UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN3160, IN4160 Digital system design

Introduction + HDL, PL and Design flow

Yngve Hafting

Overview

- General information
  - Course management
  - Schedule
  - Course Goals
  - Curriculum
  - Lab assignments
  - Who are we
- Motivation
  - Why Digital Design?
  - Why HDL?
- Intro to programmable Logic
  - What is programmable logic?
  - Why choose programmable logic?
- Design Flow for digital designs
- Intro to our hardware…:
  - Our hardware: Zedboard
    - Architecture
    - Documentation
  - «Our» HDL: VHDL
- Assignments and suggested reading for this week



Course Management

- Lecturers:
  - **Roar** Skogstrøm (II’er IFI, Kongsberg Defence Communications)
  - **Alexander** Wold (II’er IFI, FFI)
  - **Yngve** Hafting (Universitetslektor IFI/ROBIN)
- Lab supervisors / teachers:
  - **Arne** Martin Dybendal Foldvik (Student)
  - **Georg** Magneshaugen (Student)
  - Karl **Jørgen** Giercksky Russnes (Student)
  - **Sander** Elias Magnussen Helgesen (Student)
  - Seyed **Mojtaba** Karbasi (PhD)

Lectures

Tuesday 14:15 -16:00, Zoom (and OJD, C «3437»)

Thursday 12:15-14:00, Zoom (and OJD, C «3437»)

Lab

LISP (2428): No group education, lab supervision poll next slide

https://www.mn.uio.no/ifi/om/finn-fram/apningstider/

Monday	Tuesday	Wednesday	Thursday	Friday
			Lecture 12-14	
	Lecture 14-16			

Web

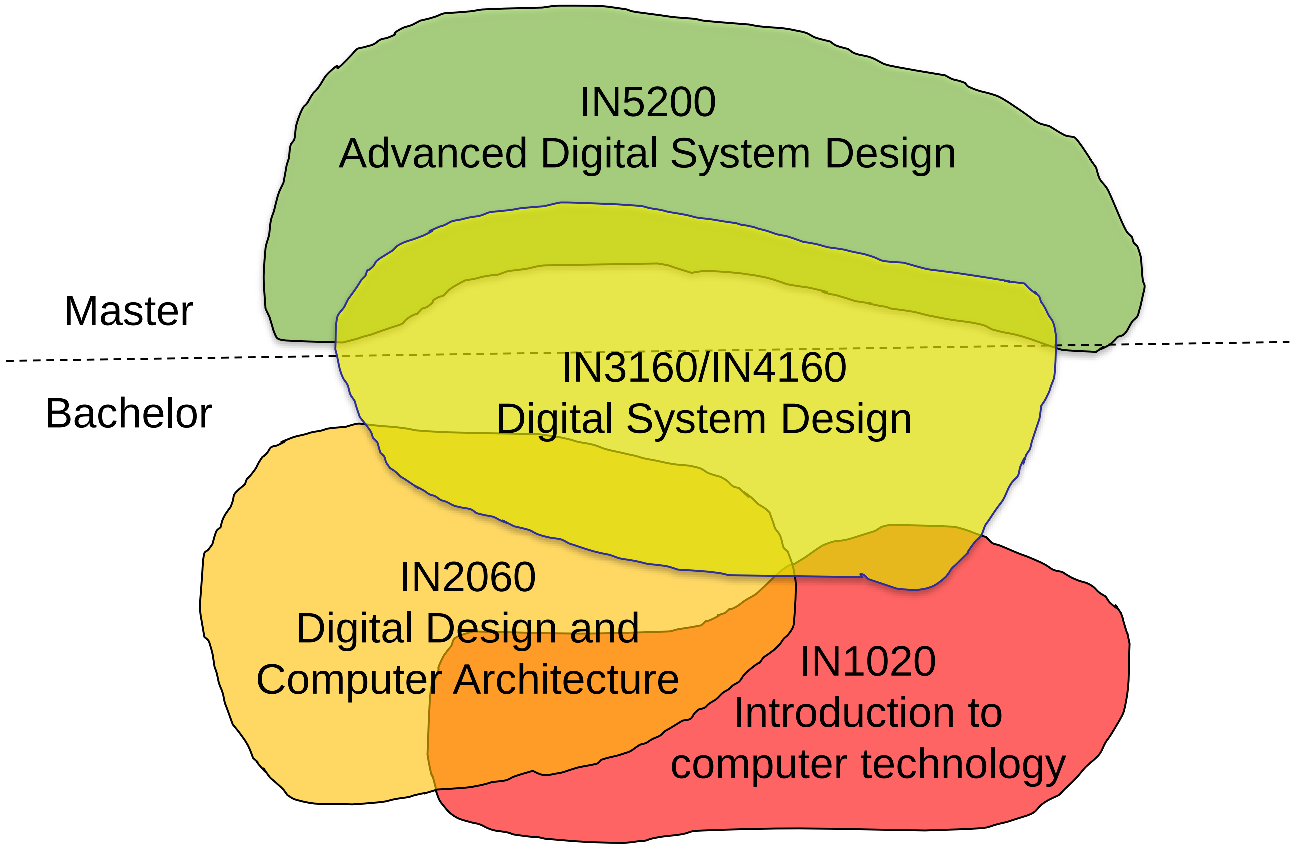
http://www.uio.no/studier/emner/matnat/ifi/IN3160/

(covers also INF4160)

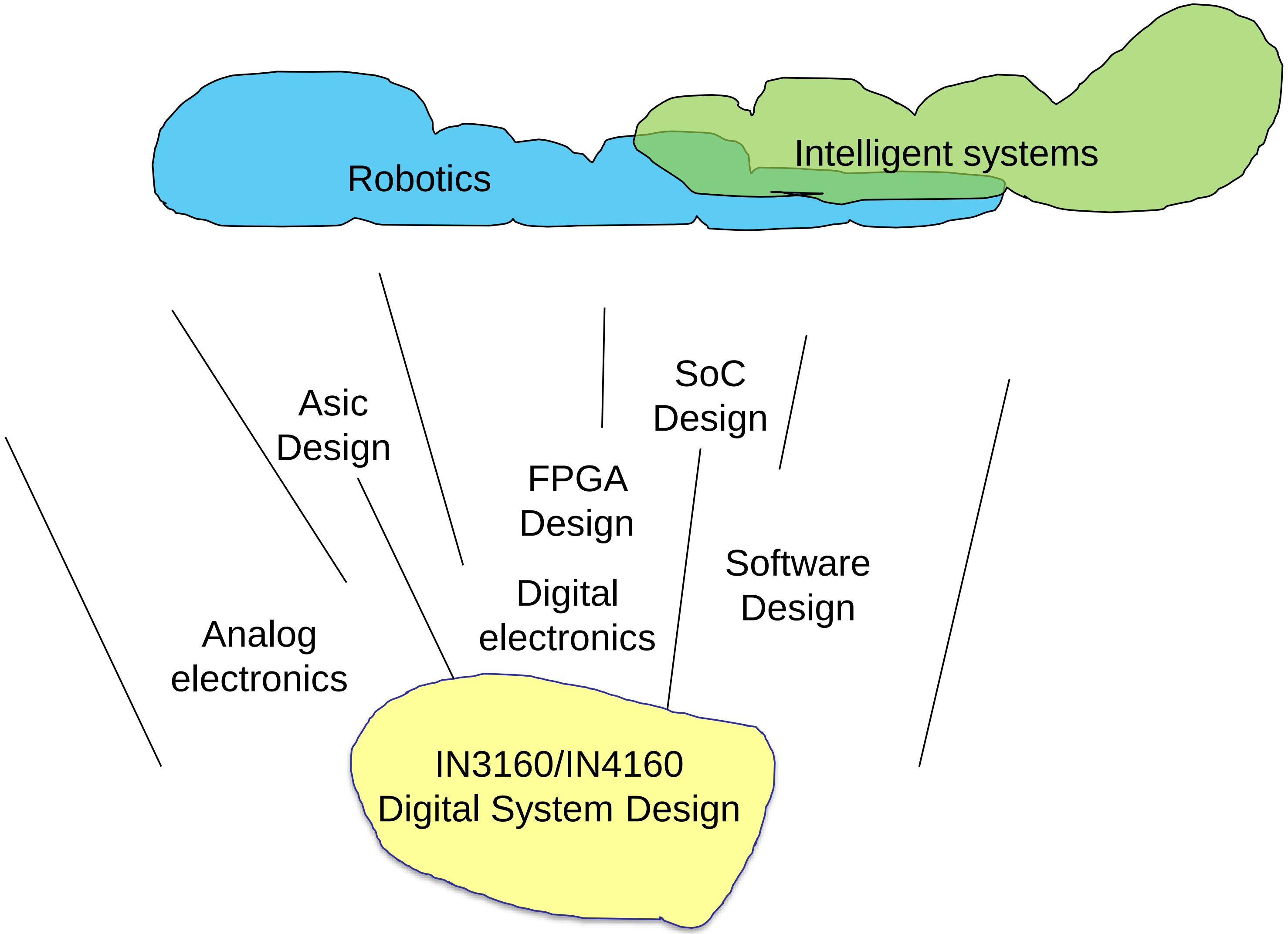
Where do we stand + lab supervision poll?

- www.menti.com
- Code 8115 6823

Study program connections



Relevancy


Robotics
Intelligent systems
Asic Design
SoC Design
FPGA Design
Digital electronics
Software Design
Analog electronics
IN3160/IN4160
Digital System Design

Course Goals and Learning Outcome

https://www.uio.no/studier/emner/matnat/ifi/IN3160/index-eng.html

- In this course you will learn about the design of advanced digital systems.
- This includes programmable logic circuits, a hardware design language and system-on-chip design (processor, memory and logic on a chip).
- Lab assignments provide practical experience in how real design can be made.

- After completion of the course you´ll:...

... IN3160 vs IN4160 ...

