



Computer Aided Design of Digital Systems

Synthesis 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



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 hundreds 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**
- **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

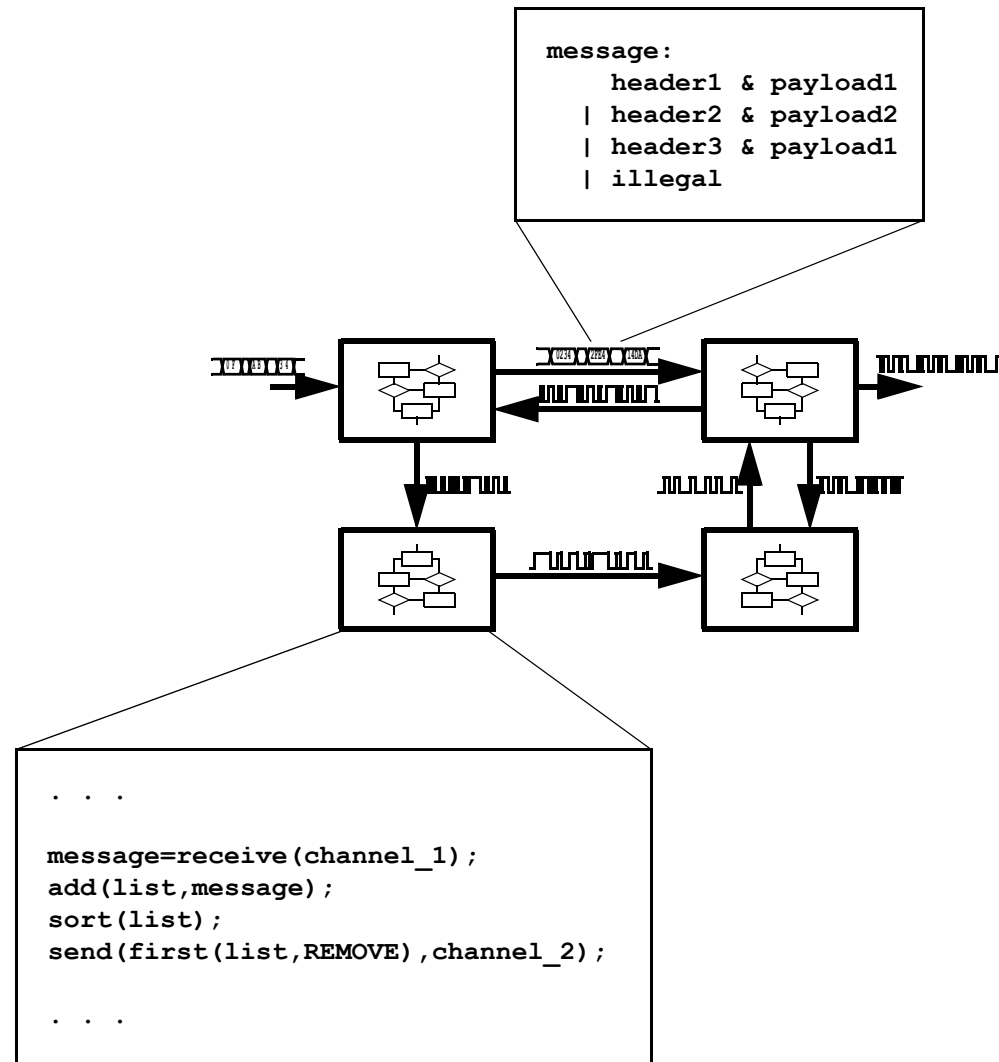


Design steps

- **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

