Darf es billiger als TTL sein?

Der µP im Shack

Ein Vortrag von Ulrich Habel, DF 4 IAH
26.01.2019

Überblick über die letzten Jahrzehnte

Darf es billiger als TTL sein?

- Verblüfft, regt zum Nachdenken an:

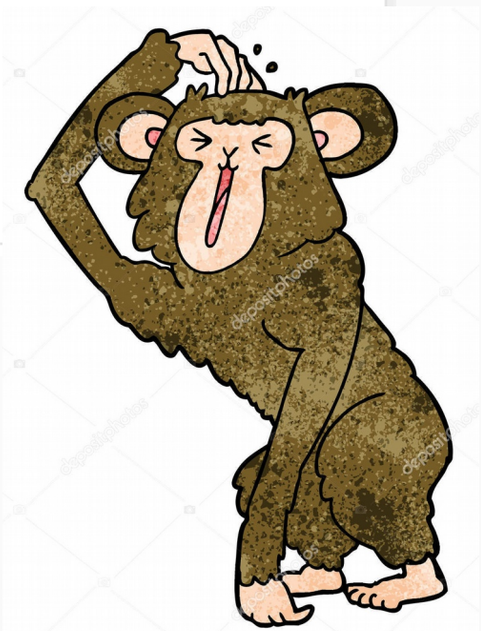
DIP28 ATMEGA328P-PU Microcontroller mit ARDUINO R3 UNO Bootloader or Not

Brandneu

EUR 1,60 bis **EUR 16,80** Aus China

Sofort-Kaufen

Kostenloser Versand



- Eine **MCU** kostet weniger als ein **7400** TTL IC (März 2018 Preis unter 1,- €)

Vergleich dazu: 4x NAND 7400 zu à 1,21€

☐			296-1626-5-ND	SN74LS00N	Texas Instruments	IC GATE NAND 4CH 2-INP 14DIP	38.541 - Sofort 10.000 - Lagerbestand des Herstellers	1,02000 €	1

Wo geht die Reise hin? Zeit zum Umdenken

Stichwort: Miniaturisierung

Zeitstrahl der Halbleiter-Entwicklung
1926 → 2019

1926: Der Feldeffekt-Transistor wird patentiert

1947: Erster Transistor wird implementiert

1961: Die TTL-Logik wird erfunden

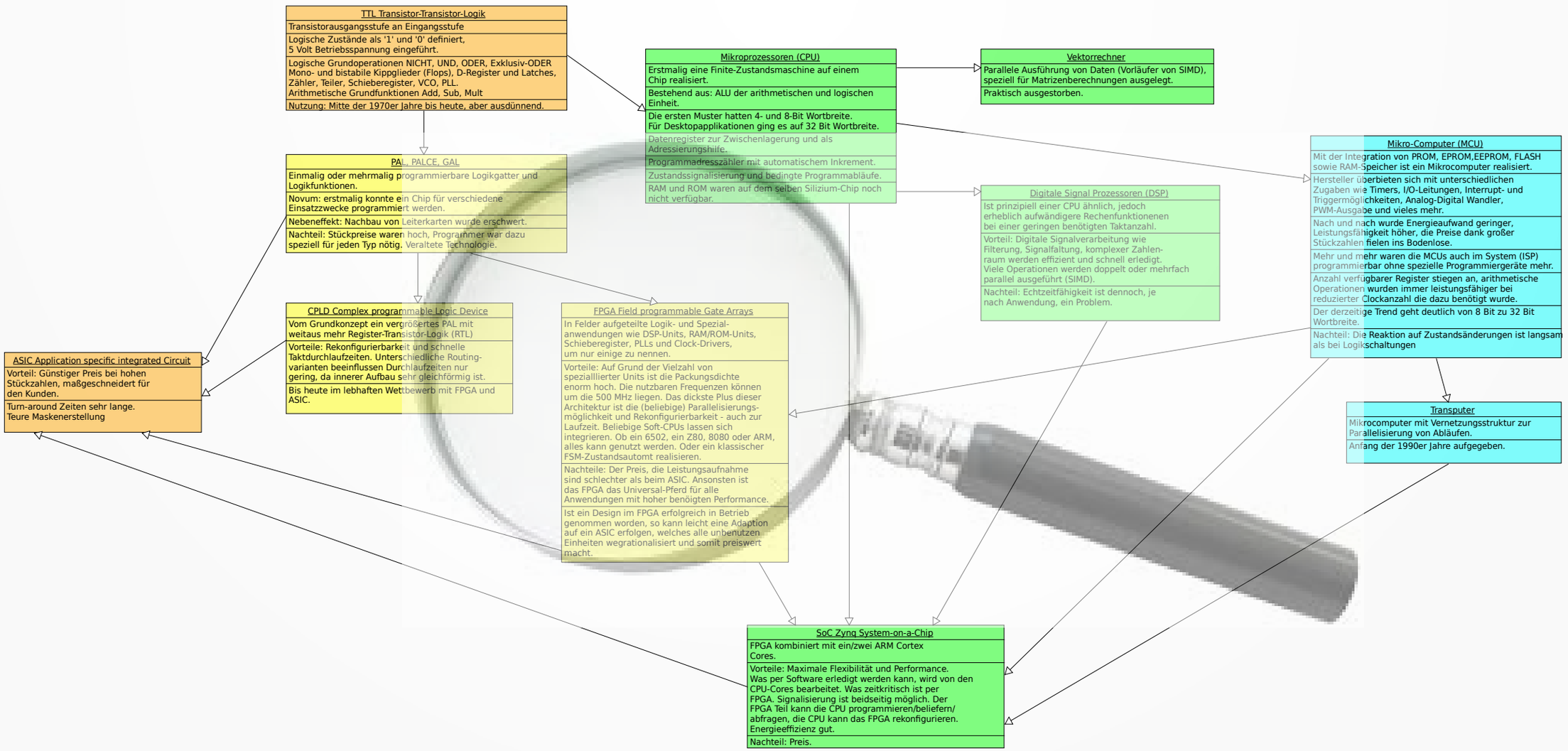
1964: Die 5400-Serie wird etabliert (Militär-Reihe)