IN3160

- After completion of the course you´ll:
  - understand important principles for design and testing of digital systems
  - understand the relationship between behavior and different construction criteria
  - be able to describe advanced digital systems at different levels of detail
  - be able to perform simulation and synthesis of digital systems.

IN4160

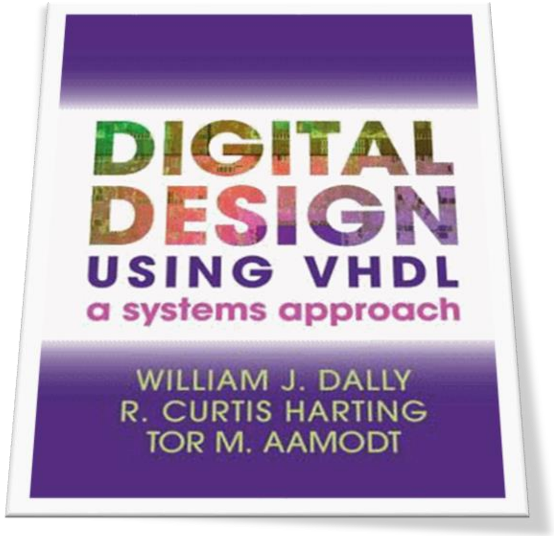
- After completion of the course you´ll:
  - understand important principles for design and testing of digital systems
  - understand the relationship between behavior and different construction criteria
  - be able to describe advanced digital systems at different levels of detail
  - be able to perform **advanced** simulation and synthesis of digital systems
  - **be able to perform advanced implementation and analysis techniques**

NOTE: these are MINIMUM requirements for passing an exam.

- You will be given the same opportunities to learn, and the curriculum is the same.
- *Grading will be stricter for IN4160 due to added minimum requirements*
- Otherwise, this course will be held as one.

Syllabus

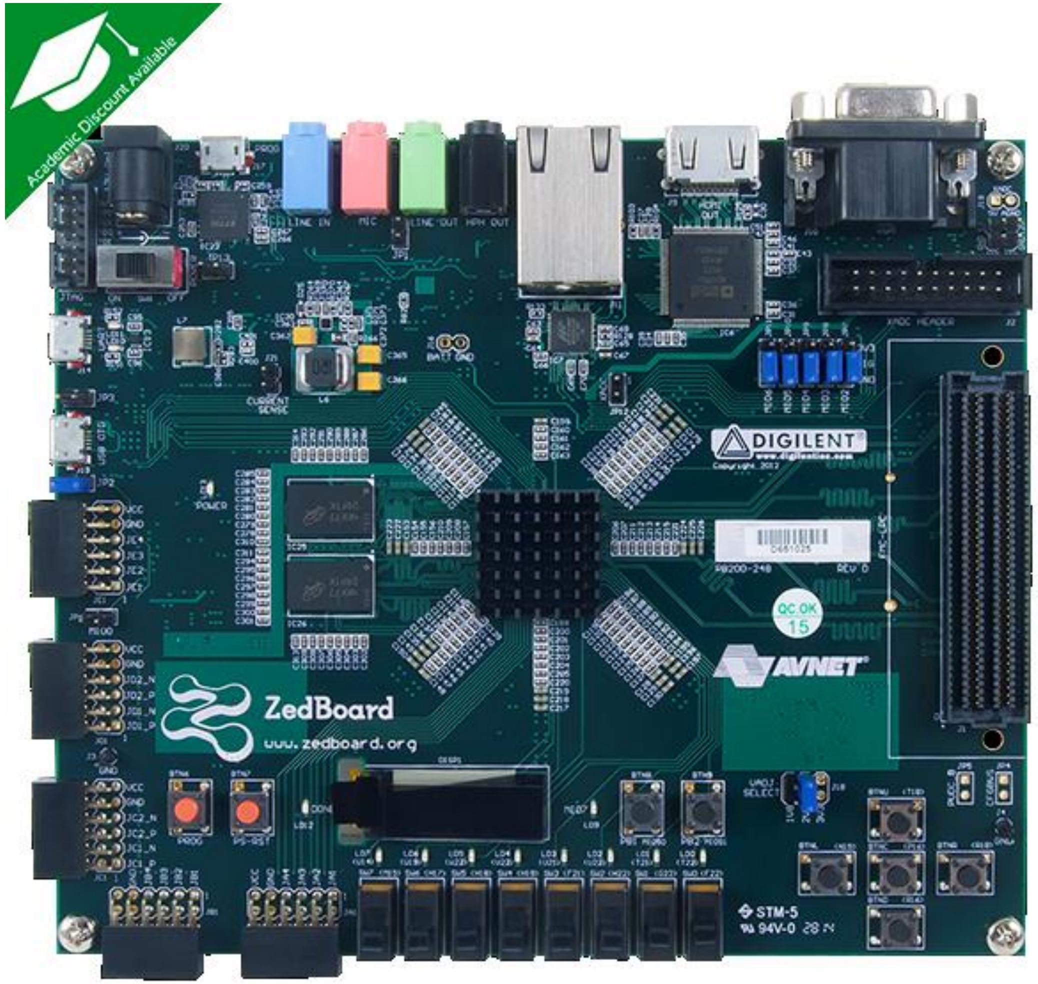
- Dally, William J. - Harting, R. Curtis - Aamodt, Tor M.
  Digital Design Using VHDL A Systems Approach
  Cambridge University Press 2016
  ISBN9781107098862



- Lectures and lecture slides
- Mandatory assignments
- Handouts
  - Cookbook (will be available digitally)
  - Articles

Compulsory lab assignments

- There are 10 compulsory lab assignments.
- All assignments must be completed to take the exam.
  - *Lab workload increases* through the semester
- Lectures are prerequisite for some assignments
  - Lectures most intensive in the beginning
- The lab assignments utilises the digilent Zedboard, featuring a Xilinx Zynq 7020 device that includes both a hardcoded ARM processor and FPGA fabric.
- By the end of this course you will design a system, using both processor and FPGA fabric, that will both regulate, read and display the speed of an electric motor connected to the board.



https://store.digilentinc.com/zedboard-zynq-7000-arm-fpga-soc-development-board/

General information

Vortex (Course web)	Canvas	Discourse
Mostly open for all • General course information (Exam dates, etc) • Lecture schedule with lecture notes and screencast • Announcements • Syllabus	**Enrolled students** • Mandatory assignments • Feedback on assignments	**Enrolled students** • Discussions • Questions

https://astro-discourse.uio.no/

- Lab starts now! (Expect to use much time on the last few assignments)
  - **Assignments are available in Canvas!**
  - Assignments are individual.
  - There will be one assignment using peer review only.
  - Some assignments may require that you show your setup to the lab supervisor.
    - ***Last year showed us labs can be done entirely remote, but on-site is strongly adviced.***
- **LISP (2428) is the LAB.**
  - Both hardware and software will be available in LISP.
    - 4 boards with camera will be available online for those in quarantine/ isolation / specieal needs.
  -
- Questions..?

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN3160
Introduksjon, HDL og PL

Hardware Description Language & Programmable Logic

Yngve Hafting

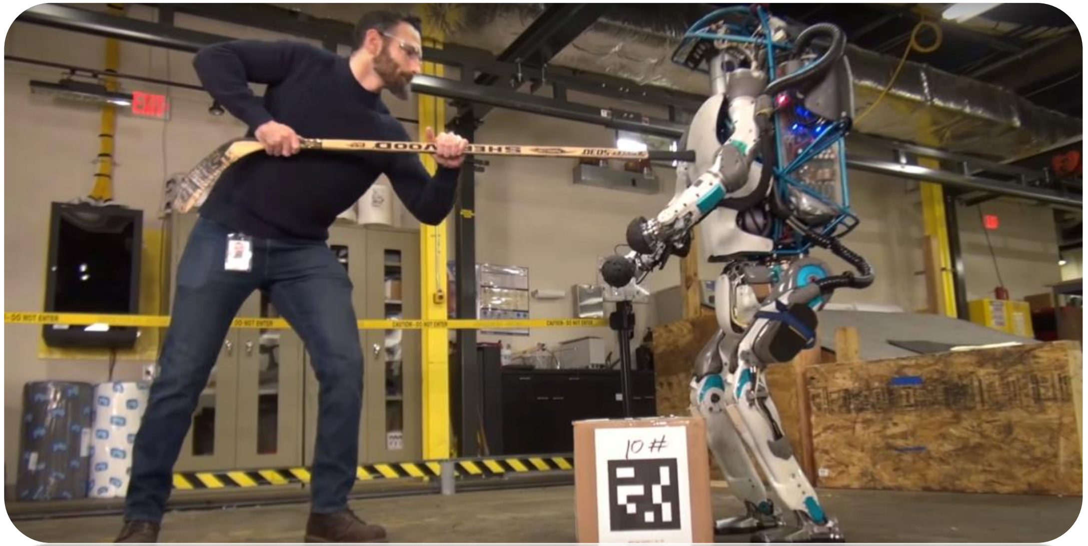
UiO : **Institutt for informatikk**

Det matematisk-naturvitenskapelige fakultet

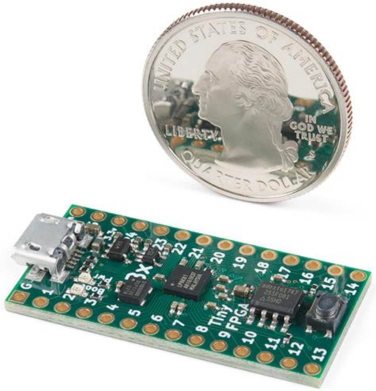
http://www.aes-eu.com/10gbe-fpga-nic.php

http://www.moisund.com/2014/04/08/opning-av-ny-basestasjon-i-aseral/

Wikipedia: U.S. Navy photo by Photographer's Mate Airman Marvin E. Thompson Jr.