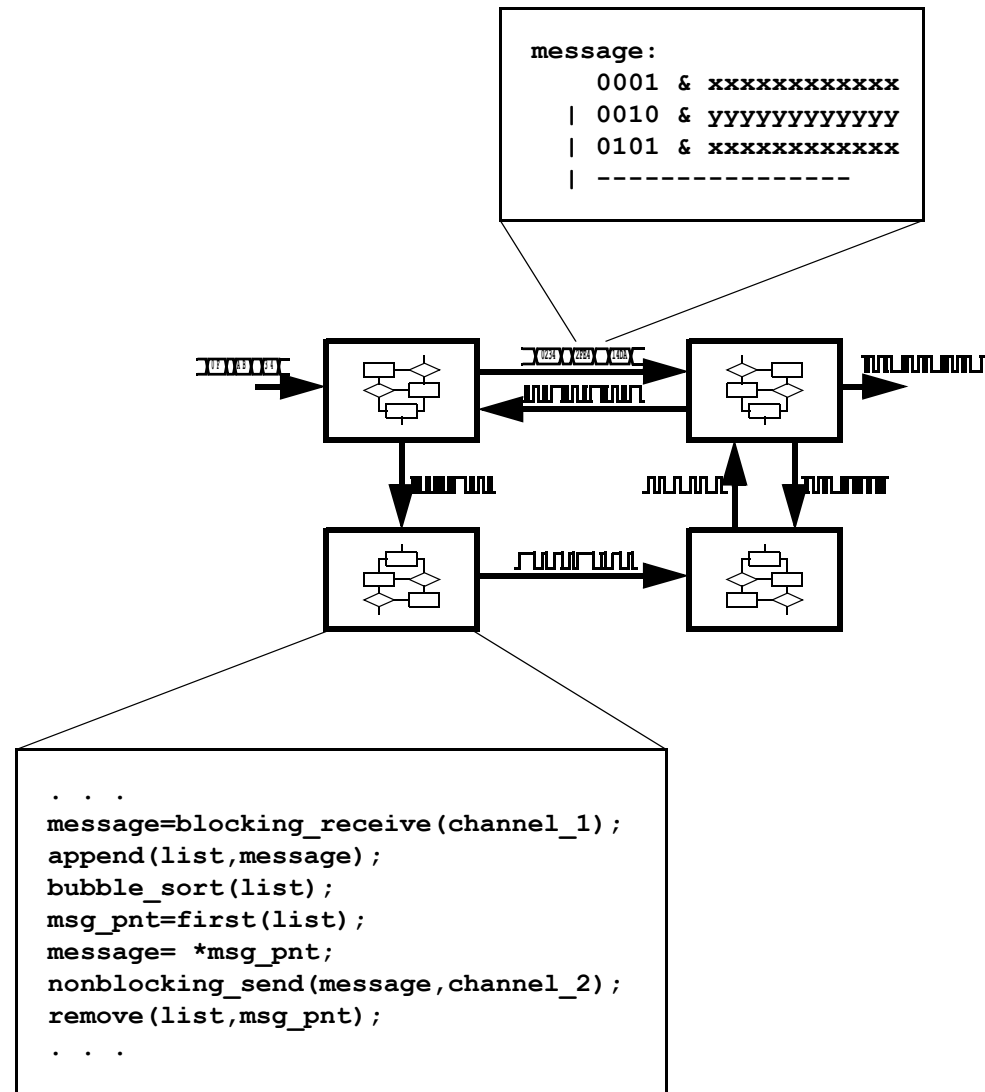
Abstraction levels

- **System level**
 - modules / methods
 - channels / protocols
- **System Level Synthesis**
 - Clustering.
Communication synthesis.



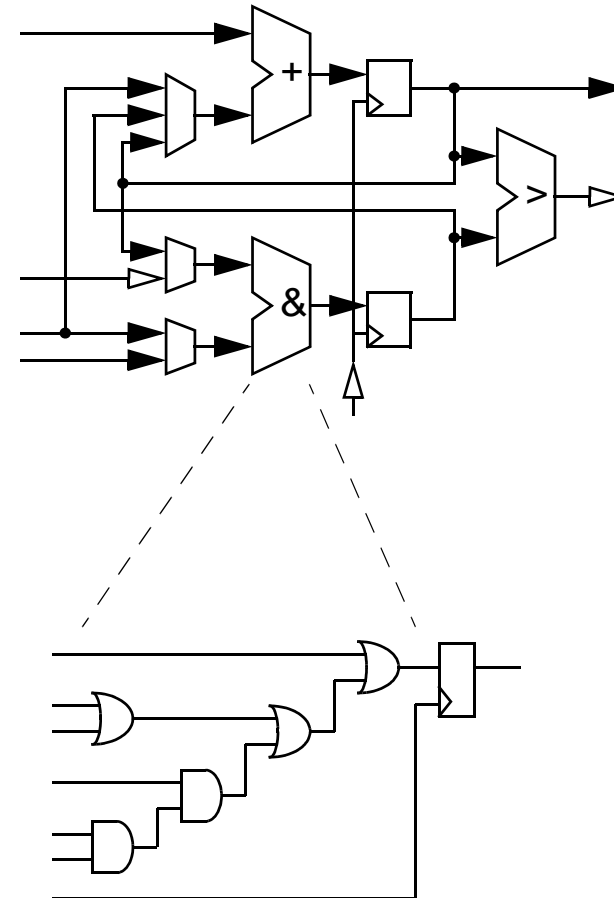
Abstraction levels

- **Algorithmic level**
 - (sub)modules / algorithms
 - buses / protocols
- **High-Level Synthesis**
 - Resource or time constrained scheduling.
 - Resource allocation. Binding.