1966: Die 7400-Serie wird vorgestellt (Konsumer-Reihe)

1970: PLA (Programmable Logic Array) maskenprogrammierbar von IBM
1971: Erste CPU wird aus TTL-Elektronik gebaut (8008, x86-Familie)
1971: Erste komplexe CPU 8080 von Intel vorgestellt
1972: Erste MCU 4004 von Intel mit 4-Bit Datenbreite vorgestellt
1978: PAL (Programmable Array Logik) im Feld programmierbar von MMI
1979: CPLD (Complex Programmable Logik Device) mehr Gatter und Interconnects
1979: FPGA (Field Programmable Gate Array) Funktionsbaugruppen verschaltbar
1978: Intel 8086 / 1979: Motorola 68000 mit 40.000 Transistoren
1982: Soc – Zwitterwesen aus FPGA und Mikroprozessor
1985: GAL (Generic Array Logik) lösch- und wiederbeschreibbar von Lattice Sem.
1985: ARM Architektur vorgestellt

2019

Zusammenfassung: Der große Plan



Der große Plan: die Transistor-Transistor-Logik

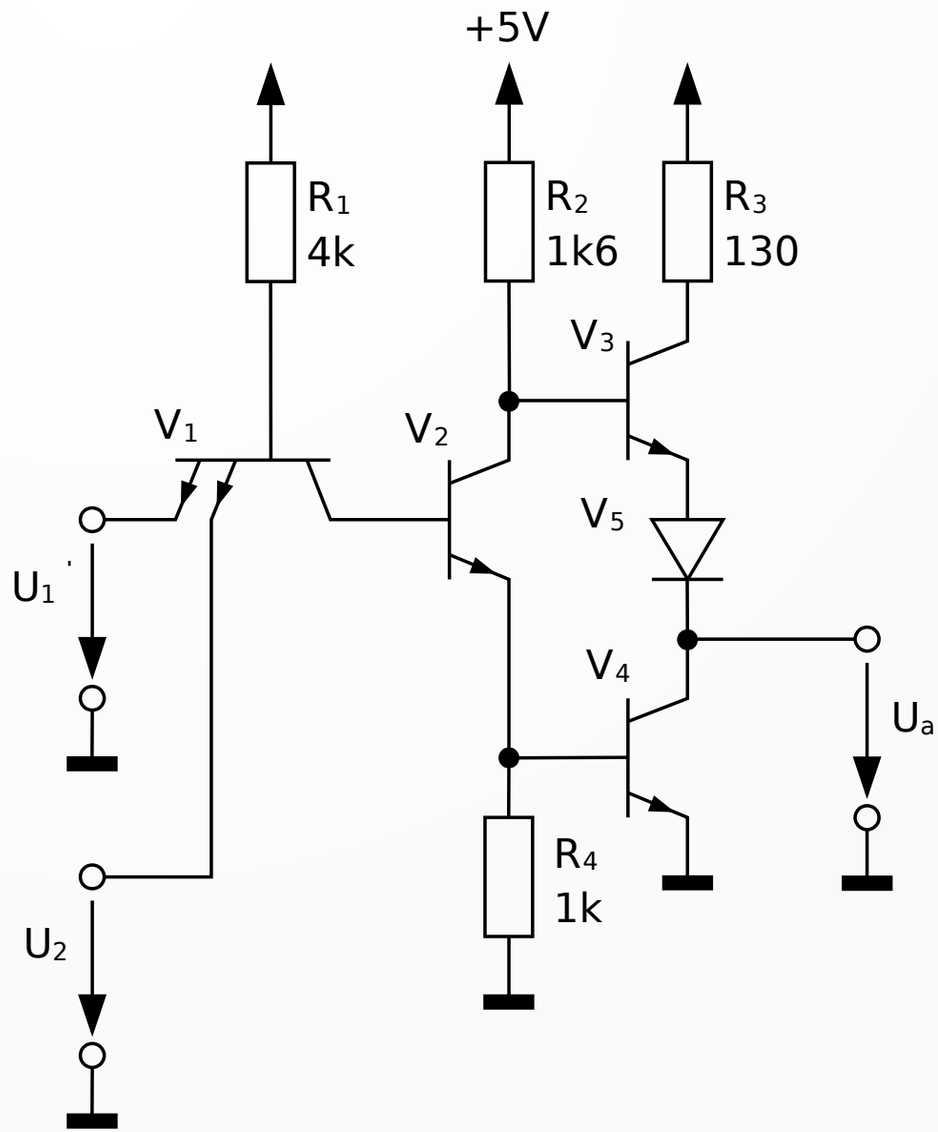
TTL Transistor-Transistor-Logik
Transistorausgangsstufe an Eingangsstufe
Logische Zustände als '1' und '0' definiert, 5 Volt Betriebsspannung eingeführt.
Logische Grundoperationen NICHT, UND, ODER, Exklusiv-ODER Mono- und bistabile Kippglieder (Flops), D-Register und Latches, Zähler, Teiler, Schieberegister, VCO, PLL. Arithmetische Grundfunktionen Add, Sub, Mult
Nutzung: Mitte der 1970er Jahre bis heute, aber ausdünnend.