https://www.youtube.com/watch?v=rVlhMGQgDkY

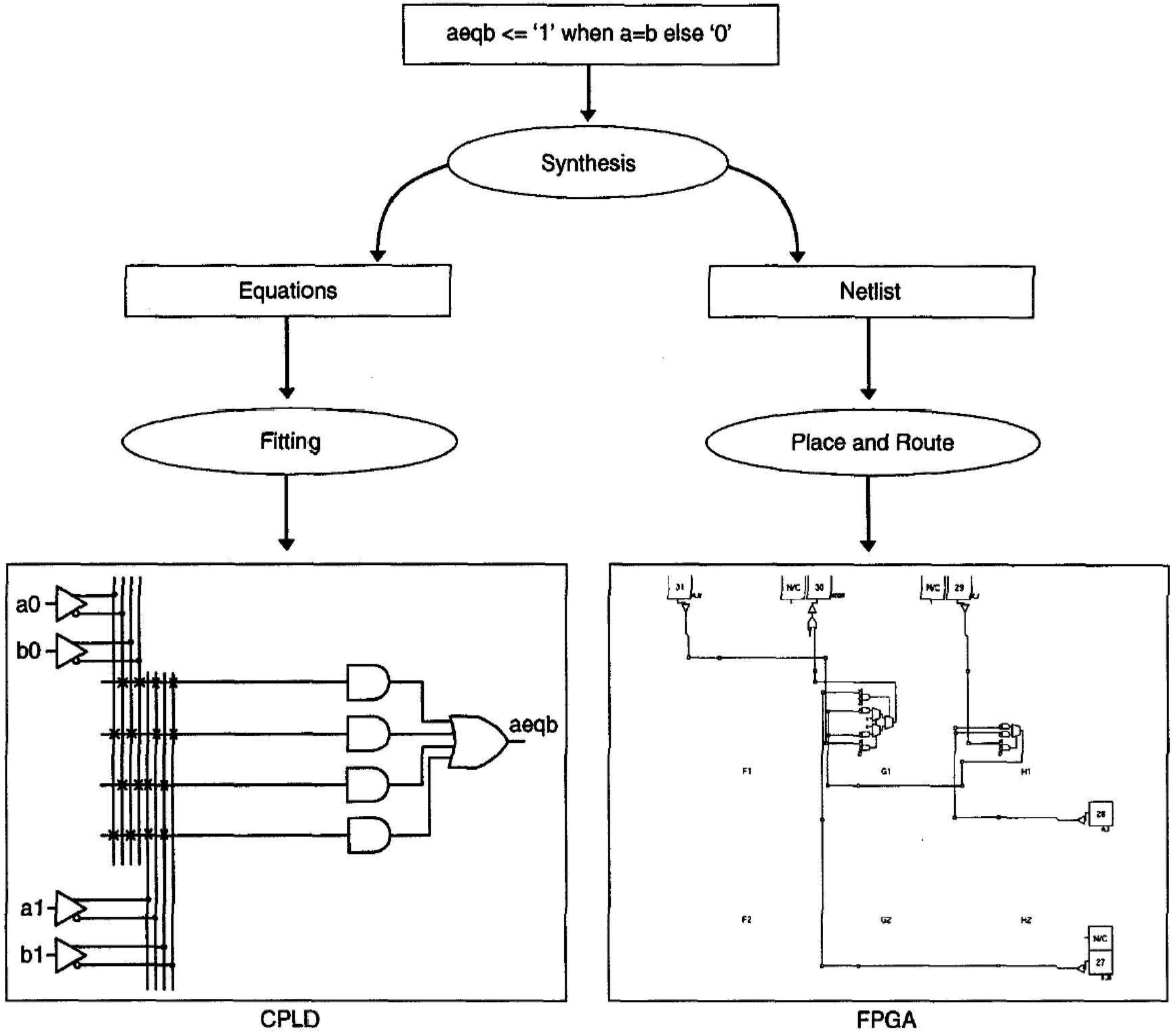


https://www.sparkfun.com/products/14829

https://www.komplett.no/product/11257/pc-nettbrett/komplett-pc/komplett-gamer-xtreme/komplett-gamer-xtreme-i250?offerId=KOMPLETT-310-11257#

Why HDL?

- Syntesis to design



Why HDL?

- Technology independent code
- Different abstraction layers

Netlist:	Boolean equations:

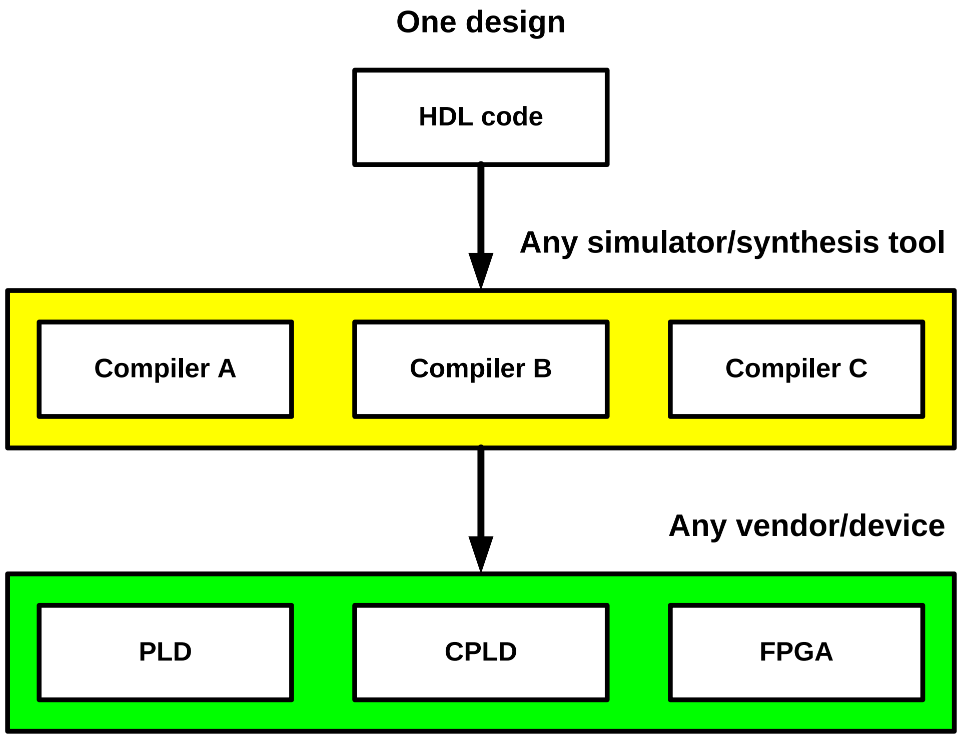
`U1: xor2 port map(a(0), b(0), x(0));`
`U2: xor2 port map(a(1), b(1), x(1));`
`U3: nor2 port map(x(0), x(1), aeqb);`
	`aeqb <= (a(0) xor b(0))`
	`nor (a(1) xor b(1));`

Concurrent statements:	**Sequential statements:**

`aeqb <= '1' when a=b else '0';`	`if a=b then`
	`aeqb <= '1';`
	`else`
	`aeqb <= '0';`
	`end if;`

Why HDL?

- Portability
- IEEE Standards
  - VHDL/System Verilog are both IEEE (Institute of Electrical and Electronics Engineers) standards
    - VHDL - IEEE 1076
    - System Verilog - IEEE 1364



Why HDL?

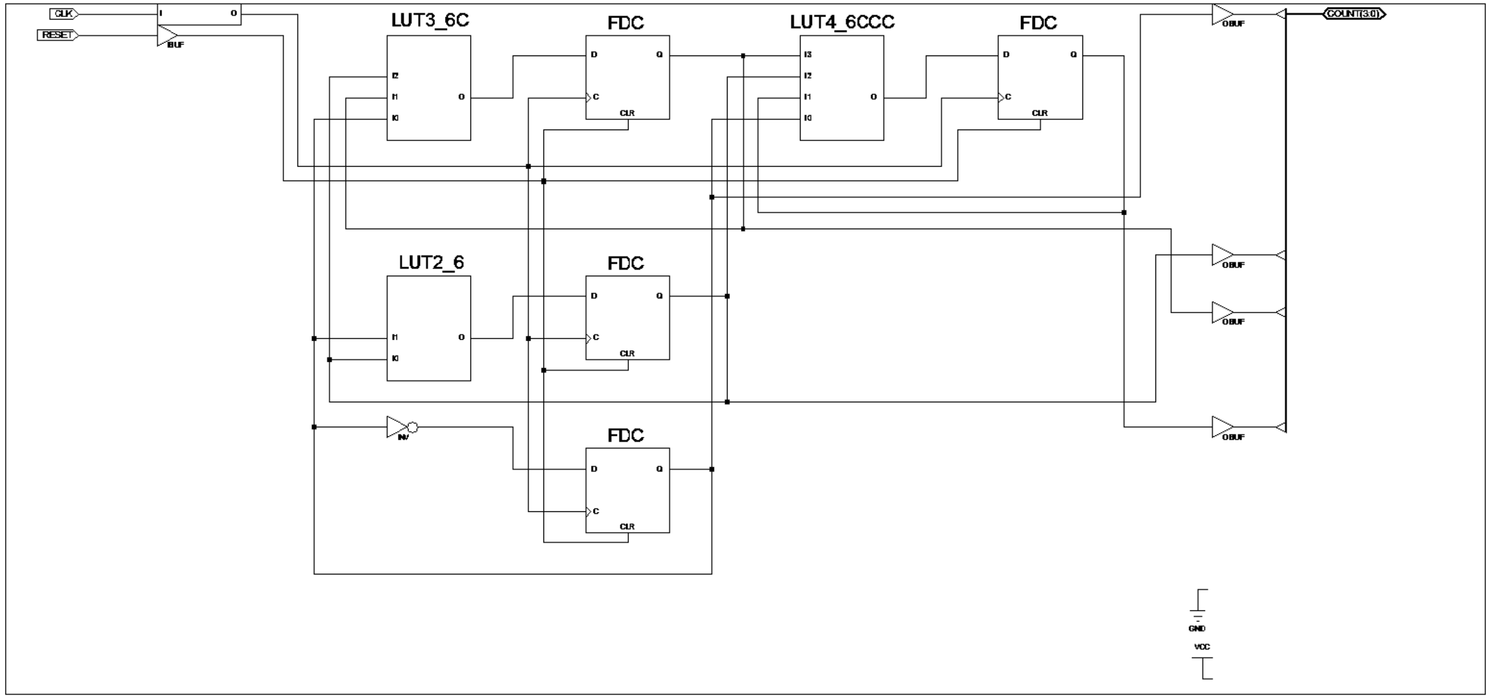
- Simple mainteenance/expansion of a design

```
COUNT : inout std_logic_vector(7 downto 0); -- Count value
---
COUNTER :
  process (RESET,CLK)
  begin
    if(RESET = '1') then
      COUNT <= (others => '0');
    elsif rising_edge(CLK) then
      COUNT <= COUNT + 1;
    end if;
  end process COUNTER;
```

Why HDL?

