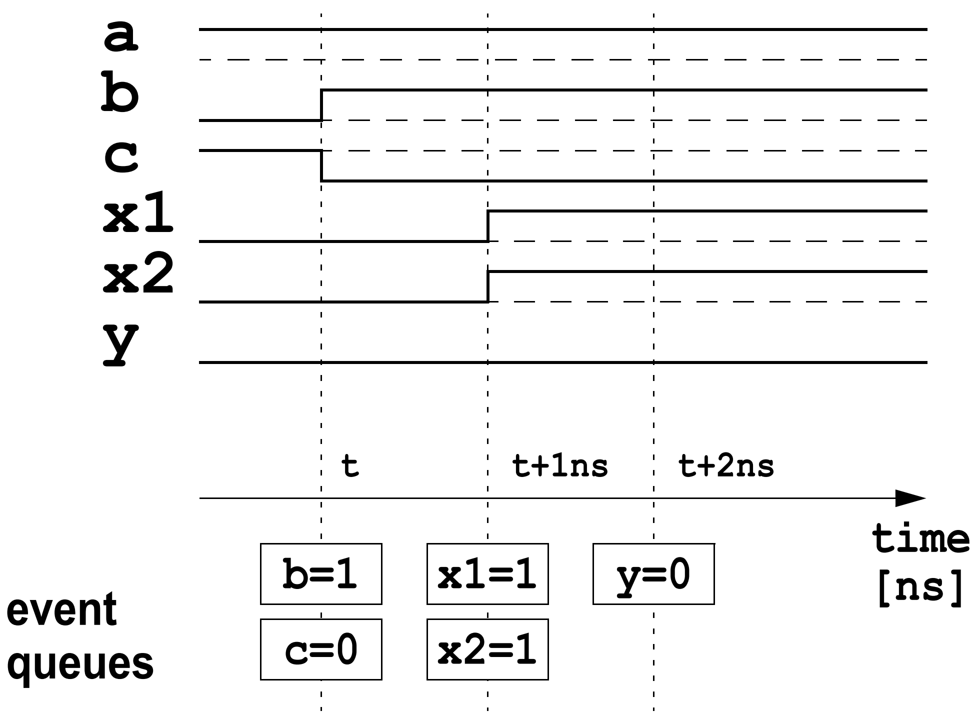
Unit-delay simulation model



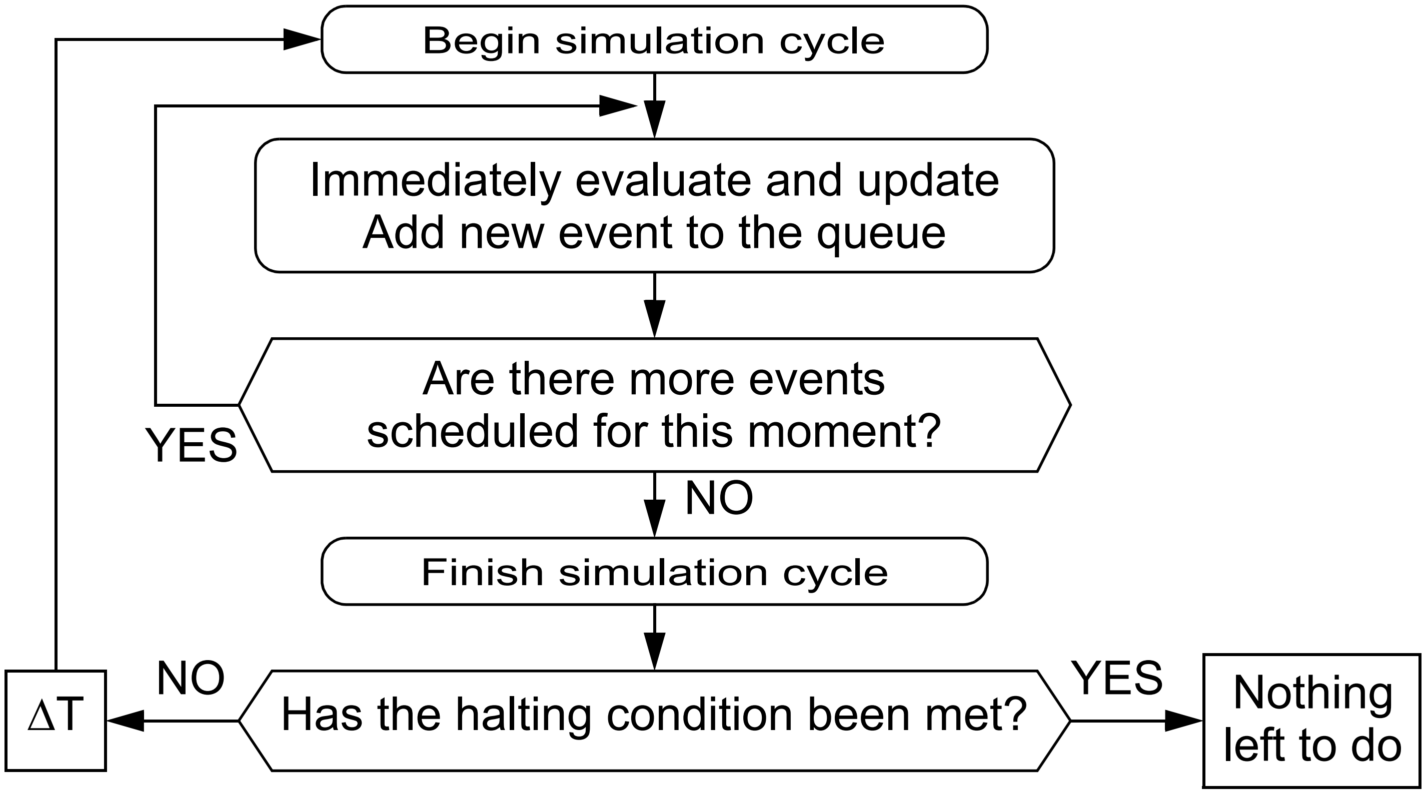
Unit-delay simulation model (example)

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



