TFE4152 - Lecture 1

Design of Integrated Circuits

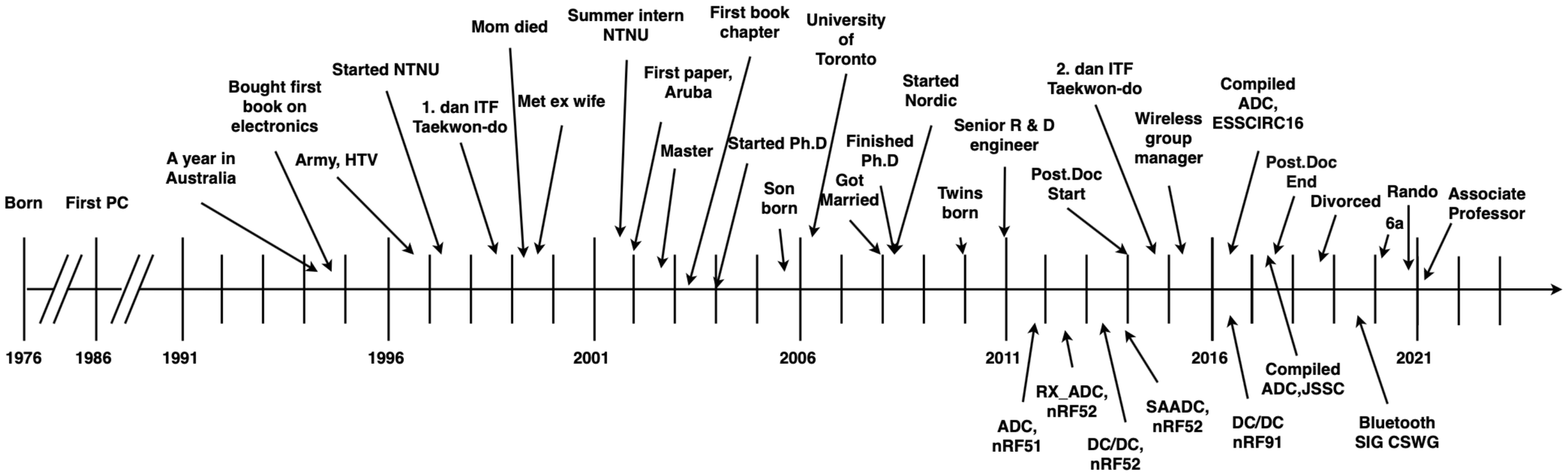
Source

Goal for today

- Who are we
- Introduce Course
- Introduce Exercises
- Introduce Project
- Introduce Software
- Why do we make ICs?
- Introduce what skills you'll learn
- Introduce how we're going to learn it