Abstraction levels

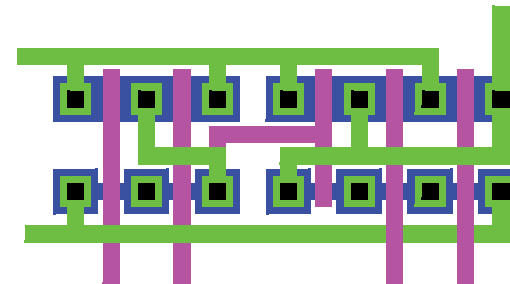
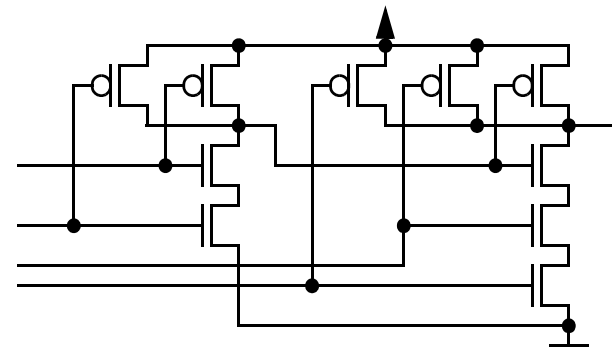
- **Register transfer (RT) level**
 - blocks / logic expressions
 - buses / words
- **RT Level Synthesis**
 - Data-path synthesis.
Controller synthesis.
- **Logic level**
 - logic gates / logic expressions
 - nets / bits
- **Logic Level Synthesis**
 - Logic minimization.
Optimization, overhead removal.



Abstraction levels

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

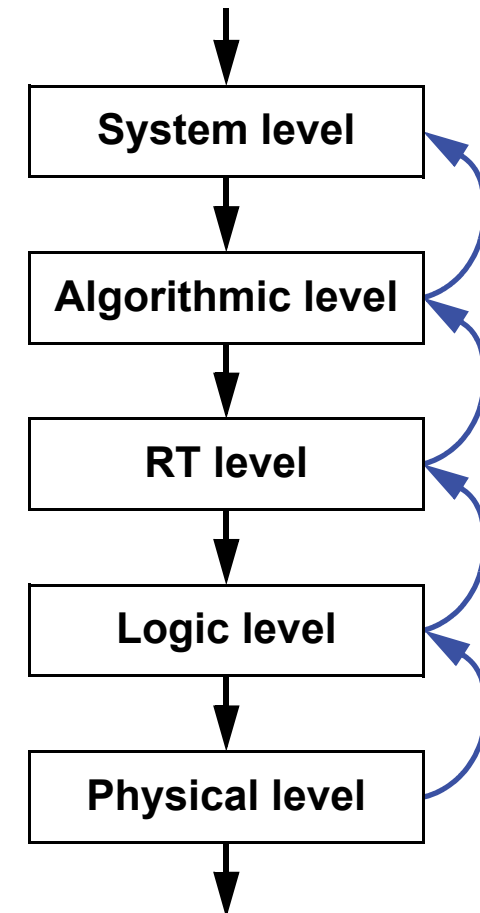
- **Physical Level Synthesis**
 - Library mapping.
Placement. Routing.