TTL (Transistor-Transistor-Logik)

Beispiel anhand eines
7400 NAND Gatters

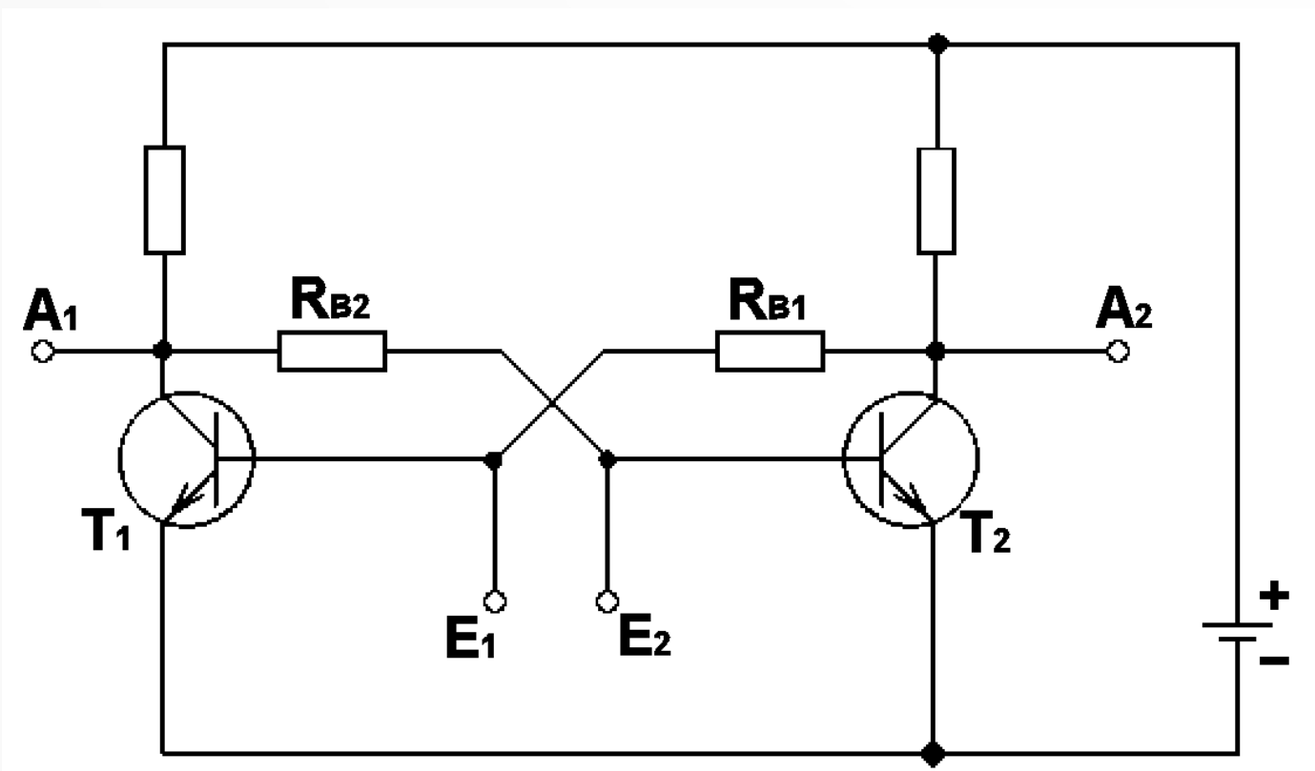
Vier solche Gatter umfassen
ein 7400 IC

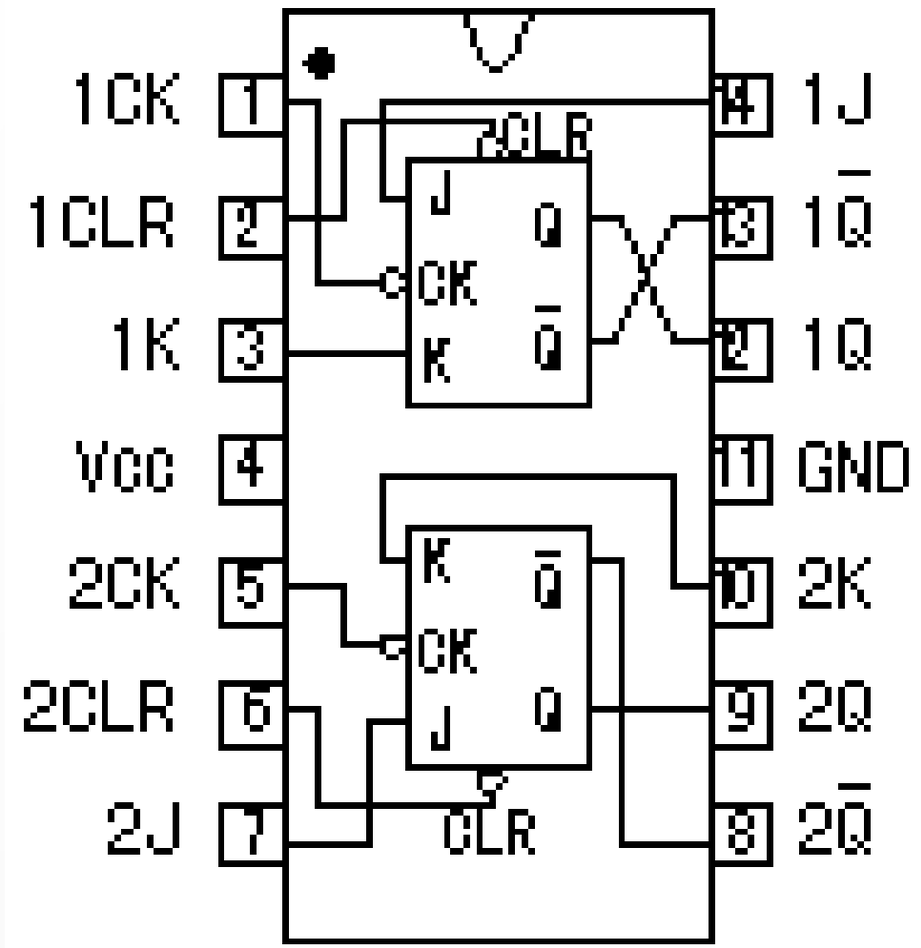
Betriebsspannung:
5.0 Volt



TTL: aufsteigende Komplexität je nach Anwendungsfall

Reset/**S**et-**F**lip**F**lop





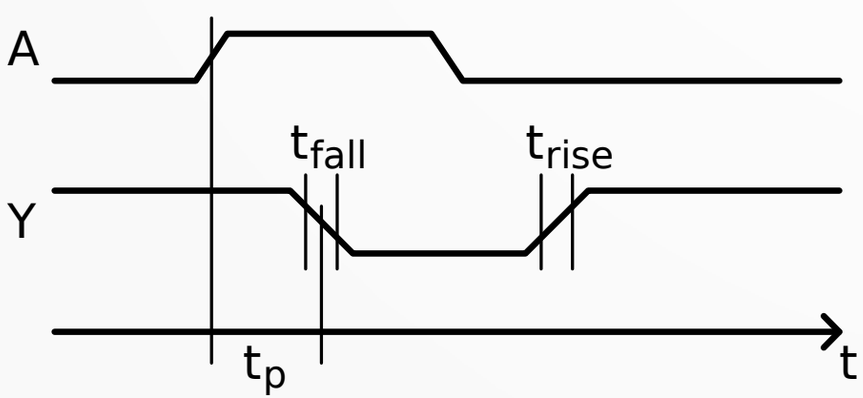
JK-RS-FF

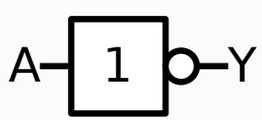
TTL: Flip-Flops, Register, ALU

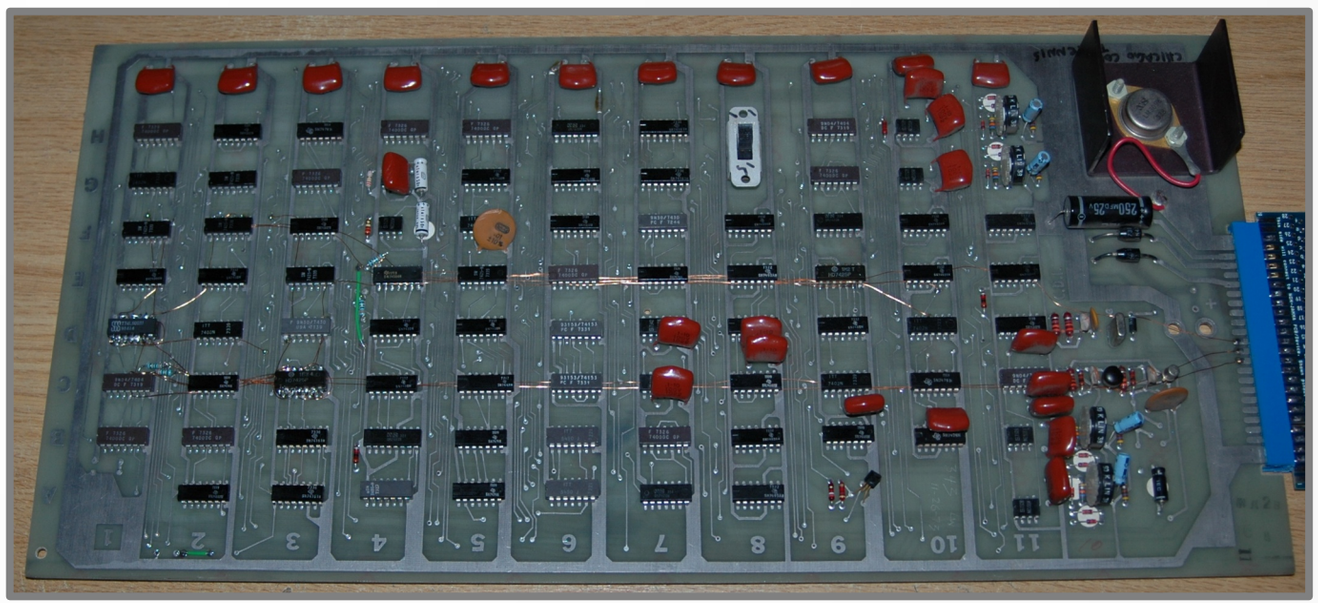
- Aufsteigende Komplexität:
  - Logikschaltungen: AND, OR, NOT, XOR, NAND, NOR
  - Flops: Familie der Kippglieder
    - Monostabil (Triggerung: gesetzt, kippt nach einer Zeit wieder zurück)
    - Bistabil (nur durch Logik gesteuert und nicht nach Zeit)
      - RS-FlipFlop (hat Setz- und Rückstelleingänge)
      - JK-RS-FlipFlop (hat taktsynchrone und asynchrone Set-/Rücksetzeing.)
      - D-FlipFlop und Latches (halten Eingangsdatum nach Clockimpuls)
      - Schieberegister
      - ALU (arithmetische Operationen: Addition und Subtraktion)