Who

Carsten Wulff carstenw@ntnu.no



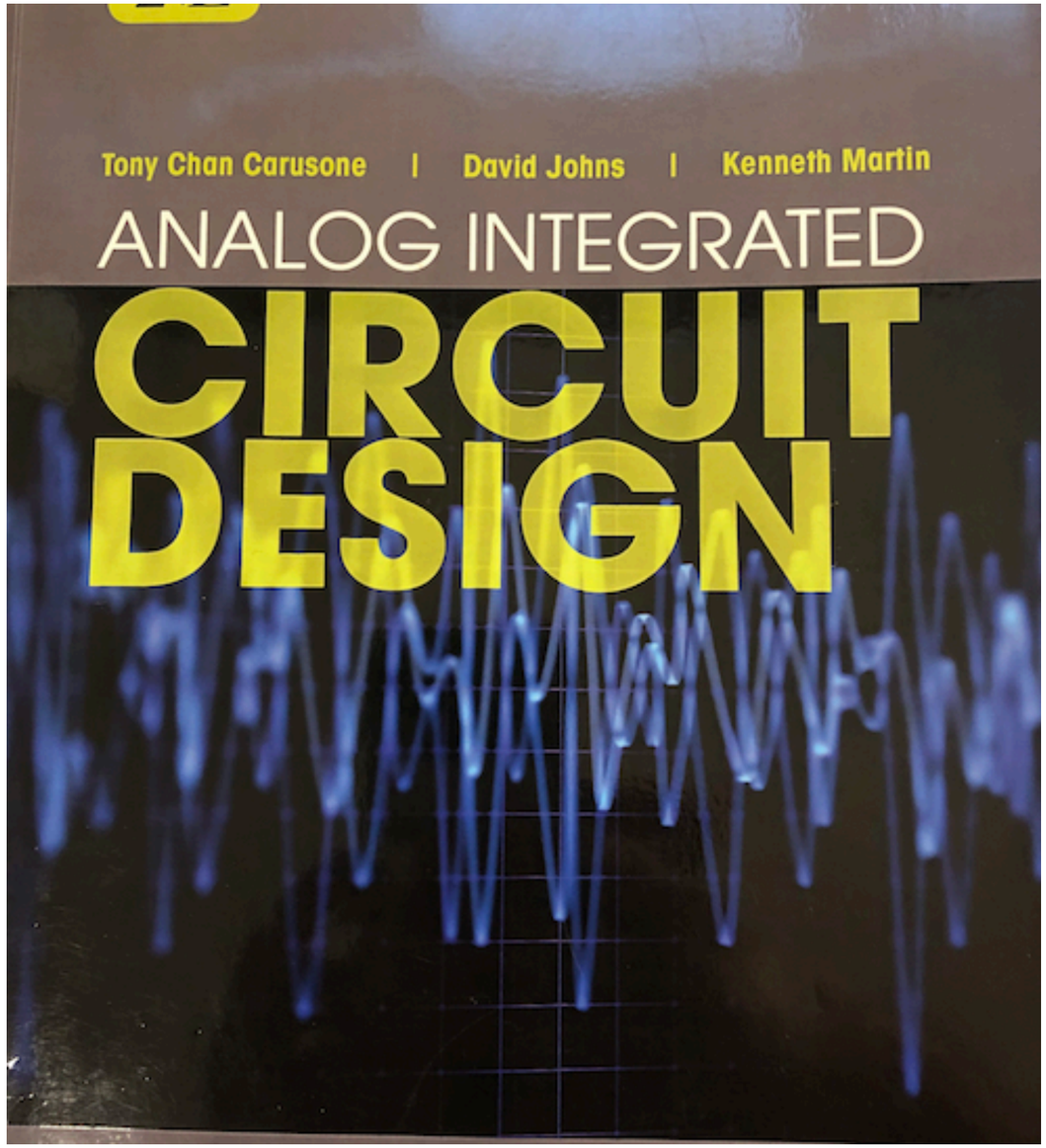
Teaching assistants

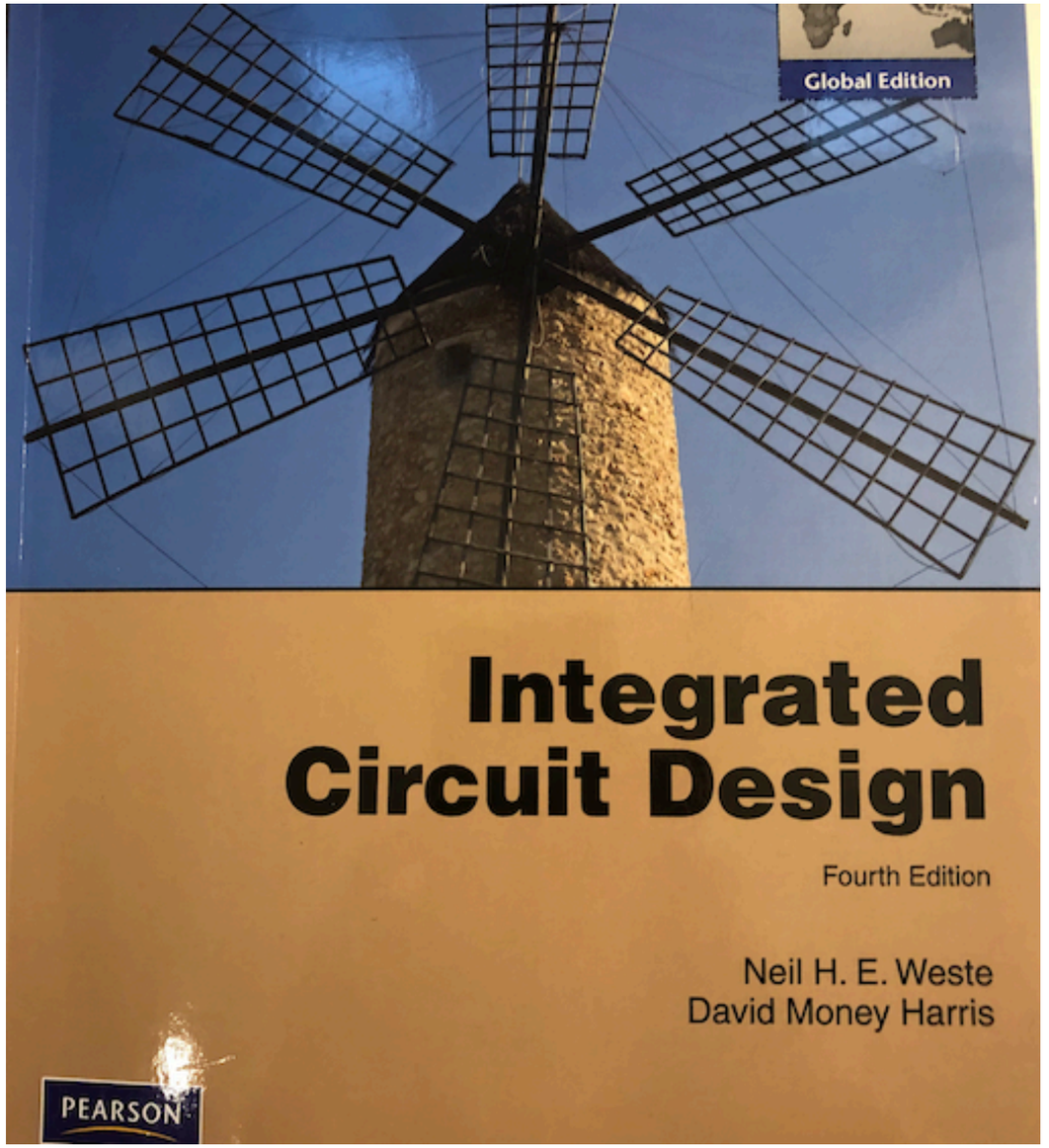
- Ehsan Lari
- Erlend Kristiansen Berg
- Jonas Gjendem Røysland

Course

Description

Time schedule


Tony Chan Carusone | David Johns | Kenneth Martin
ANALOG INTEGRATED
CIRCUIT
DESIGN


Global Edition
Integrated
Circuit Design
Fourth Edition
Neil H. E. Weste
David Money Harris
PEARSON

Curriculum

Tentative lecture plan

Week	Book	Monday	Book	Friday
34		Introduction, what are we going to do in this course. Why do you need it?	WH 1	Manufacturing of integrated circuits
35	CJM 1.1	pn Junctions	CJM 1.2 WH 1.3, 2.1-2.4	Mosfet transistors
36	CJM 1.2 WH 1.3, 2.1-2.4	Mosfet transistors	CJM 1.3 - 1.6	Modeling and passive devices
37		Guest Lecture - Sony	CJM 3.1, 3.5, 3.6	Current mirrors
38	CJM 3.2, 3.3,3.4 3.7	Amplifiers	CJM, CJM 2 WH 1.5 WH 15	SPICE simulation and layout
39		Verilog		Verilog
40	WH 1.4 WH 2.5	CMOS Logic	WH 3	Speed
41	WH 4	Power	WH 5	Wires
42	WH 6	Scaling Reliability and Variability	WH 8	Gates
43	WH 9	Sequencing	WH 10	Datapaths - Adders
44	WH 10	Datapaths - Multipliers, Counters	WH 11	Memories
45	WH 12	Packaging	WH 14	Test
46		Guest lecture - Nordic Semiconductor		
47	CJM	Recap of CJM	WH	Recap of WH