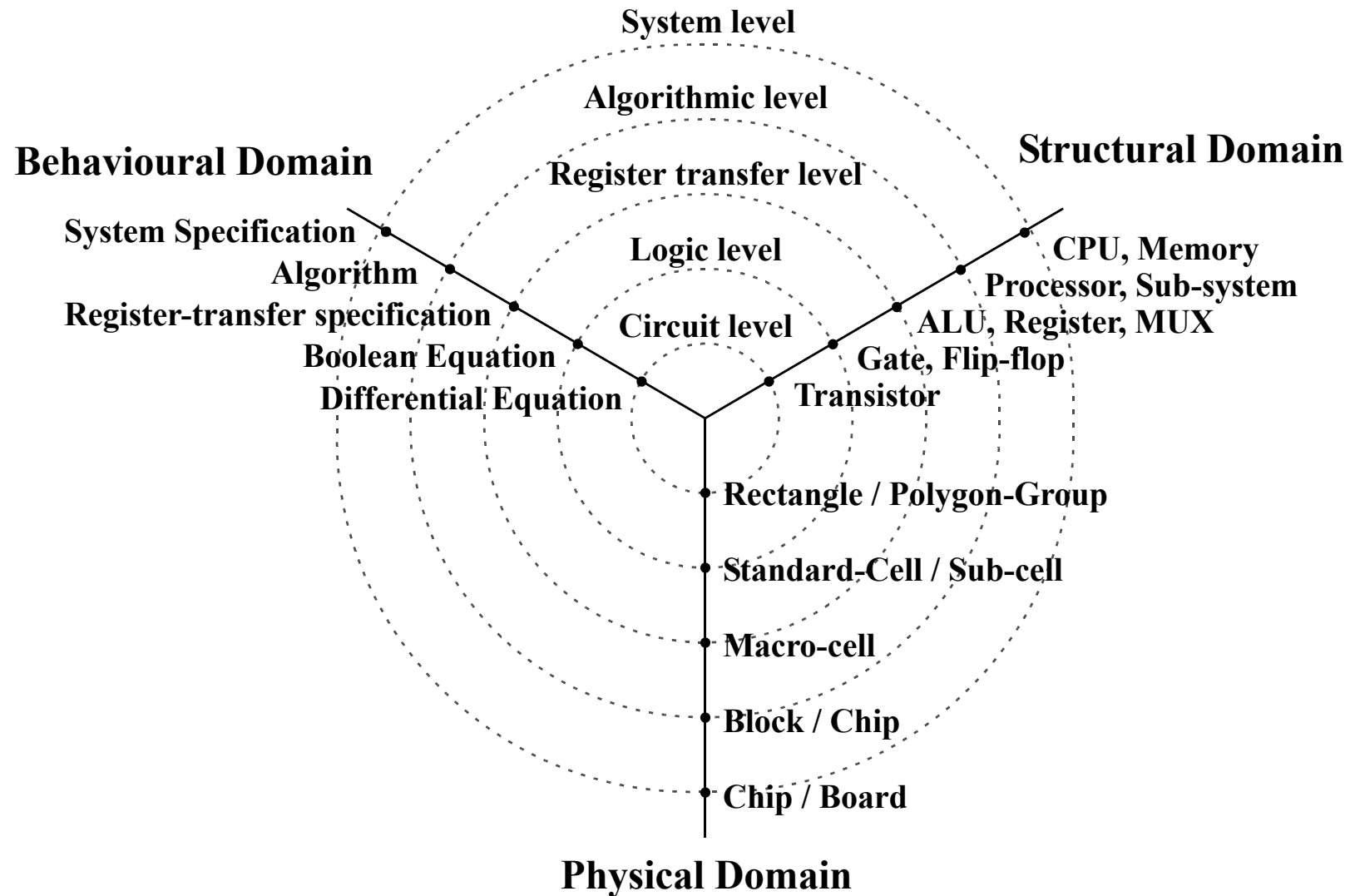
Design flow

- **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

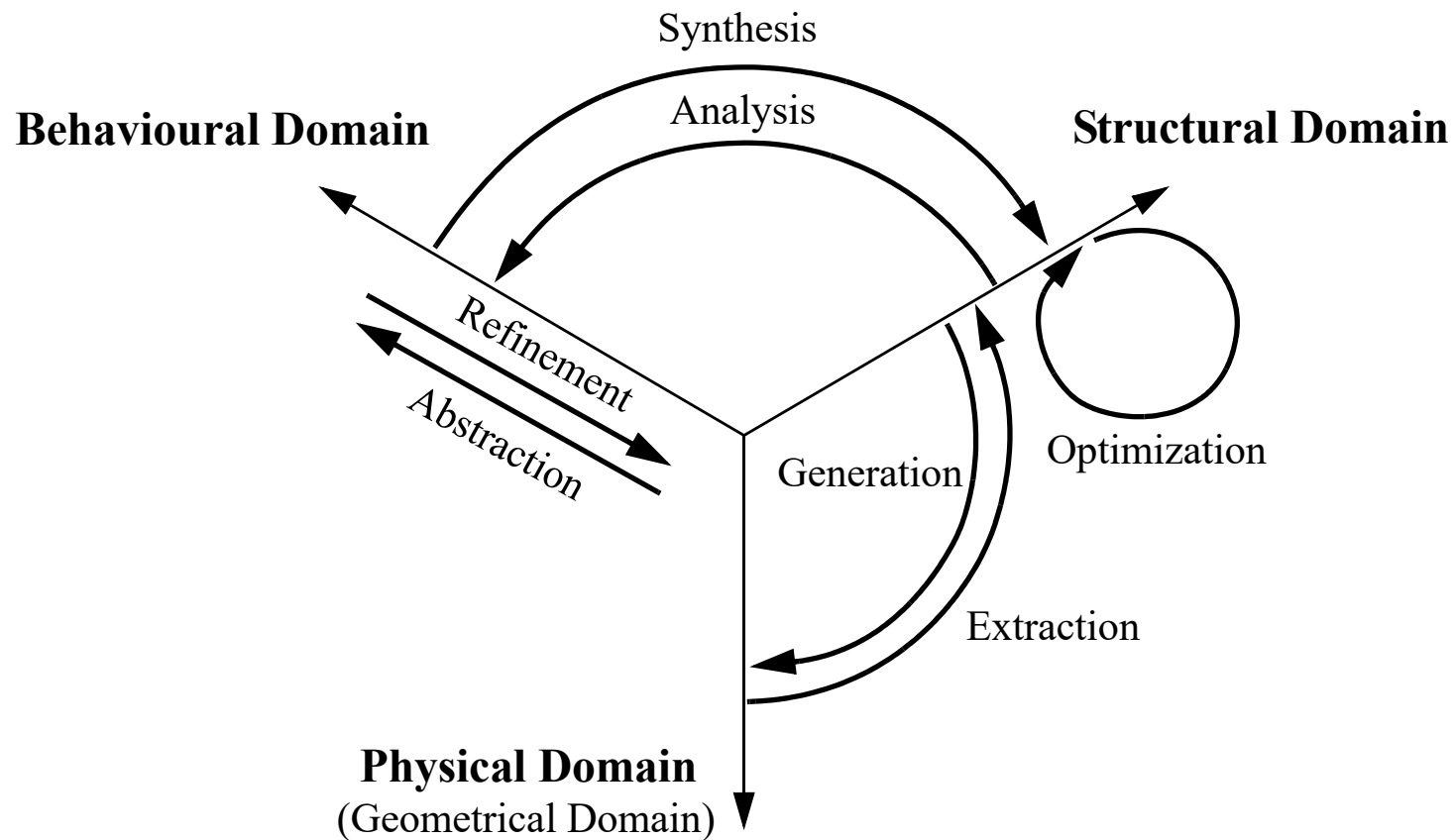
HW design flow



Y-chart

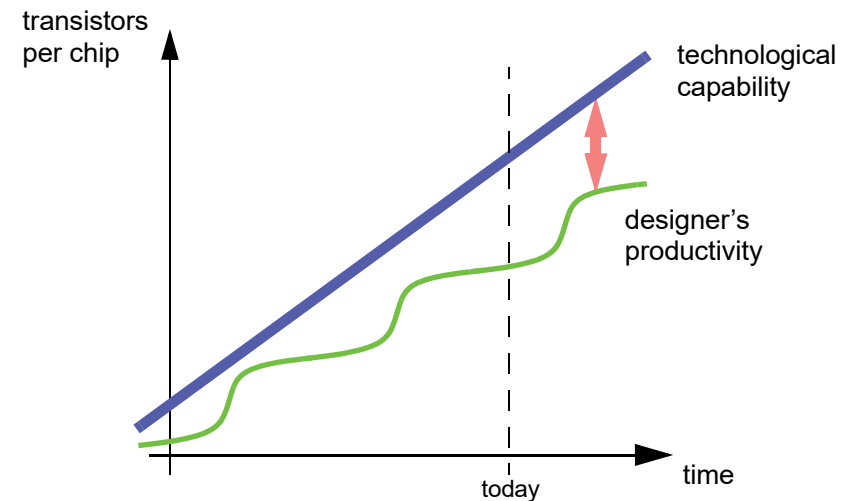


Y-transformations



Design automation history (A bit idealistic view)

- **Synthesis ~ design automation**
- **1990 – 4 K gates / year / designer**
- ...
- **2000 – synthesis (RTL→GDSII) – 91K**
- ...
- **2010 – homogeneous parallel processing (multi-core) – 1200K**
- ...