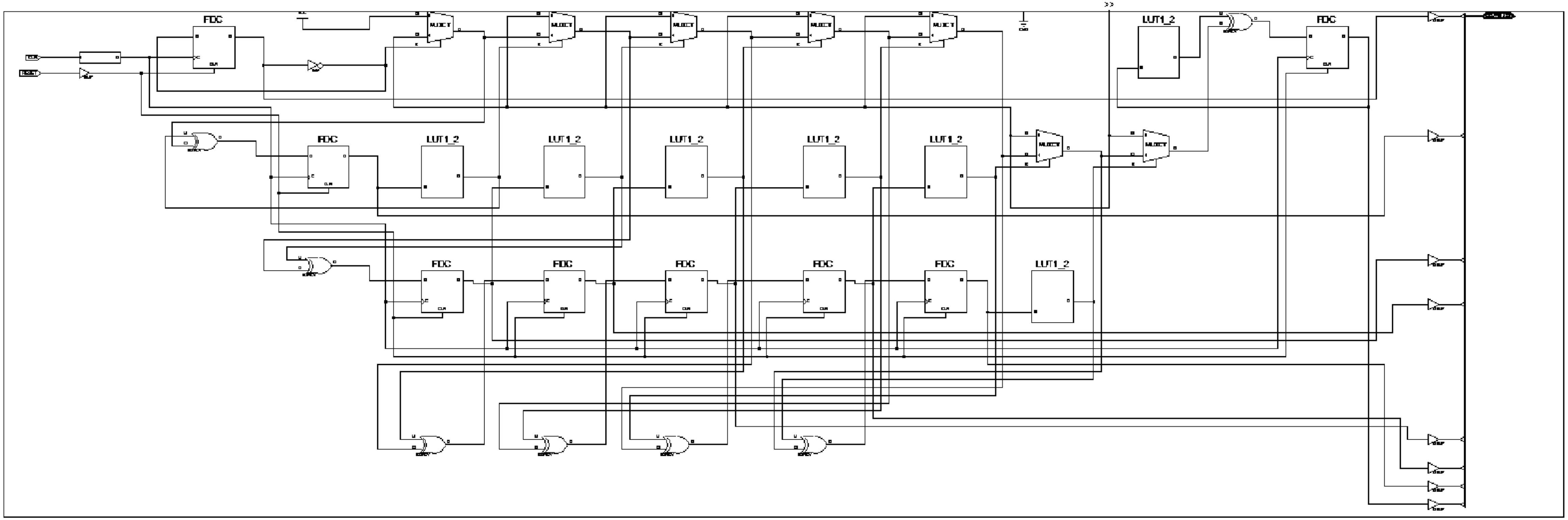
4 bit counter:

```
COUNT : inout std_logic_vector(3 downto 0); -- Count value
```



8 bit …

```
COUNT : inout std_logic_vector(7 downto 0); -- Count value
```

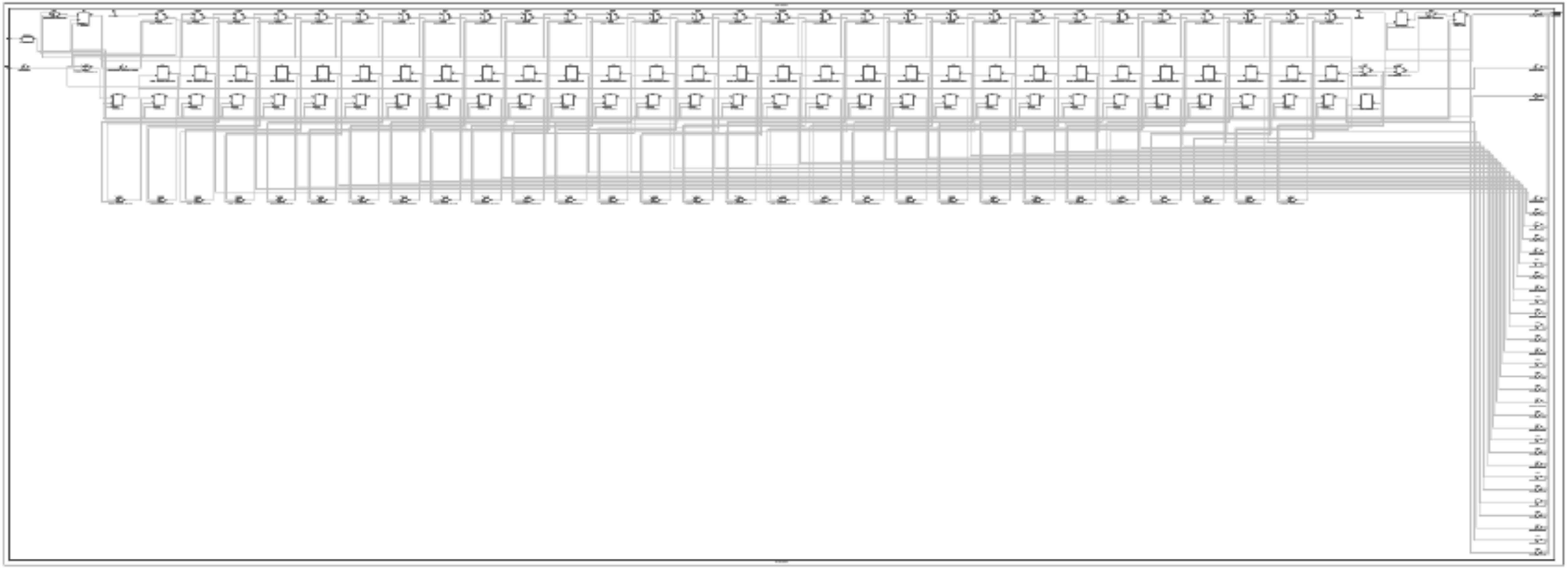


16 bit …

```
COUNT : inout std_logic_vector(15 downto 0); -- Count value
```

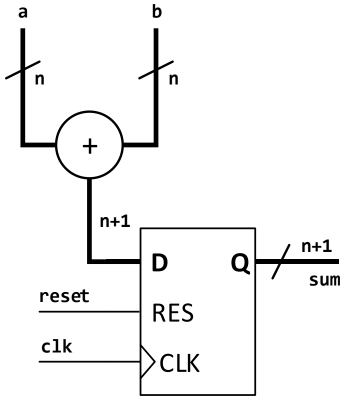
32 bit...

```
COUNT : inout std logic vector(31 downto 0); -- Count value
```



HDL vs software

```
process(reset, clk)
begin
  if (reset = ‘1’) then
    sum <= '0';
  elsif rising_edge(clk) then
    sum <= a + b;
  end if;
end process;
```



HDL «Hardware description language»	Software programs
Defines the logic function of a circuit	**Defines the sequence of instructions and which data shall be used** for one or more processors or processor cores
CAD tools *syntetizises* designs to enable realization using physical gates.	***A compiler translates program code to machine code instructions*** that the processor can read sequentially from memory
Implemented using programmable logic (*PL,* FPGA, CPLD, PLD, PAL, PLA, …) or *ASICs* (application specific circuits) (“ASICs”, processors, ..-chips,.. etc.)	**Is stored in computer memory**
Verilog (SystemVerilog) **VHDL (VHDL 2008)** (System C m. fl.)	C, C++, C#, Python, Java, assemblere (ARM, MIPS, x86, …) Fortran, LISP, Simula, Pascal, osv…

```
int sum(int a, int b){
  int s;
  s = a + b;
  return s;
}
```