Exam

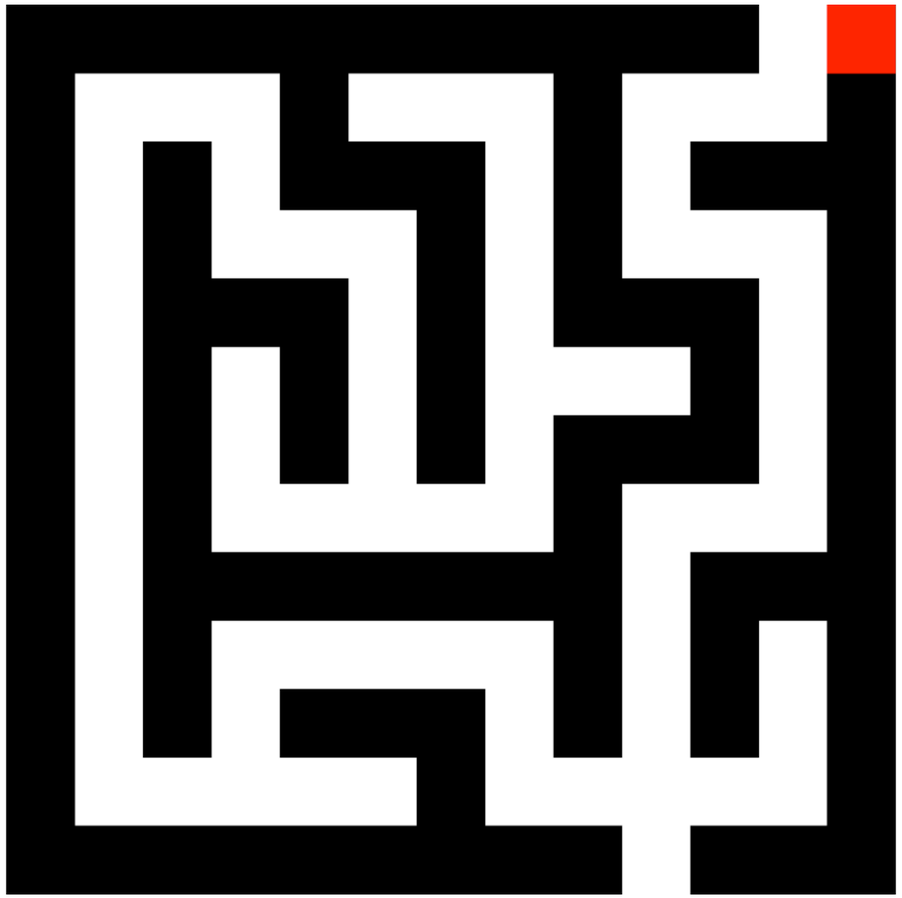
- December 2021?
- 4 hours
- D aids code - No handwritten or printed aids allowed. Preapproved calculator, in accordance to the exam regulations, allowed
- 70% of the final grade
- A - F grade (F = Fail)

Exercise

Facts

- 4 exercises on blackboard (somewhat modified from last year), 1 more to come, exercise 6 is special
- last years exercise and solutions on blackboard (soon)
- must have 4 of 6 exercises approved
- strict deadline (Friday XX 23:59)
- no second chances

Exercise 6: Maze in SystemVerilog



Exercise 6: Maze

1. Explain how the mazeEscaper.v works
2. Make a soluition that is better
3. Upload PDF with explanation and verilog to Blackboard
4. Send mazeEscaper.v as attachment to carstenw@ntnu.no with subject TFE4152-Comp-Maze

Plan

Date	Week	Topic
2021-09-10	36	PN Junctions
2021-09-24	38	Transistors
2021-10-08	40	Current Mirrors and Amplifiers
2021-10-22	42	CMOS logic
2021-11-05	44	Logic circuits
2021-11-19	46	Maze

Project

- 30 % of final grade
- Groups of 2 people. Find a partner soon. Sign up on blackboard.
- Deadline: 19'th November before 12:00 (24 hour format).
- Strict deadline, $t > 12:00 \equiv fail$. Both members in group must submit report.

A 10 000 Frames/s CMOS Digital Pixel Sensor

IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 36, NO. 12, DECEMBER 2001

A 10 000 Frames/s CMOS Digital Pixel Sensor

Stuart Kleinfelder, SukHwan Lim, Xinqiao Liu, and Abbas El Gamal, *Fellow, IEEE*

***Abstract*—A 352 × 288 pixel CMOS image sensor chip with per-pixel single-slope ADC and dynamic memory in a standard digital 0.18-μm CMOS process is described. The chip performs "snapshot" image acquisition, parallel 8-bit A/D conversion, and digital readout at continuous rate of 10 000 frames/s or 1 Gpixels/s with power consumption of 50 mW. Each pixel consists of a photogate circuit, a three-stage comparator, and an 8-bit 3T dynamic memory comprising a total of 37 transistors in 9.4 × 9.4 μm with a fill factor of 15%. The photogate quantum efficiency is 13.6%, and the sensor conversion gain is 13.1 μV/e$^-$. At 1000 frames/s, measured integral nonlinearity is 0.22% over a 1-V range, rms temporal noise with digital CDS is 0.15%, and rms FPN with digital CDS is 0.027%. When operated at low frame rates, on-chip power management circuits permit complete powerdown between each frame conversion and readout. The digitized pixel data is read out over a 64-bit (8-pixel) wide bus operating at 167 MHz, i.e., over 1.33 GB/s. The chip is suitable for general high-speed imaging applications as well as for the implementation of several still and standard video rate applications that benefit from high-speed capture, such as dynamic range enhancement, motion estimation and compensation, and image stabilization.**

***Index Terms*—ADC, CMOS image sensor, digital pixel sensor, high-speed imaging, image sensor, memory.**

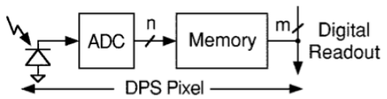


Fig. 1. Simple DPS pixel block diagram.

rate applications that require high-speed capture such as sensor dynamic range enhancement and motion estimation [8]–[10].