TTL: von den **Vorteilen** zu den **Nachteilen**

Deterministische Durchlaufzeiten







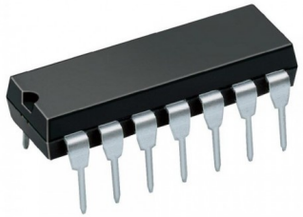
Unübersichtliche Komplexizität eines TTL-Grabs

Ergo: mit steigender Komplexizität nicht mehr handzuhaben

. . . die *Marktanteile* für **TTL** schwinden

- **Auswahl** an nativer TTL-Elektronik **dünnt sich aus**.

- TTL ICs zum Teil nicht mehr im **DIP** / **DIL**-Gehäuse verfügbar.



- **Verfügbarkeit** der 74xx Familie **reduziert**,
- **Preise** für noch aktive ICs **steigen**,
- Last-Order Aufrufe besiegeln die **Ab-kündigungen**.

TTL: am Scheideweg zwischen der Rekonfigurierbaren Logik und der µP-Architektur

- An dieser Stelle trennt sich zunächst einmal der Weg um den **unterschiedlichen Anforderungen** gerecht zu werden in:

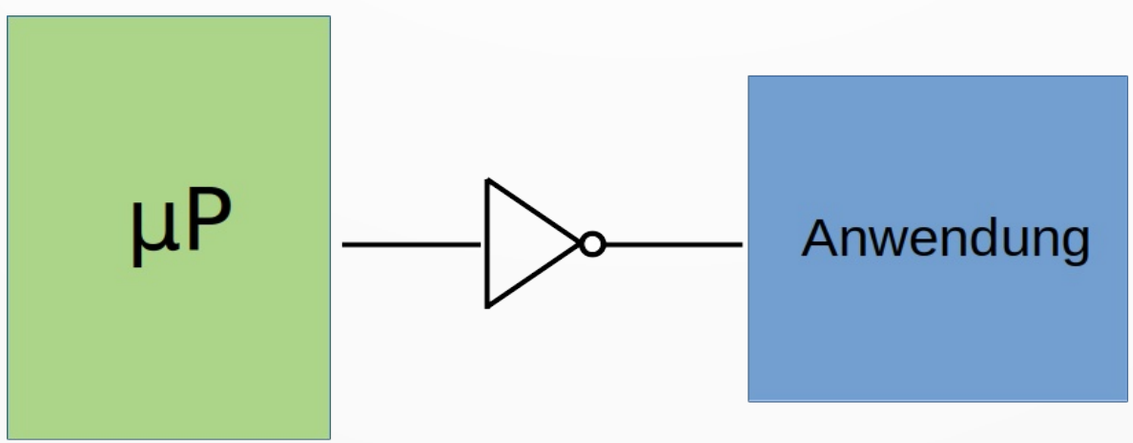
- Logik-basierenden Ansätzen wie die **CPLD**- und die **FPGA**-Technologie.
  (ab Seite 31)

- … und der **µP**-basierten Technologie, die durch ein Stück Software schrittweise mit der Außenwelt interagiert.

1. Betrachtung: TTL ↔ µP (CPU, MCU)

- **Komplexizität:**
  - **TTL** Zeitkritische Signalformung und -generierung
  - **µP** Programmabläufe als erweiterte State-Machines