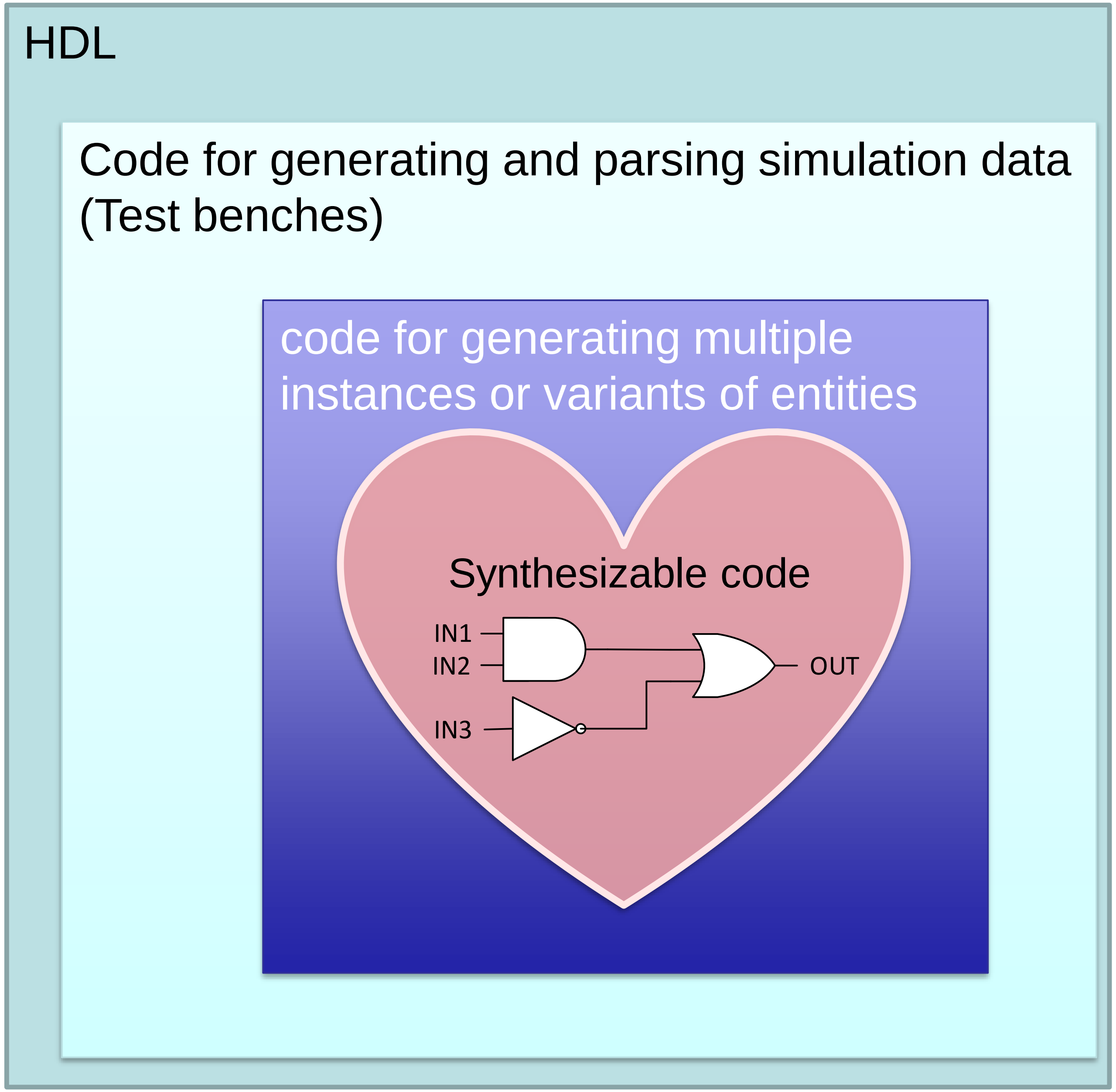
```
MOV R5, #0        ;set base adr
LDR R7, [R5, #8] ;load reg R7
LDR R8, [R5, #12];load reg R8
ADD R0, R7, R8   ;R0=R7+R8
STR R0, [R5, #16];store R0
```

```
01001100 10011001 00100100 01001010
11001100 10111001 01100100 11110110
01001100 10011001 00111100 11101010
01001100 10011001 00100100 01011010
…
```

(binary code is random, for illustration only)

HDL

- VHDL = VHSIC HDL:
  - Very High Speed Integrated Circuit **Hardware Description Language**
  - The purpose is to generate hardware, and verify it through simulation.
  - **Synthesizable (realizable) code work concurrently (in parallel)**.
  - Code for simulation include things such as file I/O which cannot be synthesized.
  - Testbenches can and will use some synthesizable elements, but will in general look more like other sequential languages, and use sequential statements.
    <u>This may be confusing at times...</u>
  - VHDL does come with several libraries.

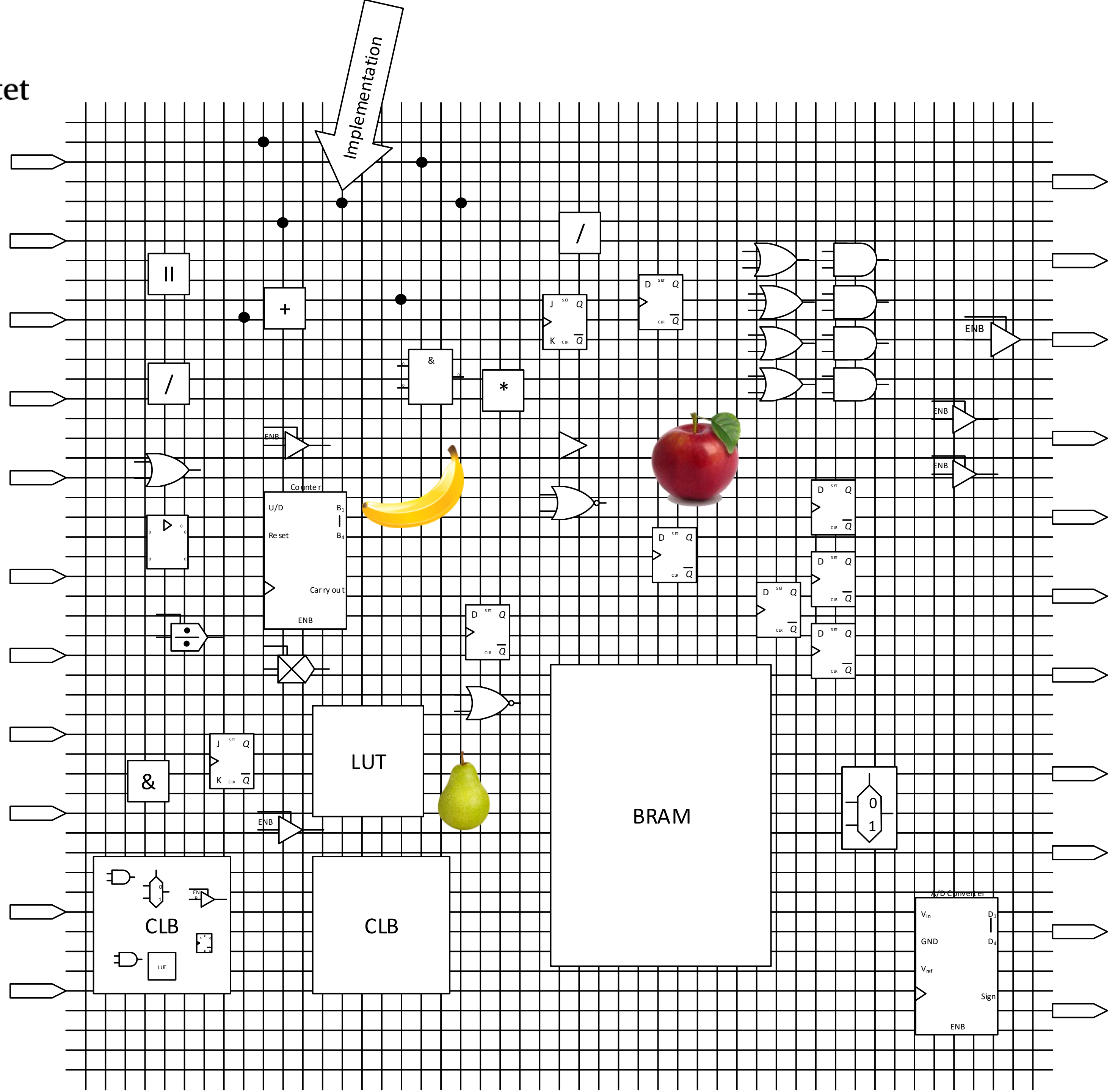


What is PL ? (Programmable Logic)

- PL = FPGA, CPLD, PLA…
  (Field Programmable Gate Array,
  Complex Programmable logic Device)

- PL vs processor
  (FPGA vs CPU, MCU) ?

- PL vs ASIC (Application Specific Integrated circuit)?



When or why choose programmable logic?

- (Verify behavior of ASIC)
- Prototyping flexibility
  - Lots of multi purpose IO
  - Reprogrammable
- Small batch production
- Parallellism
- Custom / fast
- Runtime reconfigurability
- …

When to avoid programmable logic?

- When low component cost is extremely important.
- When dedicated HW is well suited.
- When extreme speed is required => ASIC
- …