The main drawback of DPS is that it uses more transistors per pixel than conventional analog image sensors and therefore can have larger pixel sizes. Since there is a lower bound on practical pixel size imposed by the wavelength of light, imaging optics, and dynamic range, this drawback quickly disappears as CMOS technology scales down to 0.18 μm and below. Designing image sensors in such advanced technologies, which will be needed for implementing true camera-on-chip systems, is challenging due to the scaling of supply voltage and the increase in leakage currents [12].

In this paper, we describe a 352 × 288 CMOS DPS with per-pixel ADC and digital memory fabricated in a standard

Goal

Be inspired by the paper, and design a similar system.

Design analog circuits in SPICE and digital circuits in SystemVerilog.

Minimum implementation

- 2 x 2 pixel array in SystemVerilog
- state machine to control reset, exposure, analog-to-digital conversion, and readout of the pixel array
- SPICE of pixel sensor (sensor, comparator, memory)
- Report documenting that the circuits (analog and digital) work as designed

Things it's OK to ignore

- Transistor corners
- Gray counter (ask me why)
- Voltage variation
- Temperature variation
- "nice to have features", like power optimalization, testability

What you get

github.com/wulffern/dicex

```
project/
├── spice/
│   ├── Makefile                # See https://www.gnu.org/software/make/manual/html_node/Introduction.html
│   ├── pixelSensor.cir         # Empty circuit for pixelSensor
│   └── pixelSensor_tb.cir      # SPICE testbench for pixelSensor, emulates verilog
└── verilog/
    ├── Makefile
    ├── pixelSensor.fl          # Verilog file list
    ├── pixelSensor_tb.gtkw     # Save file for GTKWave
    ├── pixelSensor_tb.v        # Verilog testbench for pixelSensor
    └── pixelSensor.v           # Verilog model of analog pixelSensor circuit
```

Report 1/2

- **Introduction** = Why?
- **Theory** = How?
  - As little information as possible. Give references to sources. Assume that the reader has read the paper.
- **Implementation** = What?
  - Describe how you implemented your design
  - State diagrams, with explanation
  - Circuit diagrams, with explanation
  - One sub-chapter per block

Report 2/2

- **Verification** = Are you sure it works?
  - Describe your testbenches, and how you verified your design
  - Describe key results
- **Discussion and conclusion** = Why do you deserve a good grade?
- **Appendix**
  - SPICE netlist
  - SystemVerilog netlist
  - SystemVerilog testbench

You need to start now to be able to complete the project with a good grade

Proposed plan

Week	Plan
34	Register group
35	Read and understand paper
36	Sketch what you want to do
37	Write theory chapter in report
38	Design & simulation
39	Design & simulation
40	Design & simulation
41	Design & simulation
42	Verification
43	Verification
44	Write report
45	Write report
46	**Deadline**

Software

You may use what ever you want, but exercises and project have been developed using AIM-Spice, ngspice, iverilog, and GKTwave on Ubuntu linux 20.10

SPICE

- AIM-Spice aimspice.com (Used for exercises)
- NGSpice ngspice , version 34 (Recommended for project)

SystemVerilog

- iverilog Icarus Verilog, version v11_0
- gtkwave GTKWave, version 3.3.104
- (yosys Yosys, version 0.9)

Option 1 : Install everything on your computer

The software can be installed on Windows, Mac, and Linux

Pros	Cons
Once it's running, then it's easy	Take time to install
	Compile from source
	No support from us on installing

Option 2 : Use login.stud.ntnu.no

Pros	Cons
I've tested with login.ansatt.ntnu.no	GUI windows require a bit of work
Similar to real world (desktop + server)	Online

Option 2: Quickstart for login.stud.ntnu.no

ssh to login.stud.ntnu.no, run

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/wulffern/dicex/main/ubuntu_install.sh)"
```

Connect to login from a mac/linux so you get GUI also

```
ssh -c aes128-ctr -YC -o "ForwardX11Timeout 4W" <username>@login.stud.ntnu.no
```

Option 3: Ubuntu on docker image with VNC frontend

Run Ubuntu linux on your PC via docker and use a VNC client to connect

Pros	Cons
If it works, it's easy	Complex, so maybe there are issues
I've tested it	Large (> 3 GB)
Offline (not for first run though)	

dicex and ciceda

Design of Integrated Circuits exercises (dicex) are the testbenches and model files you'll need
https://github.com/wulffern/dicex

Custom IC Creator Electronic Design Automation (ciceda) is a docker image of a ubuntu linux with the necessary tools installed
https://github.com/wulffern/ciceda

dicex - Requirements

- Install Docker
- Install GIT
- Install TigerVNC

dicex - Getting started

In a terminal or powershell