- ***Today – hw/sw co-verification, executable specification, etc.***





Market = \$\$\$

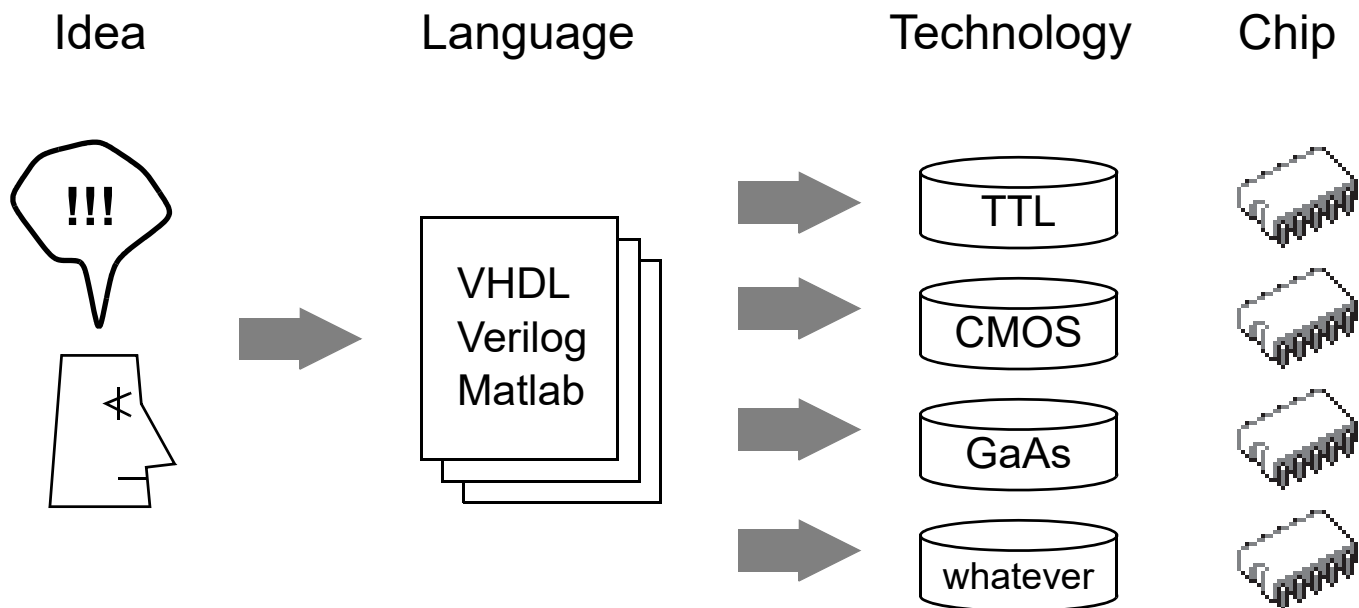
- **Design cost**
 - *design time & chips production cost*
 - *huge investments (G\$)*
 - *almost impossible to correct*
- **High cost of modifications**
 - *large production volumes are more cost effective*
 - *zero-defect is very important*
 - *following market trends is important*
- **Price is inversely 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