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

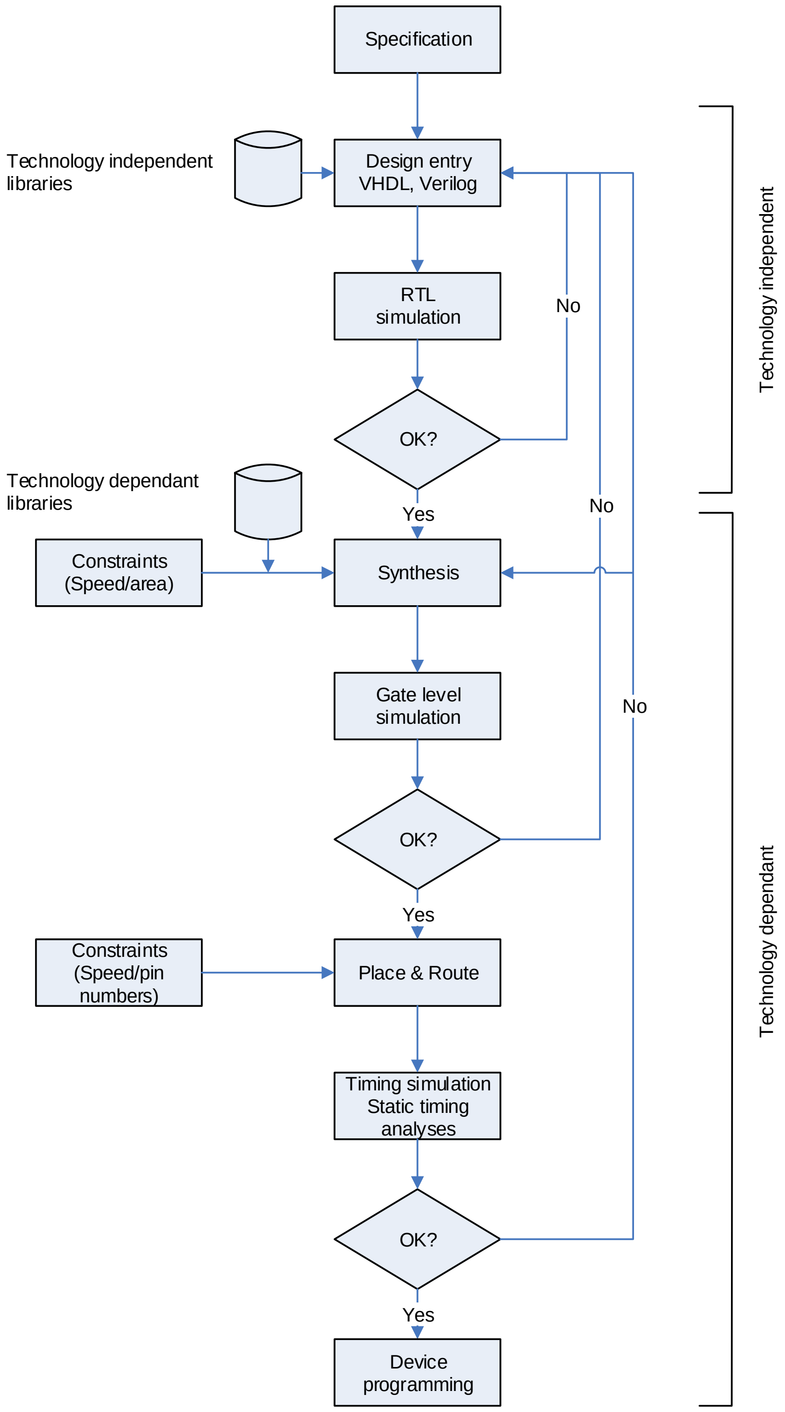
IN3160

Digital Design Flow

Yngve Hafting

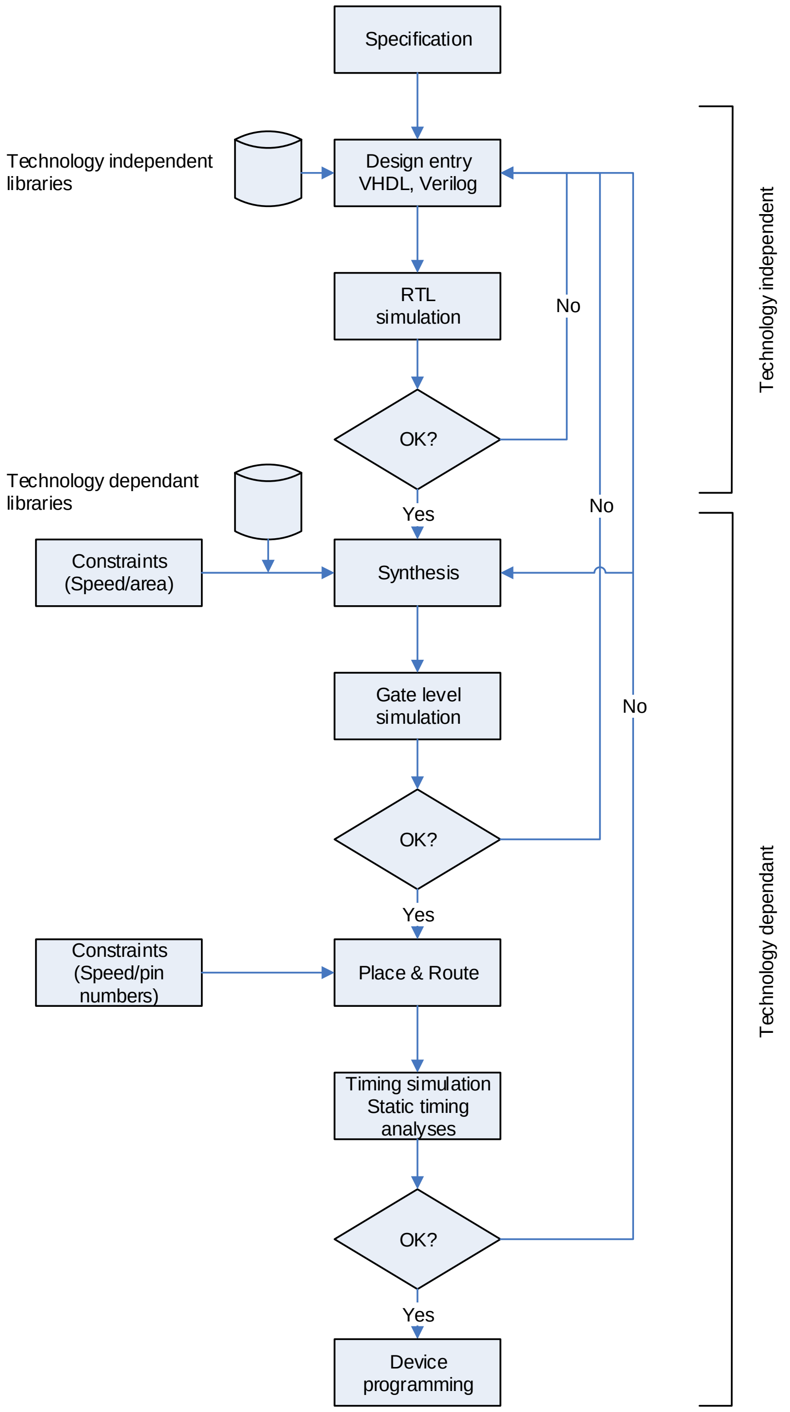
Overview

– Digital design tools.
– Specification
– Design entry, synthesis and PAR
– Timing analysis
– Timing simulation
– Testing



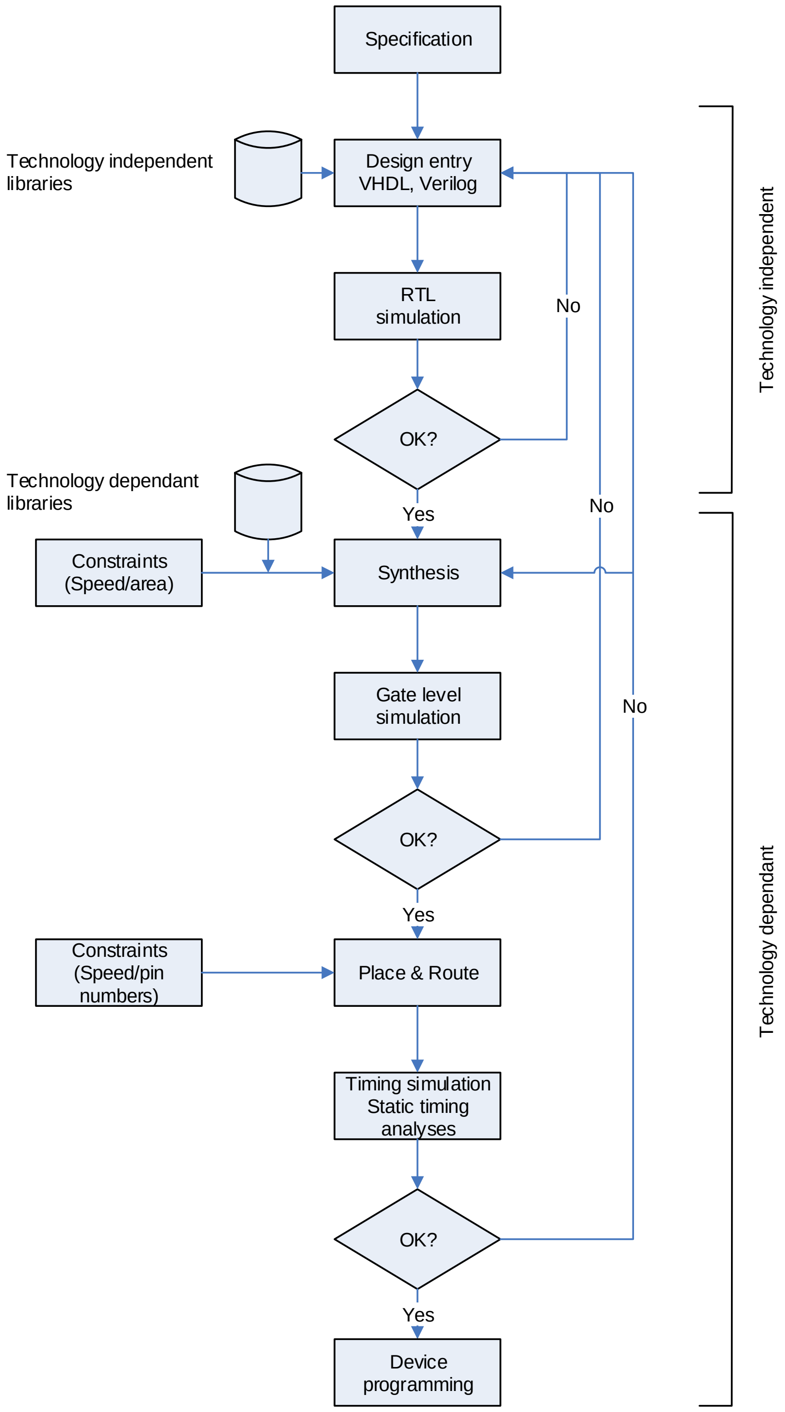
Digital Design tools…

- Design entry:
  - Use your favourite HDL text editor (Notepad++, Emacs, Vivado or Questa).
- Simulation (RTL, Gate Level, Timing)
  - Here: Typically using Questa (=Modelsim)
- Synthesis, Implementation, Programming
  - Vendor specific tools,
    - Here: Vivado by Xilinx
  - Also possible: Digilent tools for programming.