```
git clone https://github.com/wulffern/dicex
cd dicex
./ciceda_mac.sh
```

See demo video

Lower your expectations on EDA software

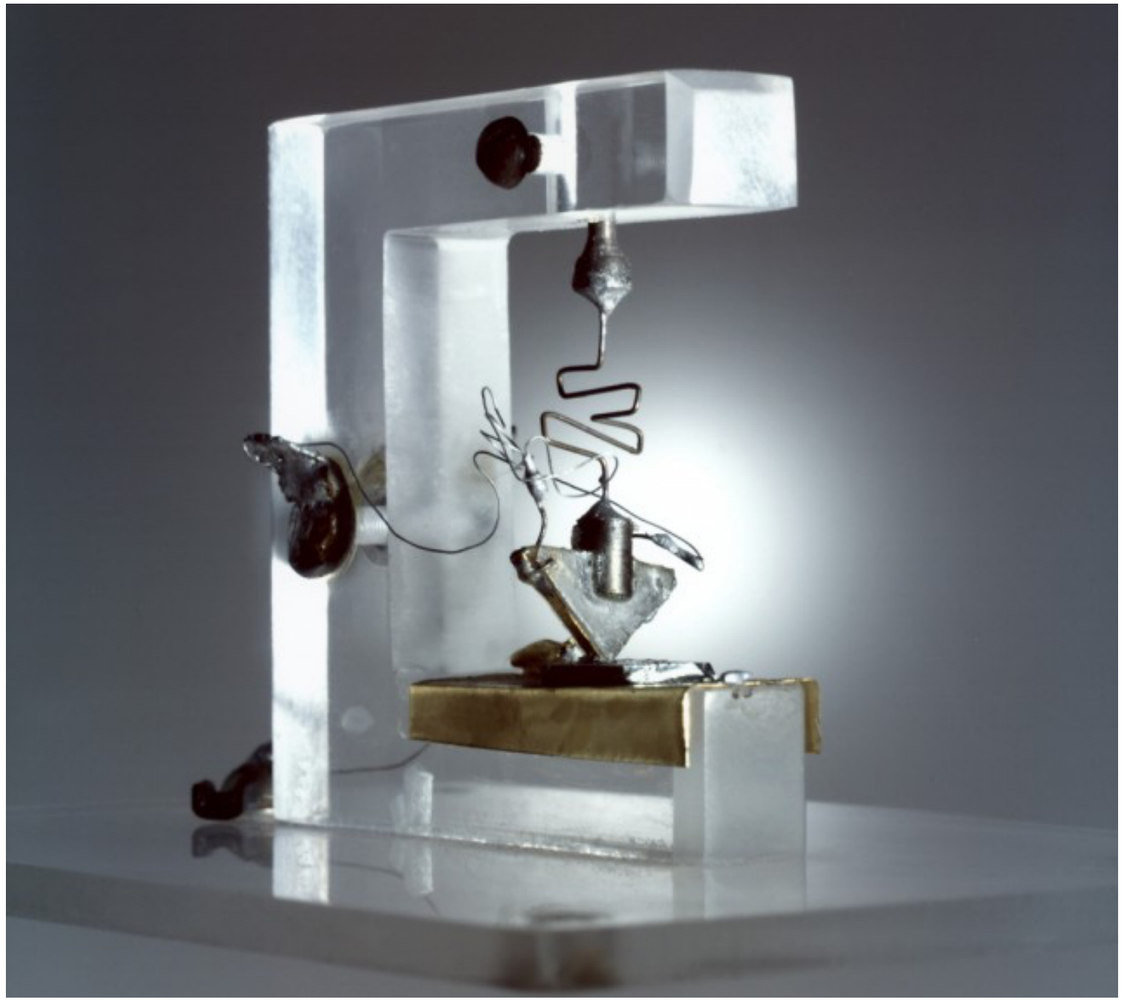
Expect that you will spend at least 2π times more time than planned *(mostly due to software issues)*

Questions?

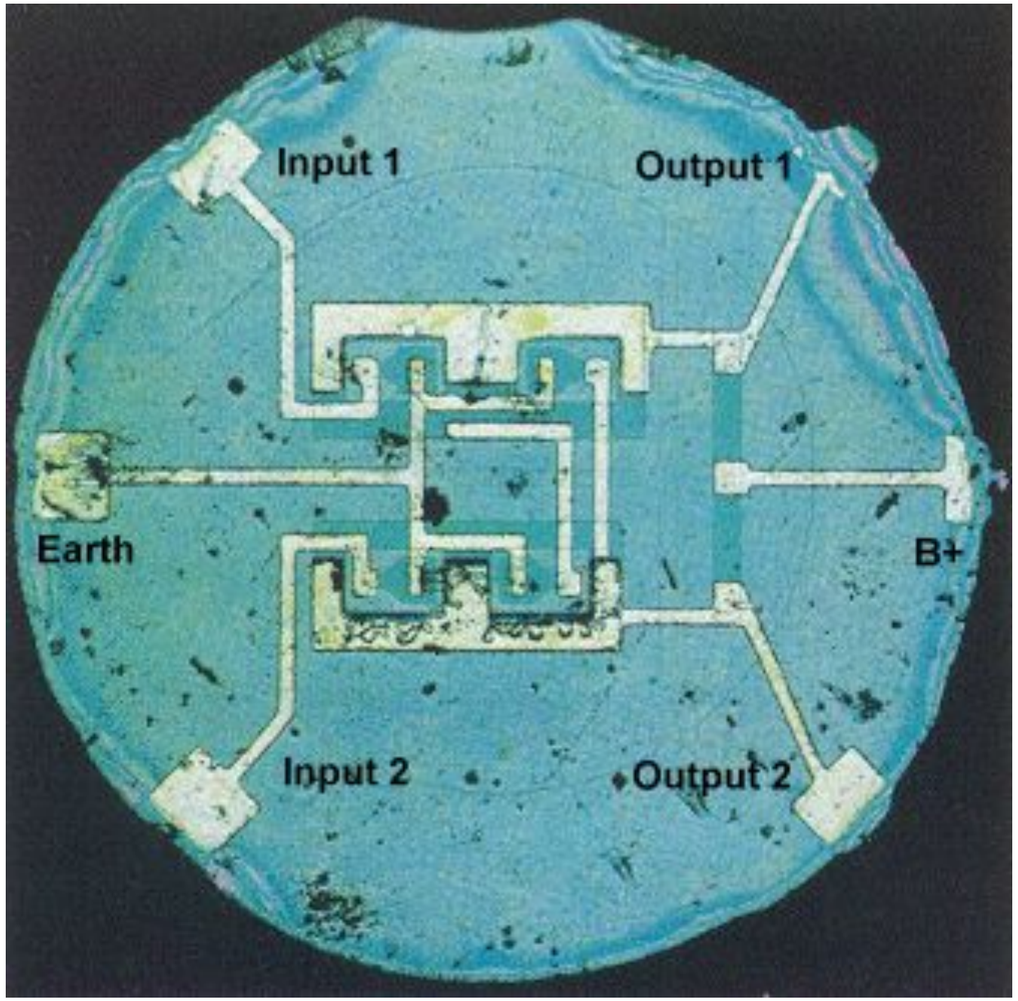
Memo time

https://github.com/wulffern/dic2021/tree/main/2021-06-13_why_integrated_circuits

Why do we make ICs?



1947: First transistor. Invented by Brattain, Bardeen and Shockley, Bell labs

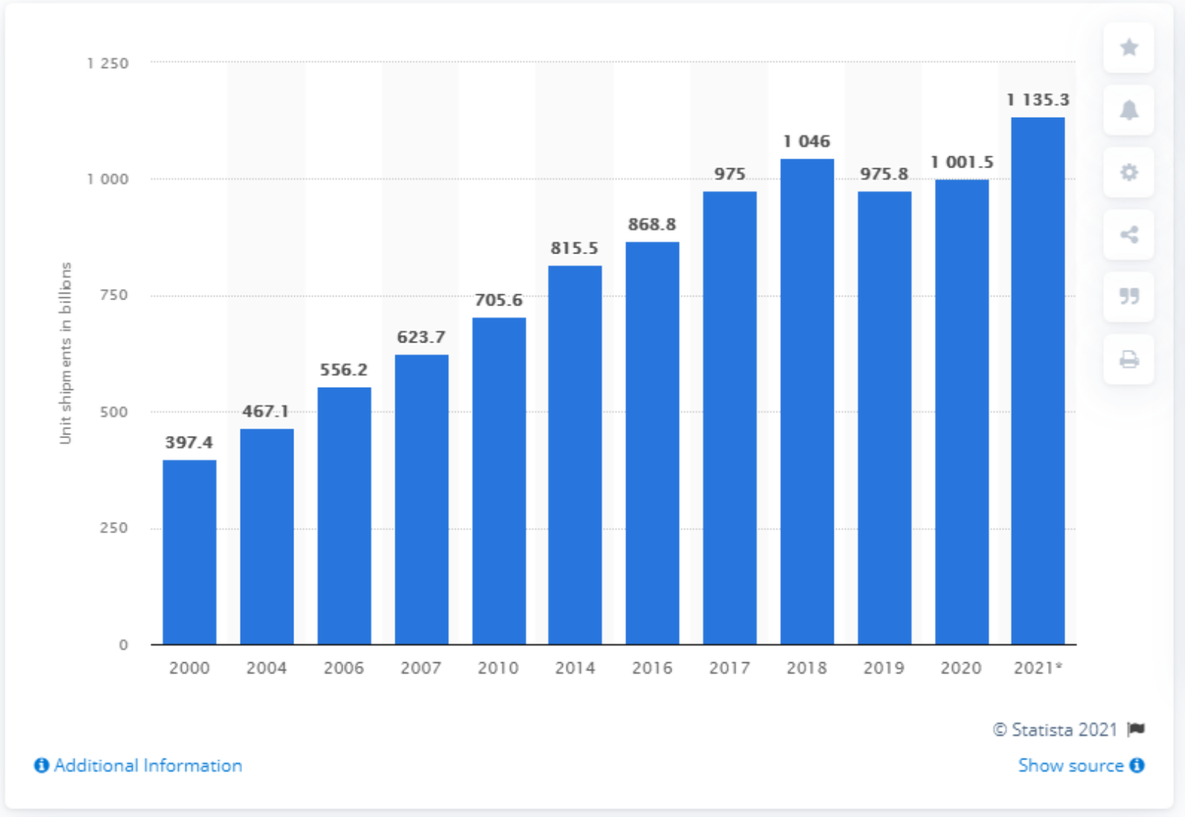


1961: First Monolithic Silicon IC Chip. Invented by Robert Noyce, Fairchild

Technology & Telecommunications › Hardware

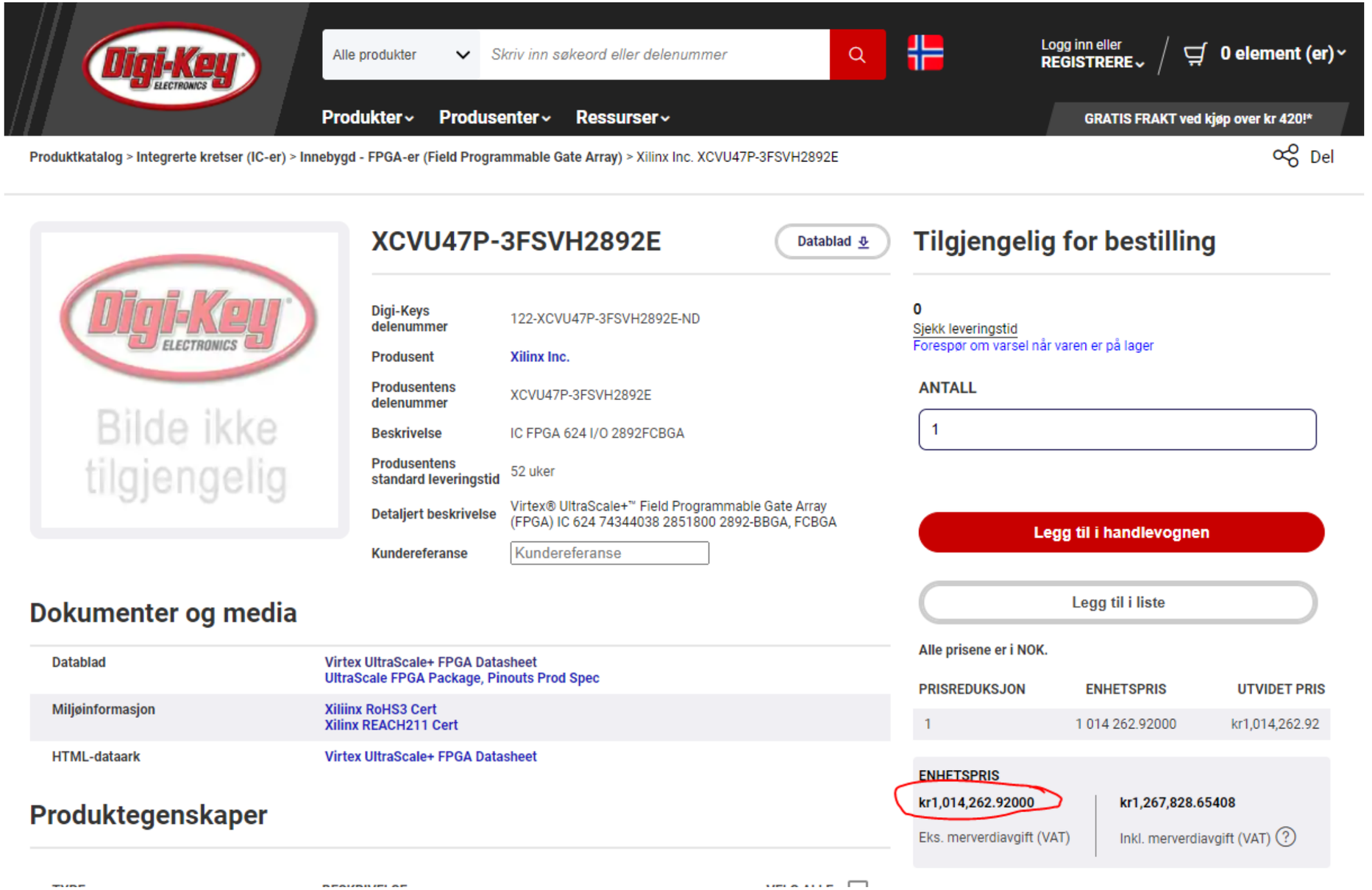
Semiconductor unit shipments worldwide from 2000 to 2021

(in billions)



https://www.statista.com/statistics/802632/world-semiconductor-shipments/

Virtex UltraScale+ VU19P (35 billion transistors)

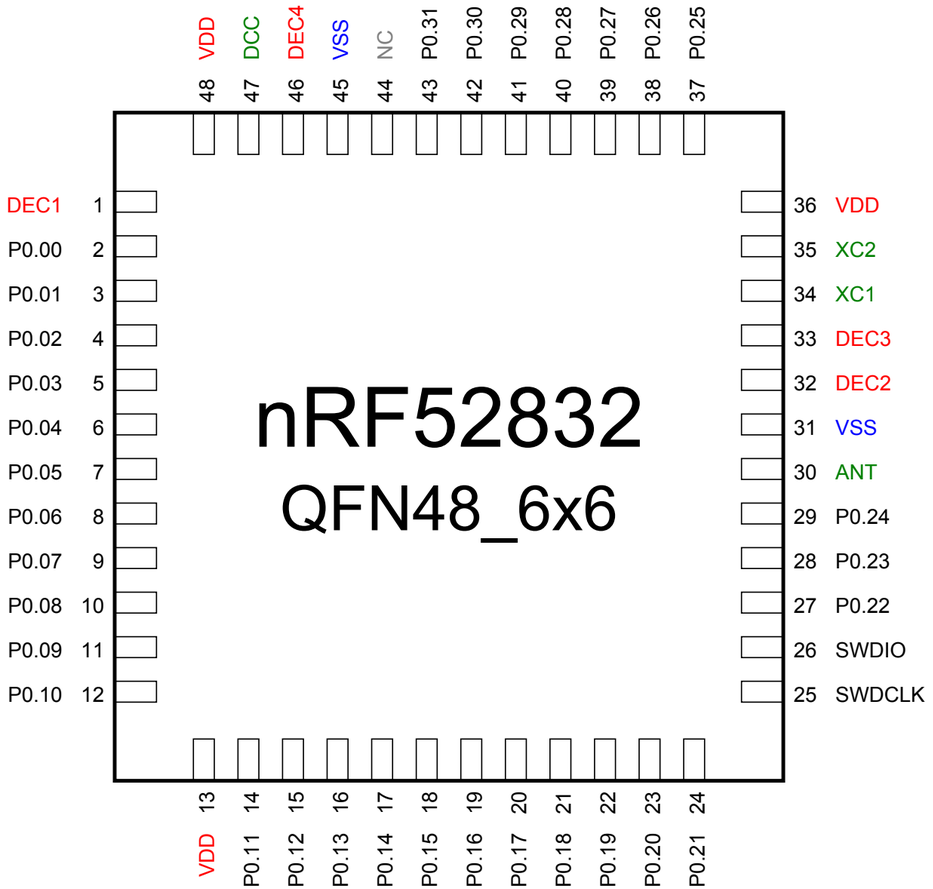

Digi-Key ELECTRONICS
Alle produkter
Skriv inn søkeord eller delenummer