- **TTL bisher häufig als „Glue-Logic“ am Prozessor genutzt:**

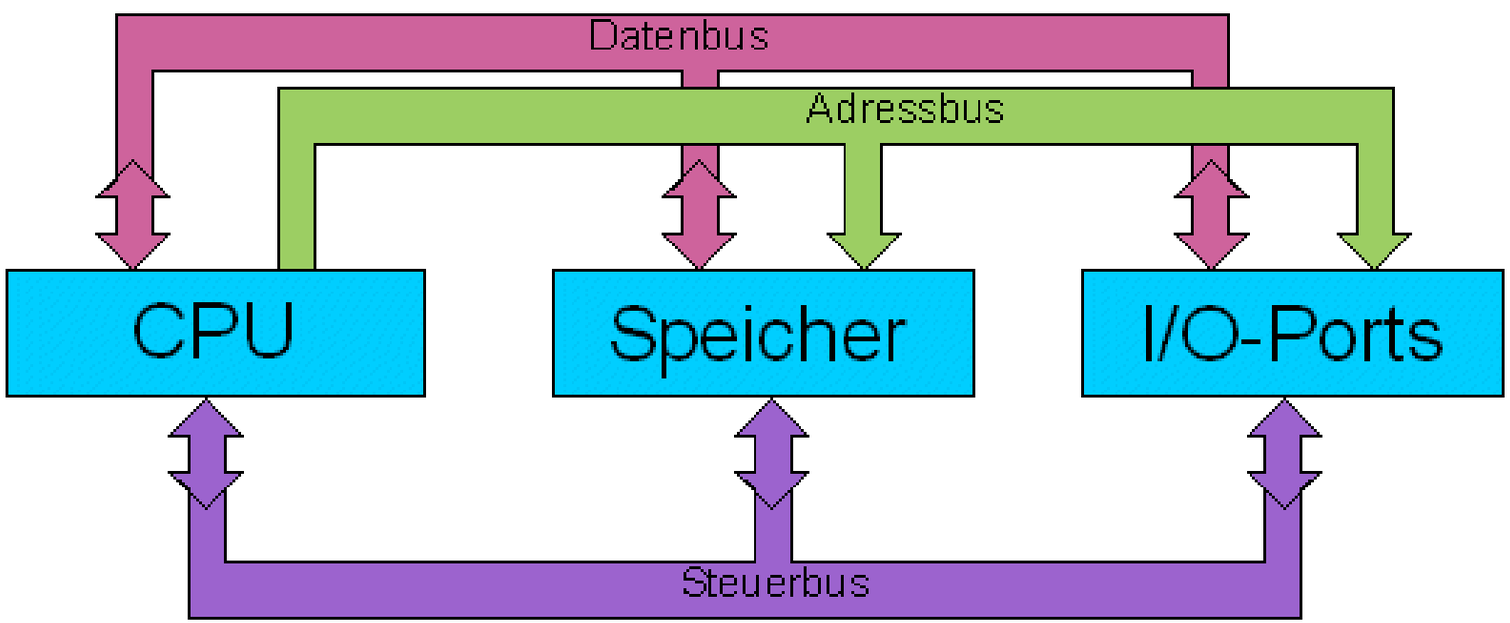


Der große Plan: Mikroprozessoren (CPU)

Mikroprozessoren (CPU)
Erstmalig eine Finite-Zustandsmaschine auf einem Chip realisiert.
Bestehend aus: ALU der arithmetischen und logischen Einheit.
Die ersten Muster hatten 4- und 8-Bit Wortbreite. Für Desktopapplikationen ging es auf 32 Bit Wortbreite.
Datenregister zur Zwischenlagerung und als Adressierungshilfe.
Programmadresszähler mit automatischem Inkrement.
Zustandssignalisierung und bedingte Programmabläufe.
RAM und ROM waren auf dem selben Silizium-Chip noch nicht verfügbar.

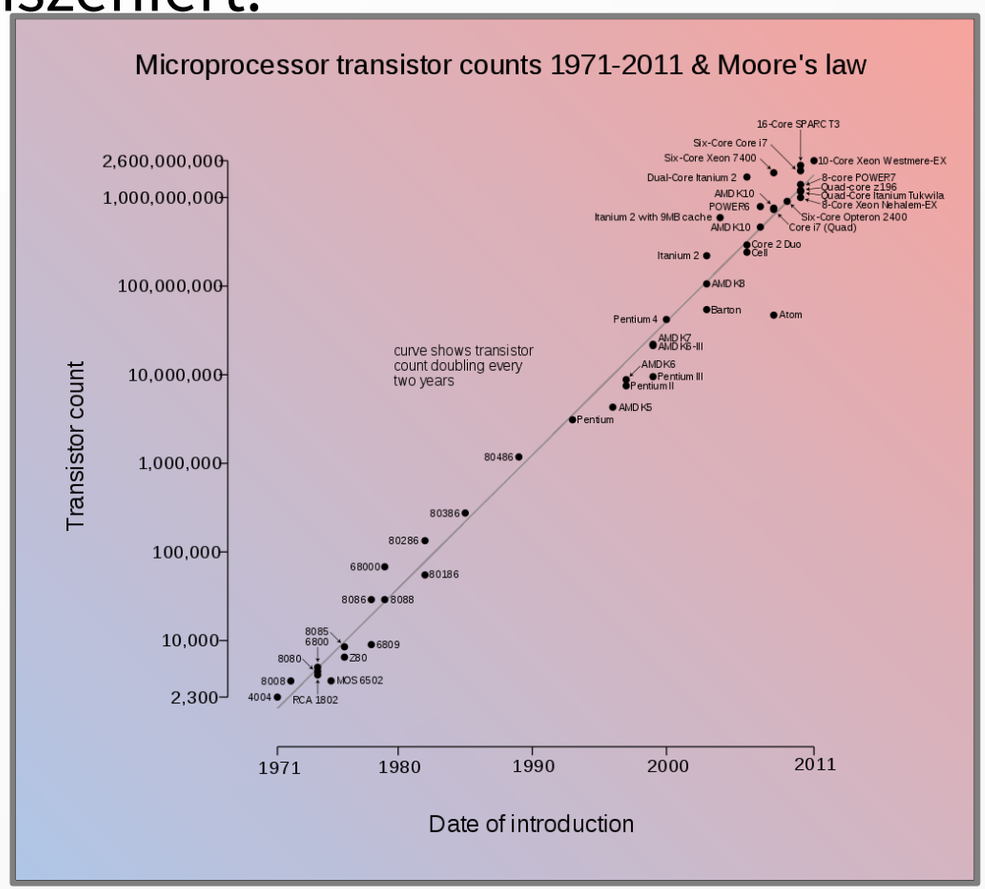
CPU: Zentrale Prozesseinheit

- Die **CPU** ist die zentrale Programmablauf Prozesseinheit
- wird ergänzt durch die Peripherie wie **Speicher** (**RAM** / **ROM**, **FLASH**) und den **Ein-/Ausgabe-Schnittstellen**