Digital Design Flow: Specification

1. Define the problem
2. Draw a functional diagram
   – block diagram with major components and connections
3. Identify IO requirements
4. Identify necessary interface circuits
5. Decide on HDL (VHDL, Verilog, System C,…)
6. *Draw a program flowchart (ASM diagram)*
   – Defines how the design shall work logically.
   – By hand or using tools such as:
     - *Visio, Draw.io, Lucid chart, etc.*

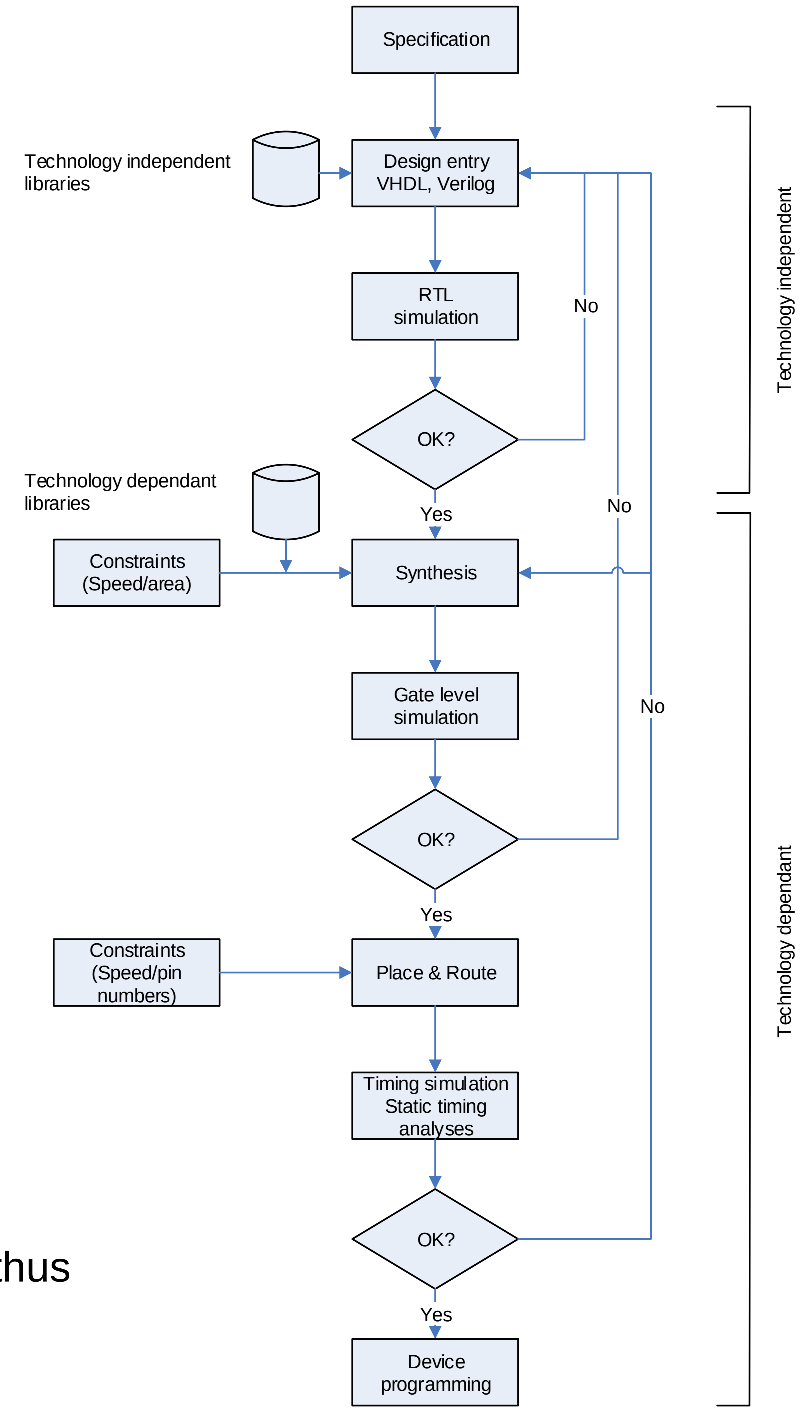


Design entry, synthesis and PAR

- RTL = Register Transfer Level
  - RTL does not use specific gates or technology
  - Designs are *mostly* done in RTL
  - RTL simulation can be used to verify logic function.

- Gate level synthesis
  - Technology specific gates are selected for all components in the design.
    - Typically a synthesizer will pick gates specific for the (FPGA) chip family we use.
  - Once we have a gate level design we can
    - calculate gate-, but not propagation delays
    - Simulate using gate delays.

- Place and route
  - After synthesis gates can be placed within a specific (FPGA) chip.
  - When place and route is performed propagation delays may also be simulated thus
  - We can do all timing simulation, including propagation delays.



Static timing analysis

- Performed by EDA tools on synthesized or routed designs
- Will attempt to
  - find critical path(s) and
  - check if timing requirements (constraints) can be met.

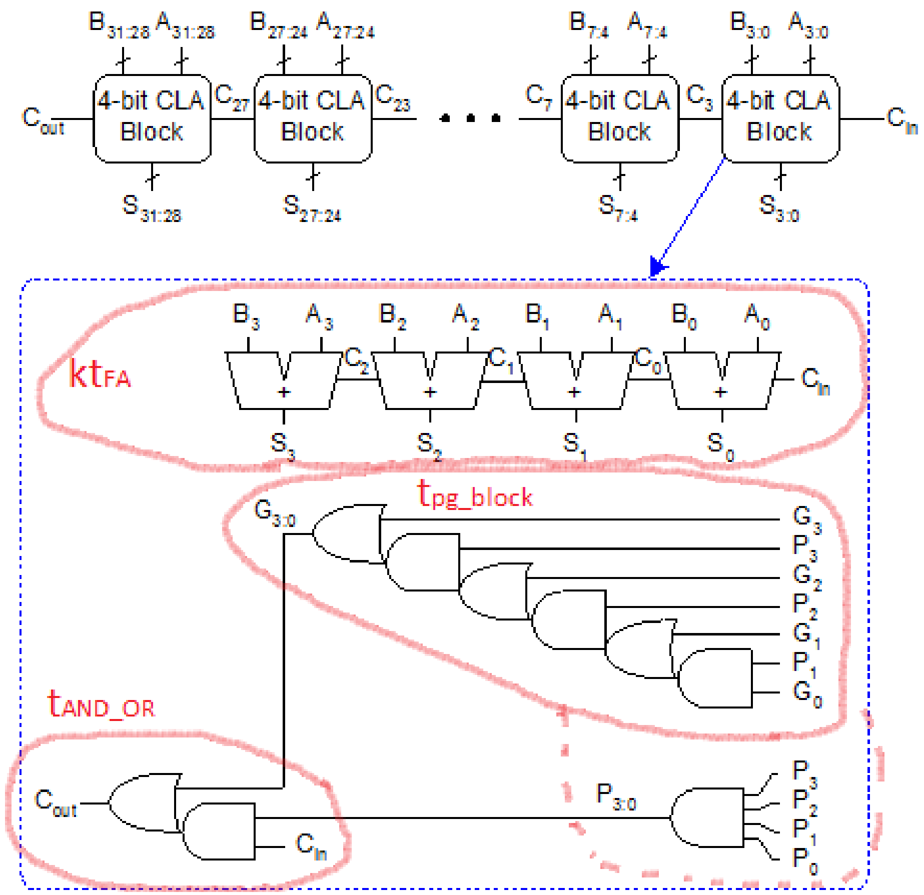
IN2060: Carry-Lookahead Adder Delay

For *N*-bit CLA with *k*-bit blocks:

$$t_{CLA} = t_{pg} + t_{pg\_block} + (N/k - 1)t_{AND\_OR} + kt_{FA}$$

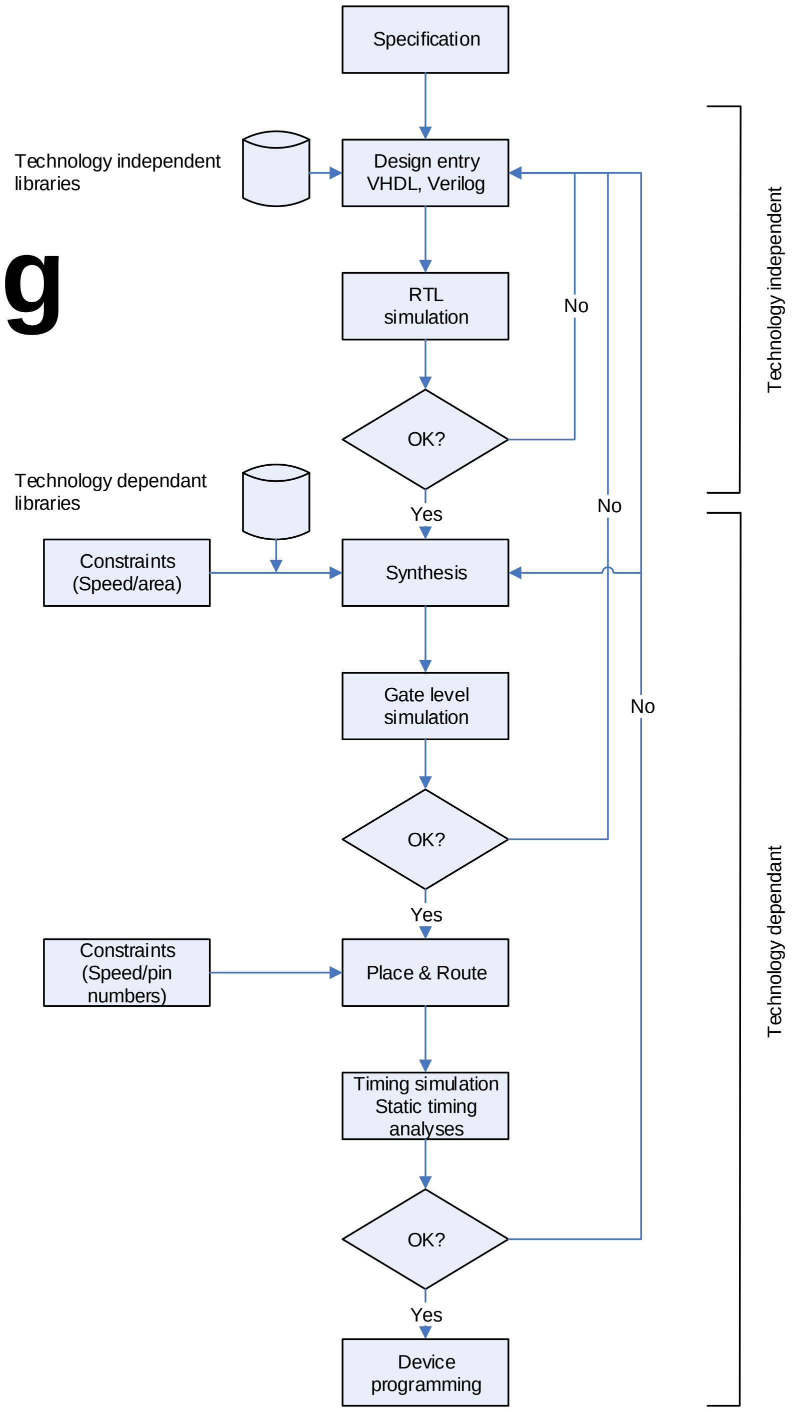
- t_{pg} : delay to generate all P_i, G_i
- t_{pg_block} : delay to generate all $P_{i:j}$, $G_{i:j}$
- t_{AND_OR} : delay from C_{in} to C_{out} of final AND/OR gate in *k*-bit CLA block