Logg inn eller REGISTRERE
0 element (er)
Produkter
Produsenter
Ressurser
GRATIS FRAKT ved kjøp over kr 420!*
Produktkatalog > Integrerte kretser (IC-er) > Innebygd - FPGA-er (Field Programmable Gate Array) > Xilinx Inc. XCVU47P-3FSVH2892E
Del
Bilde ikke tilgjengelig
XCVU47P-3FSVH2892E
Datablad
Digi-Keys delenummer 122-XCVU47P-3FSVH2892E-ND
Produsent Xilinx Inc.
Produsentens delenummer XCVU47P-3FSVH2892E
Beskrivelse IC FPGA 624 I/O 2892FCBGA
Produsentens standard leveringstid 52 uker
Detaljert beskrivelse Virtex® UltraScale+™ Field Programmable Gate Array (FPGA) IC 624 74344038 2851800 2892-BBGA, FCBGA
Kundereferanse Kundereferanse
Tilgjengelig for bestilling
0
Sjekk leveringstid
Forespør om varsel når varen er på lager
ANTALL
1
Legg til i handlevognen
Legg til i liste
Alle prisene er i NOK.
PRISREDUKSJON ENHETSPRIS UTVIDET PRIS
1 1 014 262.92000 kr1,014,262.92
ENHETSPRIS
kr1,014,262.92000
Eks. merverdiavgift (VAT)
kr1,267,828.65408
Inkl. merverdiavgift (VAT)
Dokumenter og media
Datablad Virtex UltraScale+ FPGA Datasheet