CPU: X86, 68000, ARM Cortex, RISCV, …

- Während eine Vielzahl anderer Technologien fast unbemerkt voran evoluierten, hat die Gaming-Industrie ein Wettrennen um die schnellsten **CPUs** und Grafik-Chips inszeniert.
- Für eine lange Zeit galt dabei das **Mooresche Gesetz** der **Transistoren pro CPU**.
- Die CPU wird weiterhin als **Universalprozessor** verstanden, auch wenn **SIMD-** und **DSP-Funktionen** hinzugekommen sind.



PC-CPU: Leistungsboliden und Stromfresser

- Auf Grund der immer komplexeren Spiele-Software wurden die CPUs als **Hochleistungs-Chips** gezüchtet.
- Mit dem Aufkommen immer kleinerer Notebooks musste die **Energiebilanz** verstärkt berücksichtigt werden.
- Während die **INTEL** und **AMD** Personal-Computer CPUs immer noch viel elektrische Energie in Wärme umsetzten, schoben sich die **viel effizienteren ARM Cortex-Cores** auf den Markt und ermöglichten so erst Smartphones, Tablets und anderen Embedded Geräten zum Siegeszug.

. . . gegenüber effizienteren ARM Cortex Kernen

- Auf Grund einer sehr guten Compiler-Unterstützung konnten quelloffene Betriebssysteme für die Cortex-Chips leicht portiert werden. Diese dominieren seit dieser Zeit den **Embedded-Bereich** signifikant.
- Im Gegensatz dazu ignorierten die großen PC-CPU Hersteller eine lange Zeit diese neue Generation der **energieeffizienten** CPUs.
- Inzwischen sind die Marktanteile deutlich in Richtung **effizientere** CPU-Systemen verschoben.

DSP: Signalprozessor und die ***MAC***-Anweisung

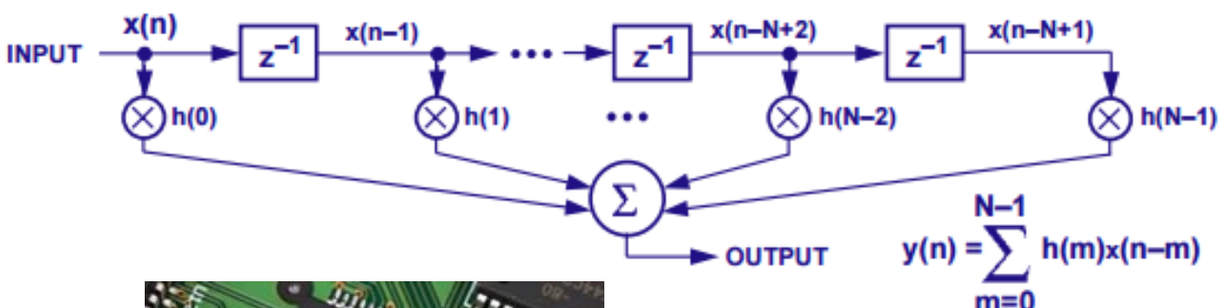
- Während den **Universal-CPUs** immer höhere Taktraten aufgezwungen wurden, entstand ein anderer Typ von Prozessor: der **hoch spezialisierte DSP** (Digital-Signal-Processor).
- Dieser ist hauptsächlich dafür optimiert, **typische Berechnungen der Signalverarbeitung** schnell und effizient zu ermöglichen.
- Eine Vielzahl von **Signalverarbeitungstechnologien** erfordern die Multiplikation von Signalvektoren mit deren Filter-Koeffizienten, um dann anschließend aufsummiert zu werden (**MAC: Multiply-and-Add Coefficient**).

Der große Plan: Digitaler-Signal-Prozessor (DSP)

Digitale Signal Prozessoren (DSP)
Ist prinzipiell einer CPU ähnlich, jedoch erheblich aufwändigere Rechenfunktionenen bei einer geringen benötigten Taktanzahl.
Vorteil: Digitale Signalverarbeitung wie Filterung, Signalfaltung, komplexer Zahlen-raum werden effizient und schnell erledigt. Viele Operationen werden doppelt oder mehrfach parallel ausgeführt (SIMD).
Nachteil: Echtzeitfähigkeit ist dennoch, je nach Anwendung, ein Problem.

Signalverarbeitung: DSP (herkömmlicher Ansatz)

- Digitalsignale: **hoher Datendurchsatz**, DSP nutzbar falls Latenzzeiten weniger kritisch.
- Lösung: Digitaler Signalprozessor mit **spezialisierten** und **parallel ablaufenden Rechenoperationen**.
- **Zeitkritische Anwendungen** aber immer noch **problematisch**.