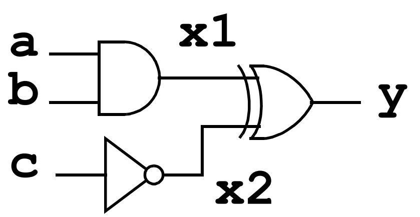


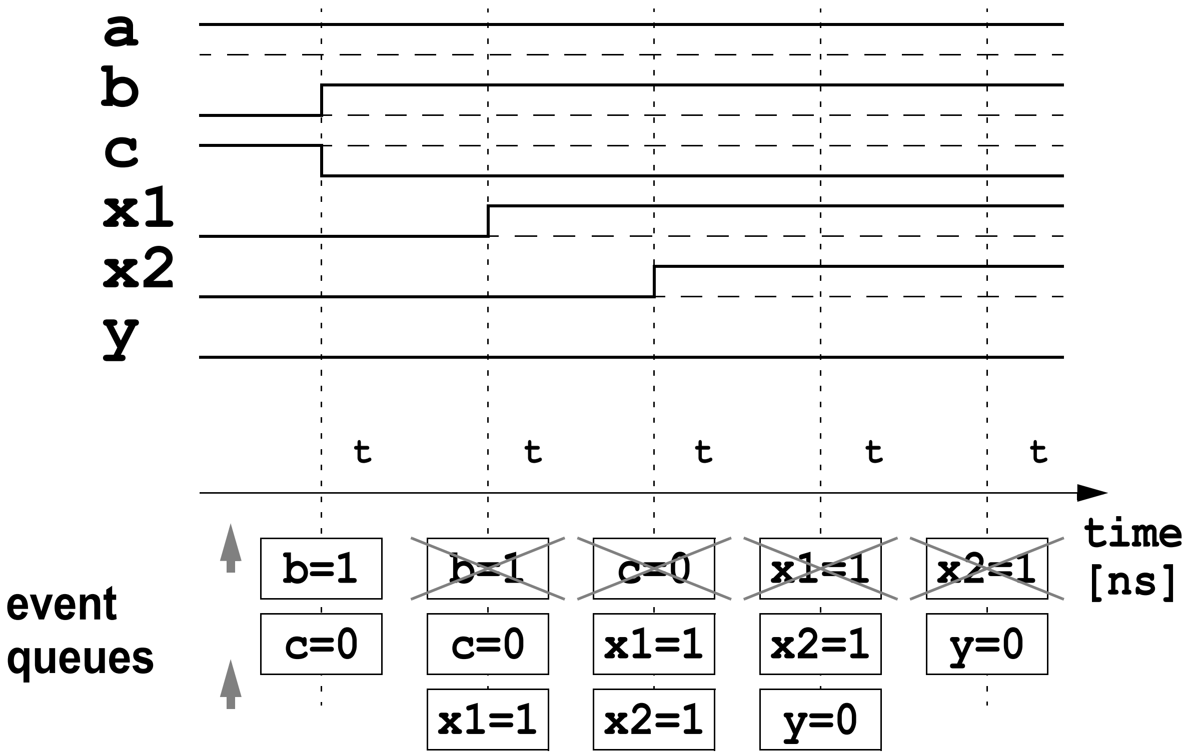
Zero-delay simulation model (Verilog)



Zero-delay simulation model (example #1)

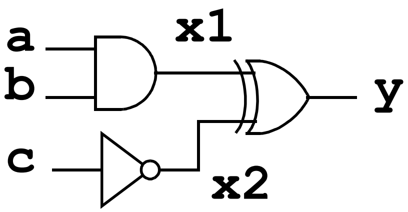