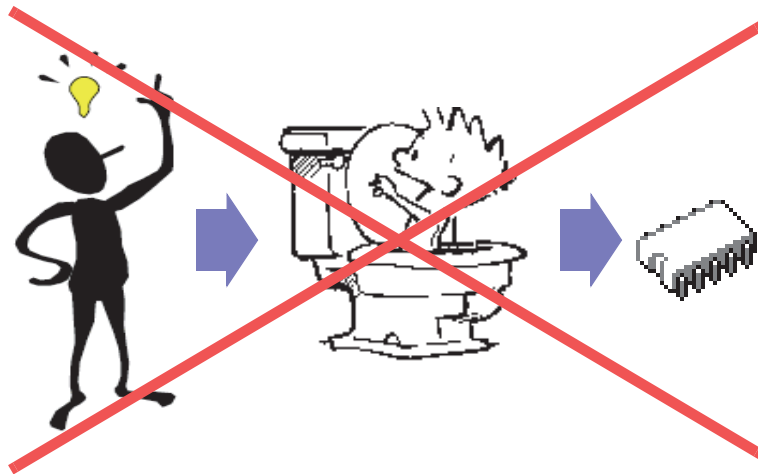
HDL – designing Systems-on-Chip (SoC) & Networks-on-Chip (NoC)



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

MYTH #1

- High level design is a single pass



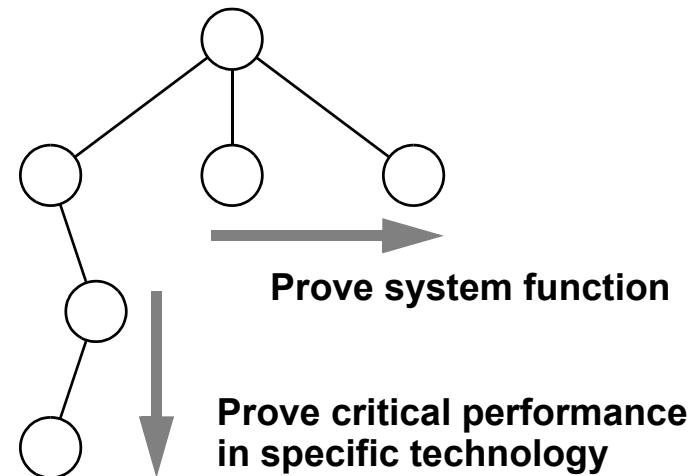
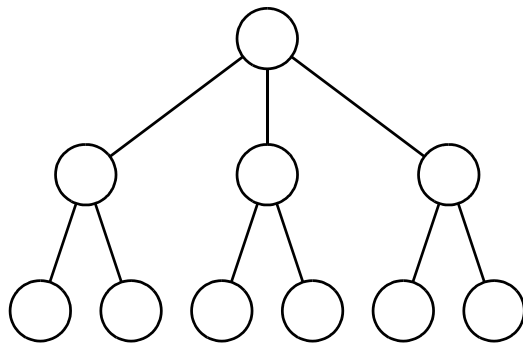
- Iterations needed
 - Functionality
 - Design goals