UltraScale FPGA Package, Pinouts Prod Spec
Miljøinformasjon Xiliinx RoHS3 Cert
Xilinx REACH211 Cert
HTML-dataark Virtex UltraScale+ FPGA Datasheet
Produktegenskaper

Why would anyone buy a 1 M NOK FPGA?

Purpose

nRF51, nRF52, nRF53 series devices from Nordic Semiconductor are all the same type of tool.

- Add connectivity (Bluetooth, Zigbee, custom radio protocol) to your product.
- Provide a microcontroller; CPU + peripherals like interfaces (SPI, UART, ADC, COMP, I2C) to process information, and interface with the world.


nRF52832
QFN48_6x6
1 DEC1
2 P0.00
3 P0.01
4 P0.02
5 P0.03
6 P0.04
7 P0.05
8 P0.06
9 P0.07
10 P0.08
11 P0.09
12 P0.10
13 VDD
14 P0.11
15 P0.12
16 P0.13
17 P0.14
18 P0.15
19 P0.16
20 P0.17
21 P0.18
22 P0.19
23 P0.20
24 P0.21
25 SWDCLK
26 SWDIO
27 P0.22
28 P0.23
29 P0.24
30 ANT
31 VSS
32 DEC2
33 DEC3
34 XC1
35 XC2
36 VDD
37 P0.25
38 P0.26
39 P0.27
40 P0.28
41 P0.29
42 P0.30
43 P0.31
44 NC
45 VSS
46 DEC4
47 DCC
48 VDD

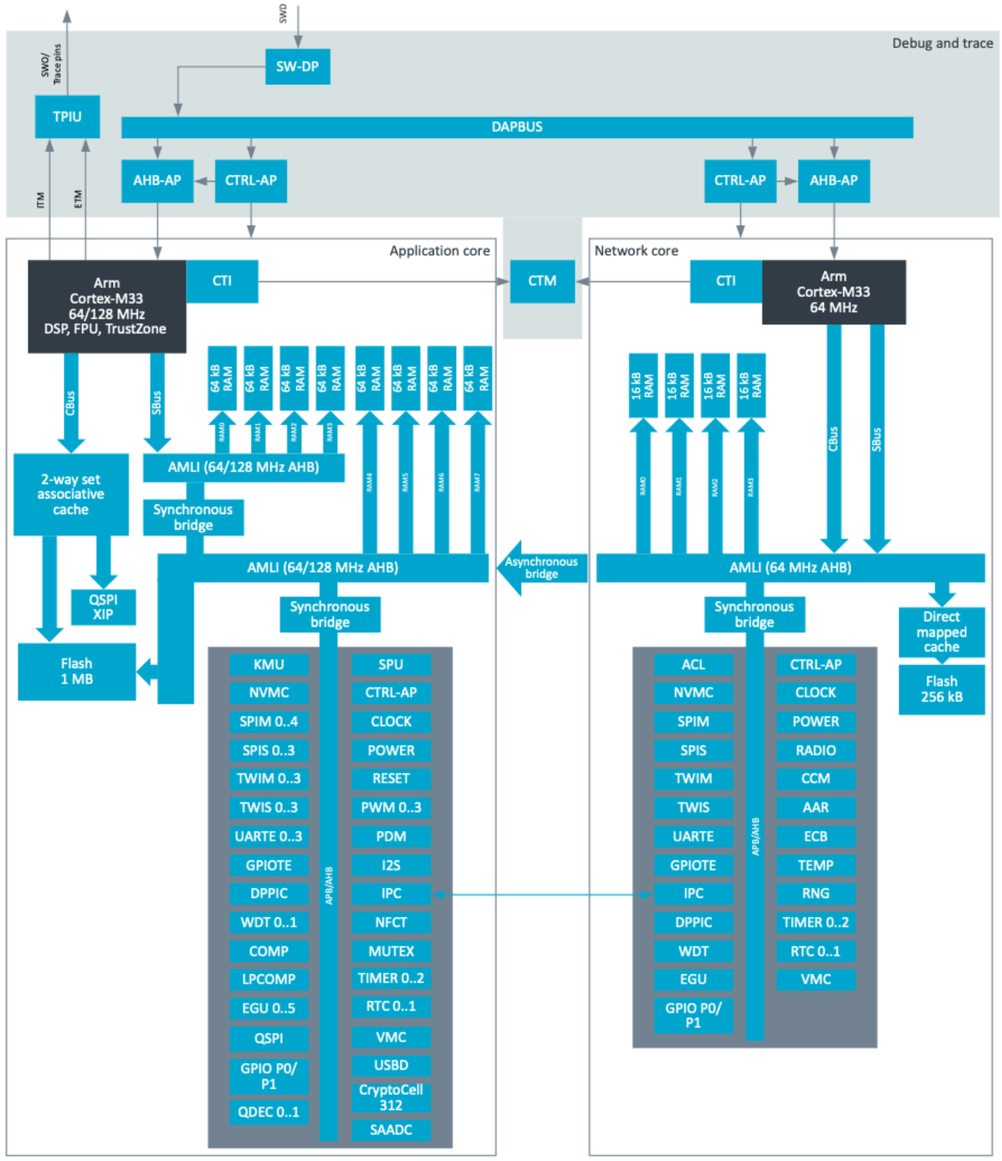


Figure 1: Simplified block diagram

Skills

- Firmware: signal processing, algorithms
- Infrastructure: Power management, reset, bias, clocks
- Domains: CPUs, peripherals, memories, bus systems
- Sub-systems: Radio’s, analog-to-digital converters, comparators
- Blocks: Analog Radio, Digital radio baseband
- Modules: Transmitter, receiver, de-modulator, timing recovery, **state machines**
- Designs: Opamps, **amplifiers**, **current-mirrors**, **adders**, random access memory blocks, **standard cells**
- Tools: schematic, layout, synthesis, place-and-route, **verilog**, **netlist**
- Physics: **transistor**, **pn junctions**, quantum mechanics

How will we learn it?

We need to build a solid foundation, so it's bottom up

Nomenclature

— *the devising or choosing of names for things, especially in a science or other discipline.*

Zen of IC design (stolen from Zen of Python)

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one-- and preferably only one --obvious way to do it.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.

Homework for Friday

Sam Zeloof Home Chip Fab