```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```

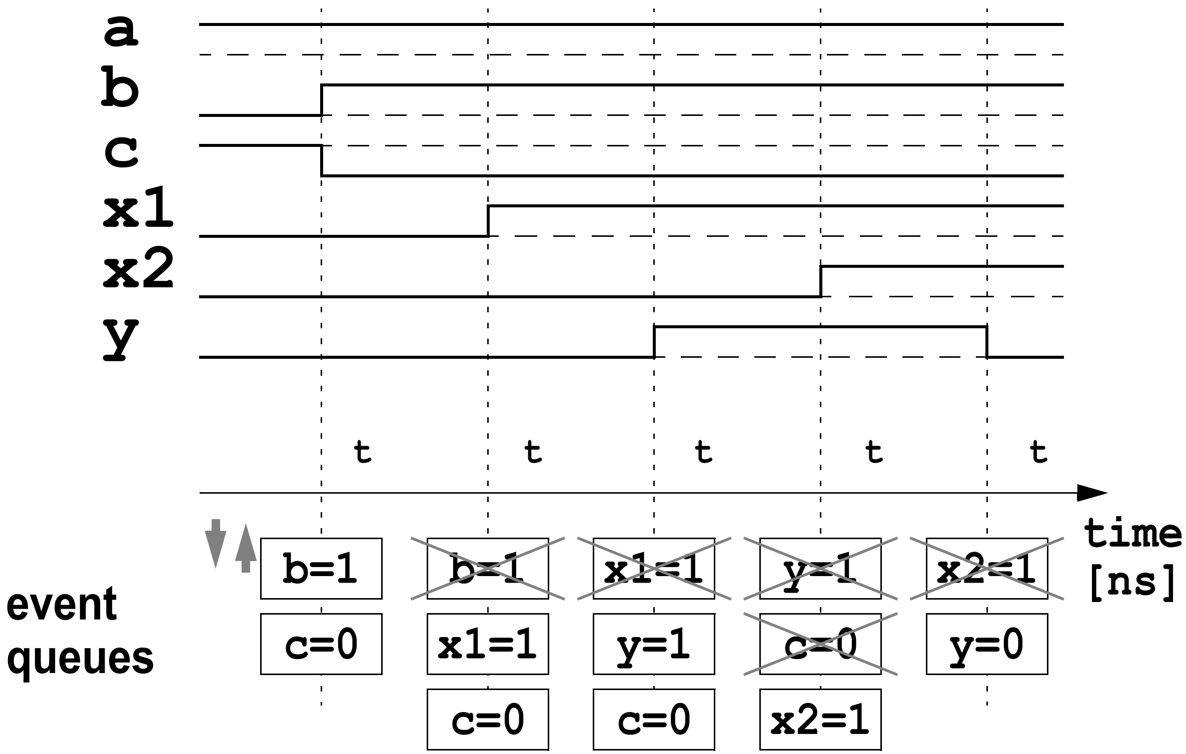




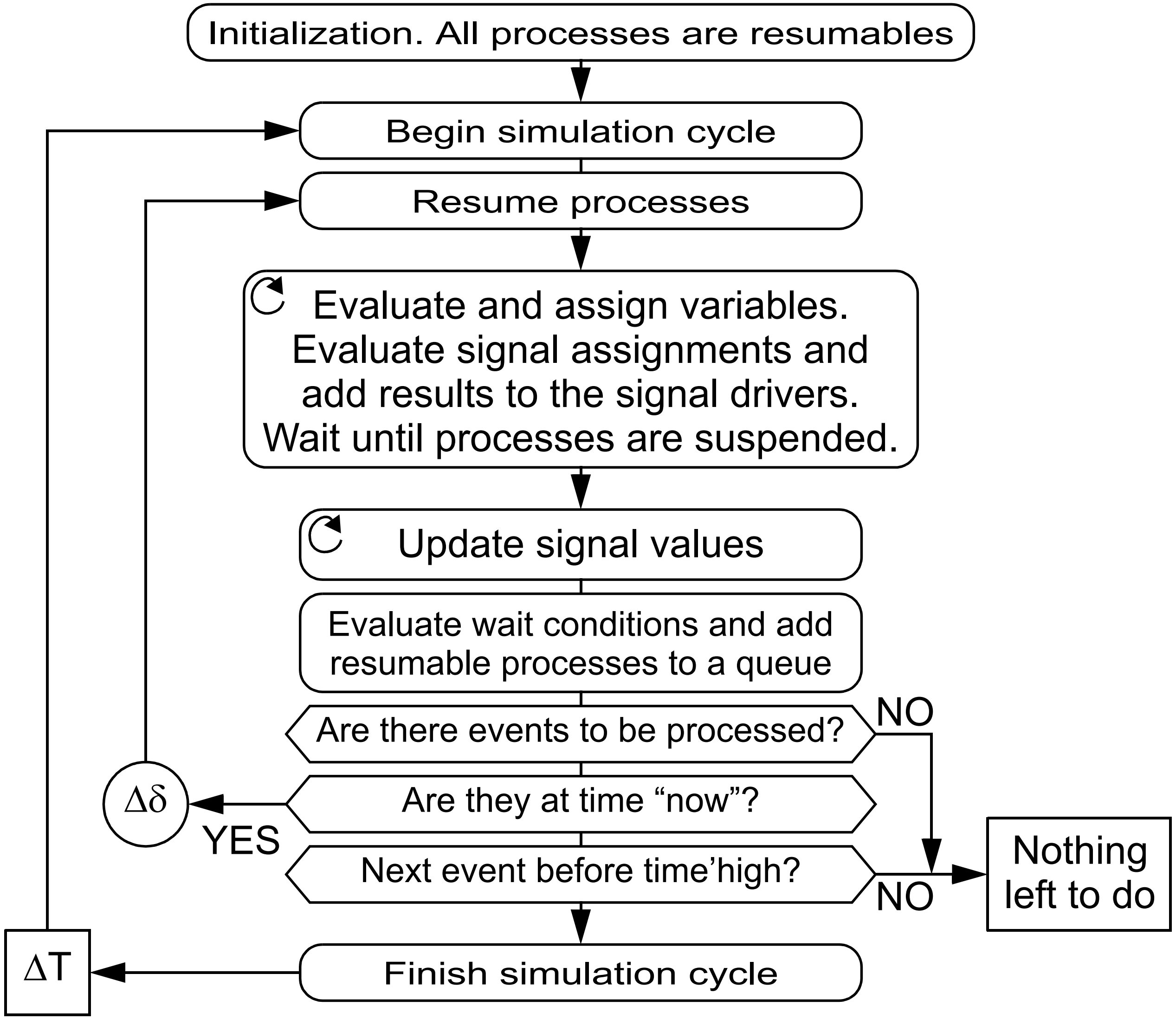
Zero-delay simulation model (example #2)

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





Delta-delay (VHDL) simulation model



Delta-delay (VHDL) simulation model (example)

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