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

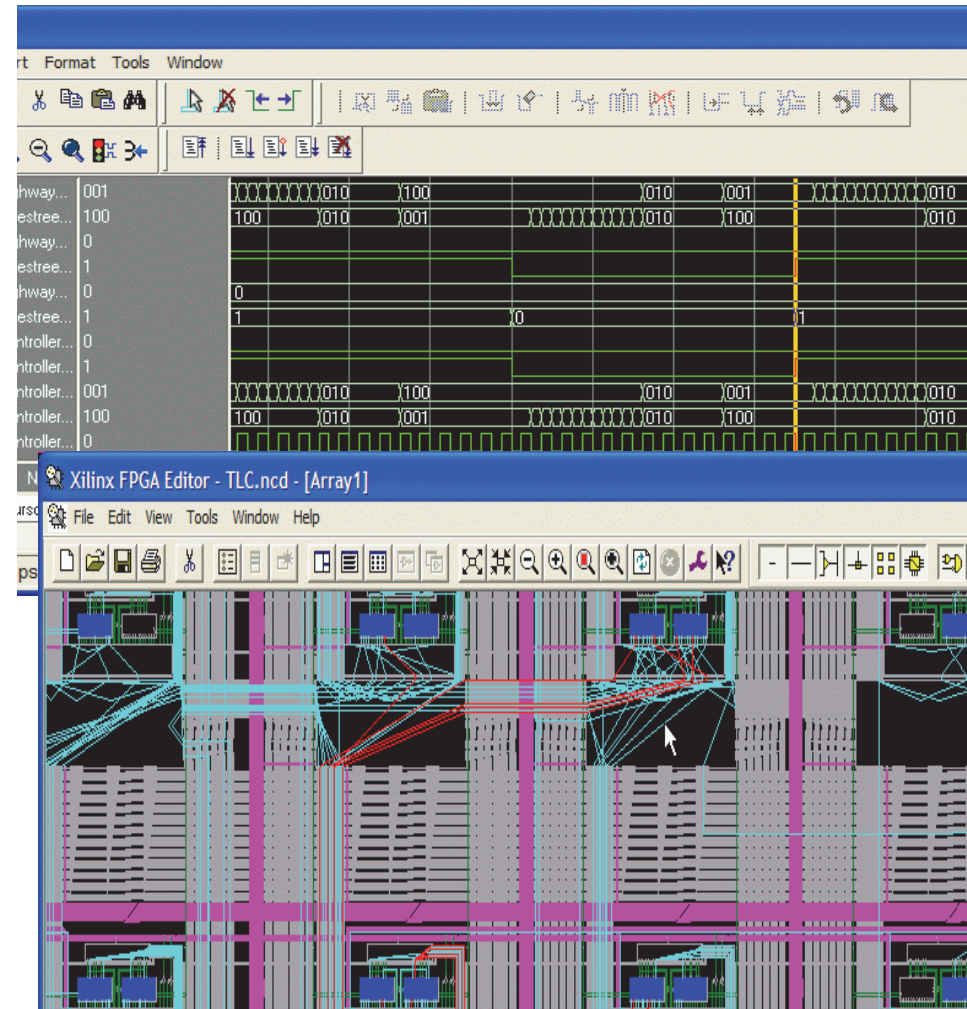
MYTH #5

- **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

Design process today

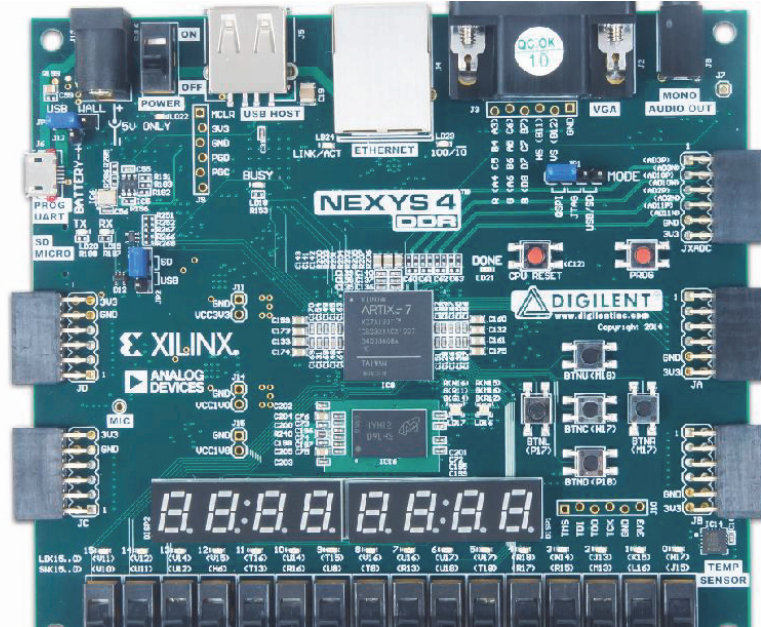
- **Hardware Description Language**

```
--
-- 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;
```