An *N*-bit carry-lookahead adder is generally much faster than a ripple-carry adder for $N > 16$



Timing simulation, programming

- Simulating synthesized or routed designs
- Can use the same *testbench* as for RTL designs
  - *Verification and test benches will be discussed further later…*
- Uses timing information for every component in use.
  - Requires much more resources than RTL simulation.
  - Can be slow for complex designs
    - Hence the option to simulate at gate level, before performing PAR.
- Device programming…
  - (Usually done from vivado, but third part tools *may* be used).

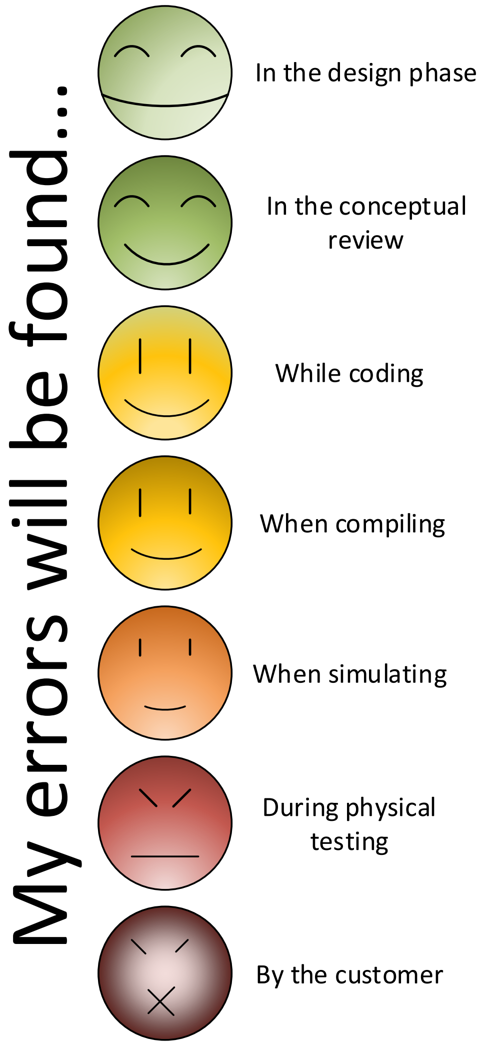


Testing

- «*Testing*» is to find physical errors in a device.
  - «*Verification*» is to check the design
    - although we use «*test benches*» for simulation

- Design for testability
  - Means that we design for physical testing.
  - We *may* touch this later in the course.

- *Spend more time in early phases!*
  - Avoid spending *much more* time fixing bugs later



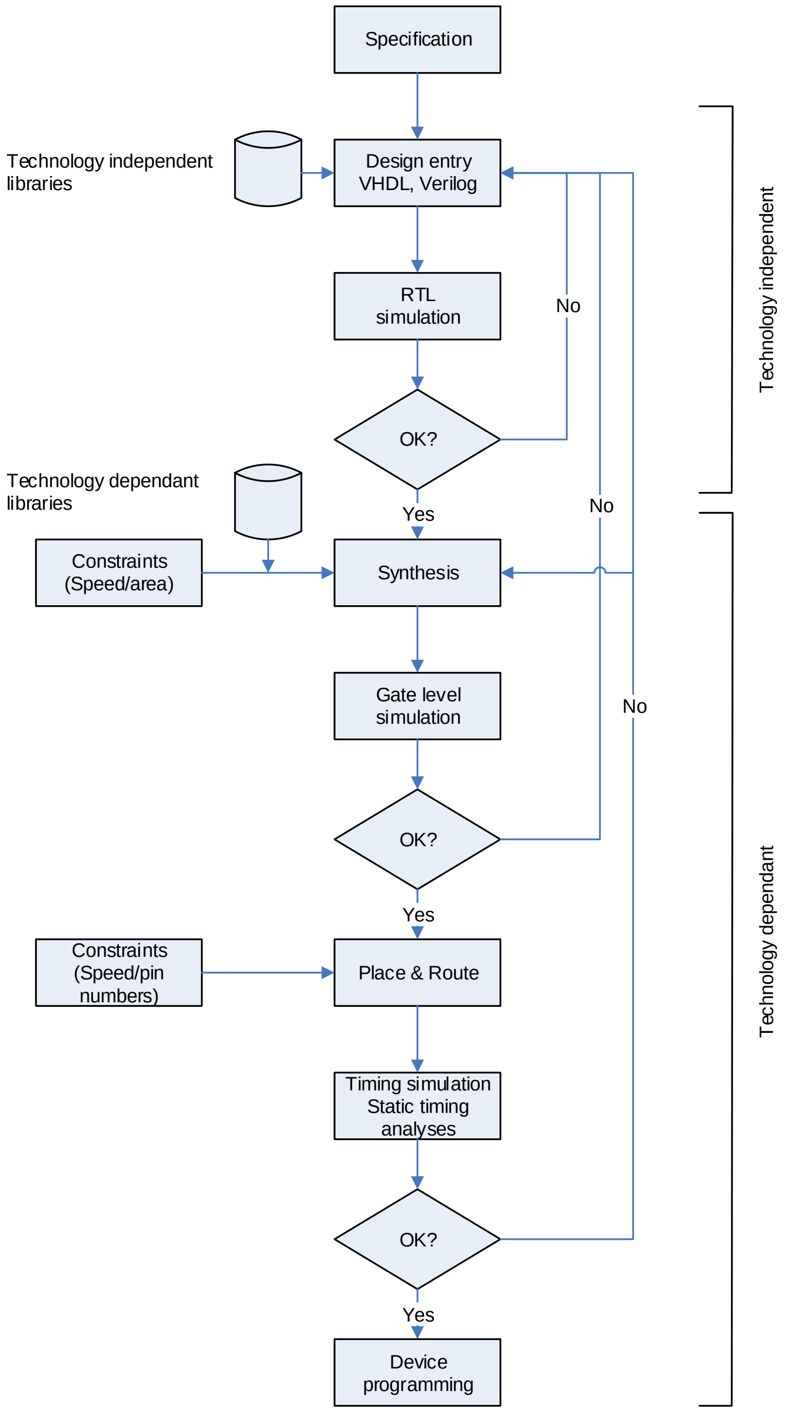
Introduction to course hardware and software tools

- Zedboard
- Questa
- Vivado

- ROBIN wiki: https://robin.wiki.ifi.uio.no/Hovedside
  - Software
    - FPGA tools https://robin.wiki.ifi.uio.no/FPGA_tools
- Cook book and ZedBoard documentation
  - Canvas – IN3160
    - Cookbook_v3_5.pdf
    - ZedBoard HW UG vX_X.pdf
  - Zynq intro video: https://www.xilinx.com/video/soc/zedboard-overview-featuring-zynq.html

Digital Design tools…

- Design entry:
  - Use your favourite HDL text editor (Notepad++, Emacs, Vivado or Questa).
- Simulation (RTL, Gate Level, Timing)
  - Here: Typically using Questa (=Modelsim)
- Synthesis, Implementation, Programming
  - Vendor specific tools…
    - Here: Vivado by Xilinx



Simulation and test benches

- Simulation can be run using three different approaches:
  1. Manually setting inputs and specifying time intervals in the GUI or console
     - This way is tedious if much testing is to be done.
     - *Normally this is only done initially.*
  2. To make scripts (tcl for Questa) in a separate (.do) file.
     - The *script commands will be added to the console during manual use, and can be copied as text into a .do file.*
     - setting up the simulation windows can be done reusing script commands.
  3. Create a test bench in VHDL
     - *This is the preferred method*
       - possible in combination with running scripts
     - VHDL can be used to generate code for applying test vectors sequentially to the inputs of an entity for simulating.
     - Test bench code is not synthesizable
     - easy to read and use test data for each particular design,
     - Can be used both prior and post synthesis or implementation

Suggested reading, Mandatory assignments

- D&H:
  - 1.4 p 11-13
  - 1.5 p 13-16
  - 1.6 p 16-17
  - 2.1 p 22-28
  - 2.2 p 28-30
  - 2.3 p 30-34
  - 3.1-3.5 p 43-51 = repetition (known from previous courses)

- Oblig 1: «Design Flow»
  - See canvas for further instruction.

Note: Some of this content will be covered in depth in later lectures.
- *Read this to familiarize yourself with content, form and language.*