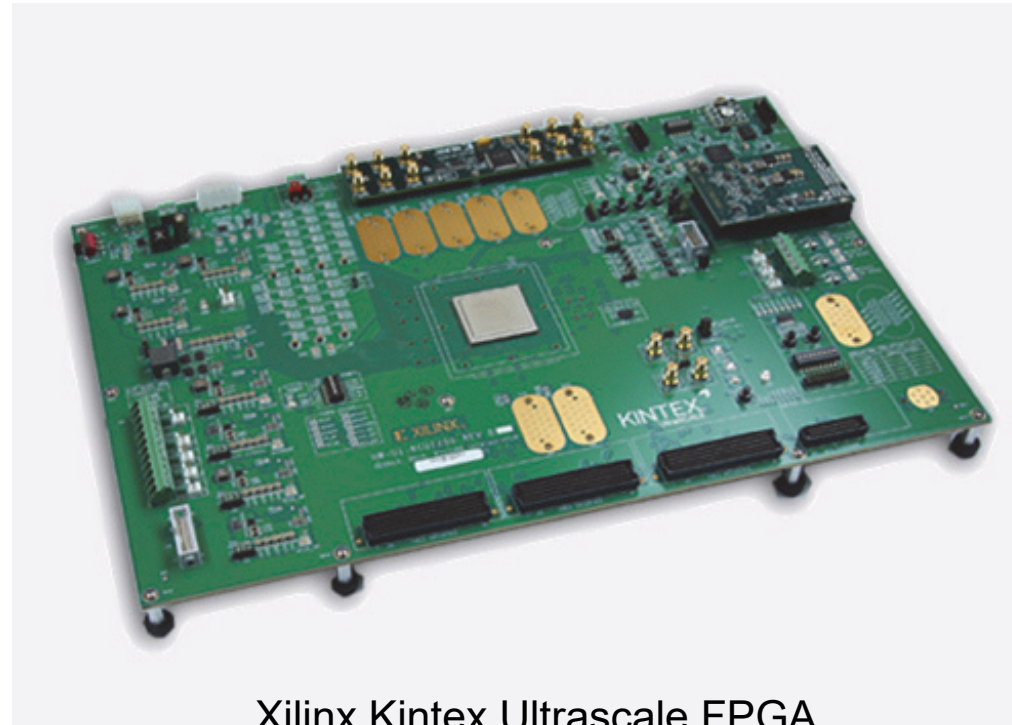


Prototyping

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

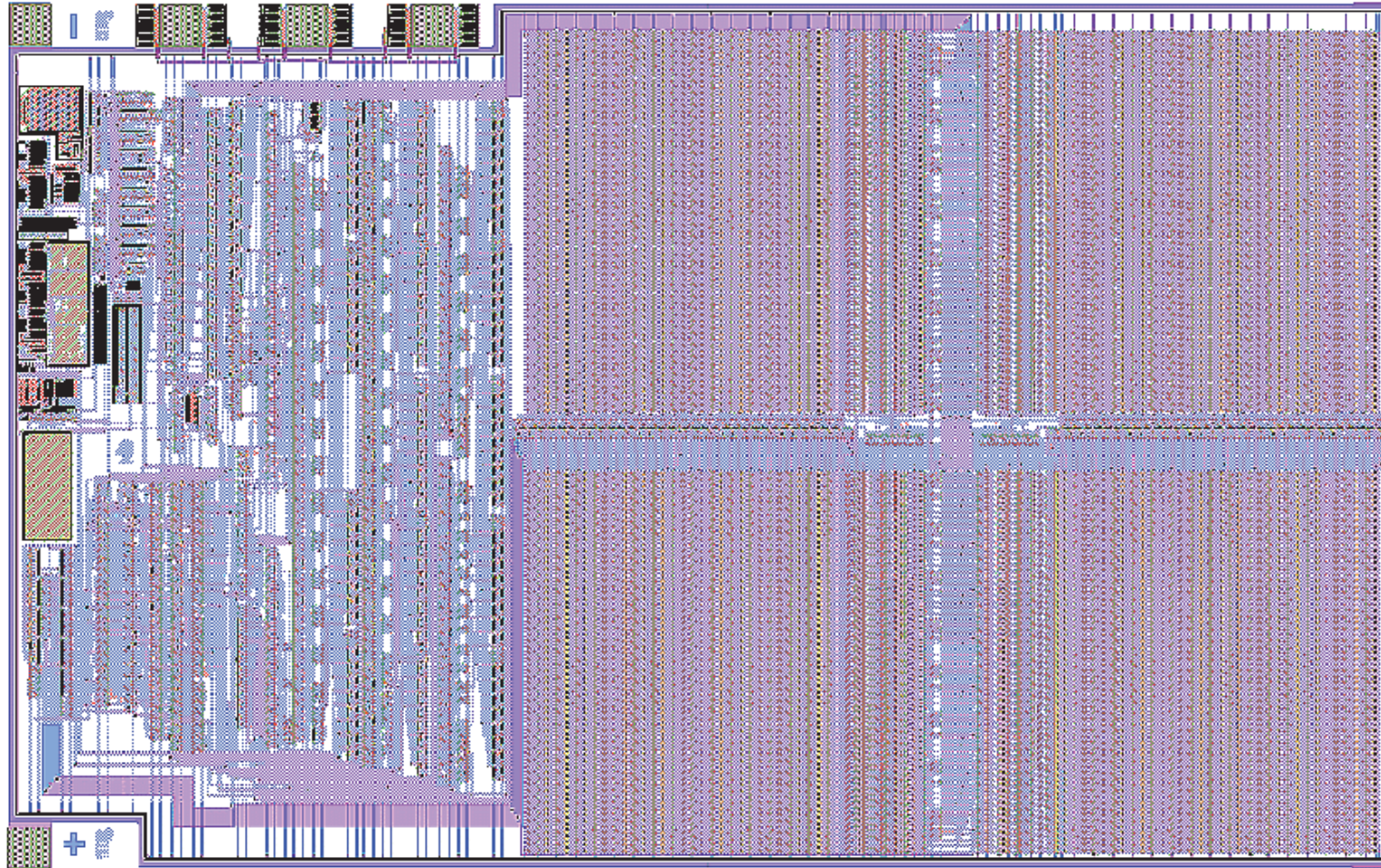


Digilent Nexys 4 DDR Artix-7 FPGA
[XC7A100T-1CSG324C; \$320]



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

Chip - the final result



Trends so far

	2000	2010	2020
memory size	2 Gbit	256 Gbit	1024 Gbit
transistors per cm ²	8·10 ⁶	160·10 ⁶	480·10 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

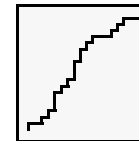
NB! These are rough approximations only!

- **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 $\rightarrow$ (10^8 m/s) $\rightarrow$
 10^{-10} s $\rightarrow$ (10%) $\rightarrow$
 1 GHz !?

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