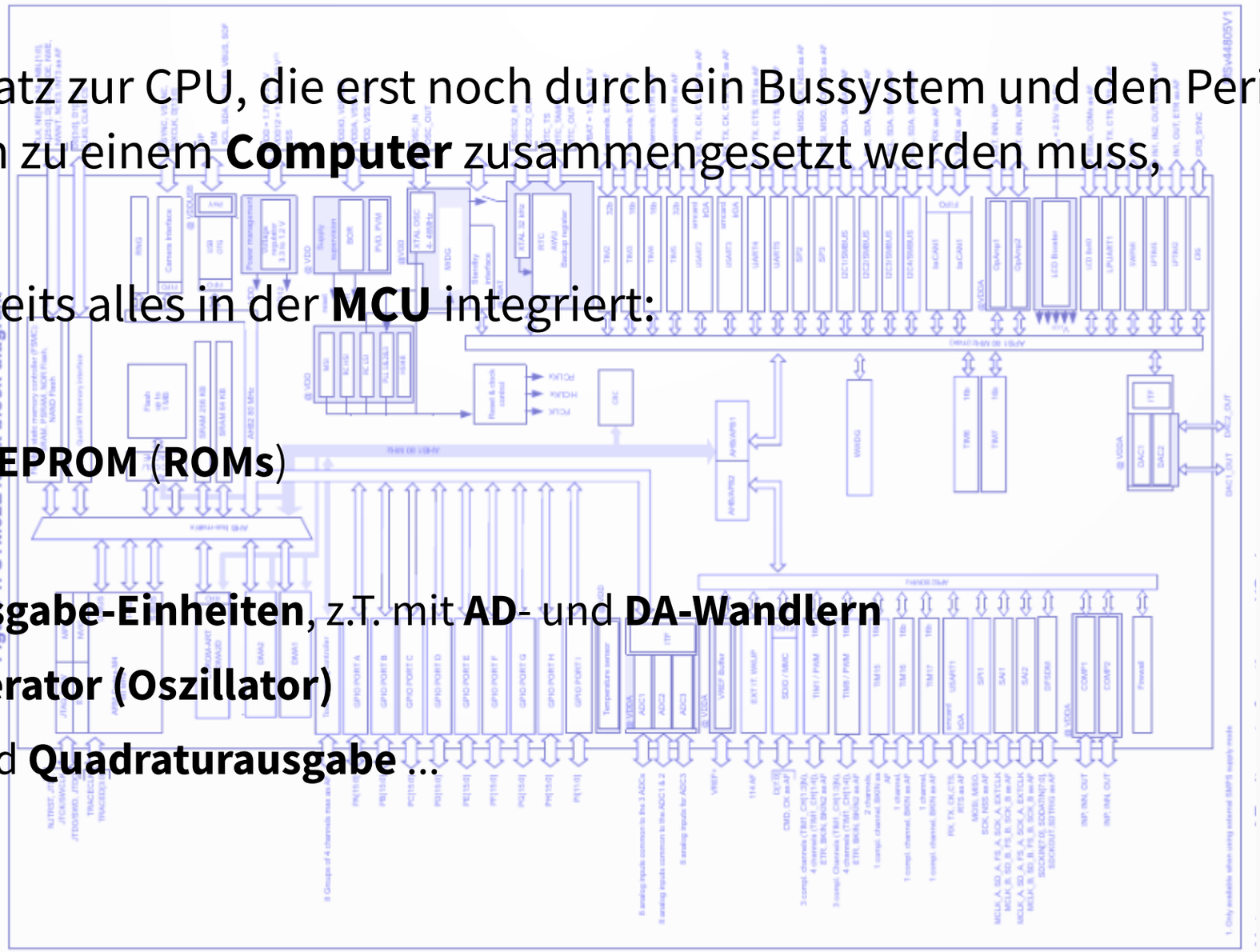
Der große Plan: MCU (Mikrocomputer)

Mikro-Computer (MCU)
Mit der Integration von PROM, EPROM,EEPROM, FLASH sowie RAM-Speicher ist ein Mikrocomputer realisiert.
Hersteller überbieten sich mit unterschiedlichen Zugaben wie Timers, I/O-Leitungen, Interrupt- und Triggermöglichkeiten, Analog-Digital Wandler, PWM-Ausgabe und vieles mehr.
Nach und nach wurde Energieaufwand geringer, Leistungsfähigkeit höher, die Preise dank großer Stückzahlen fielen ins Bodenlose.
Mehr und mehr waren die MCUs auch im System (ISP) programmierbar ohne spezielle Programmiergeräte mehr.
Anzahl verfügbarer Register stiegen an, arithmetische Operationen wurden immer leistungsfähiger bei reduzierter Clockanzahl die dazu benötigt wurde.
Der derzeitige Trend geht deutlich von 8 Bit zu 32 Bit Wortbreite.
Nachteil: Die Reaktion auf Zustandsänderungen ist langsam als bei Logikschaltungen

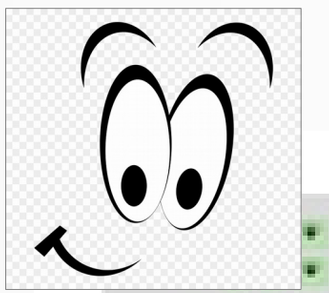
MCU: CPU, Speicher und Peripherie in einem IC

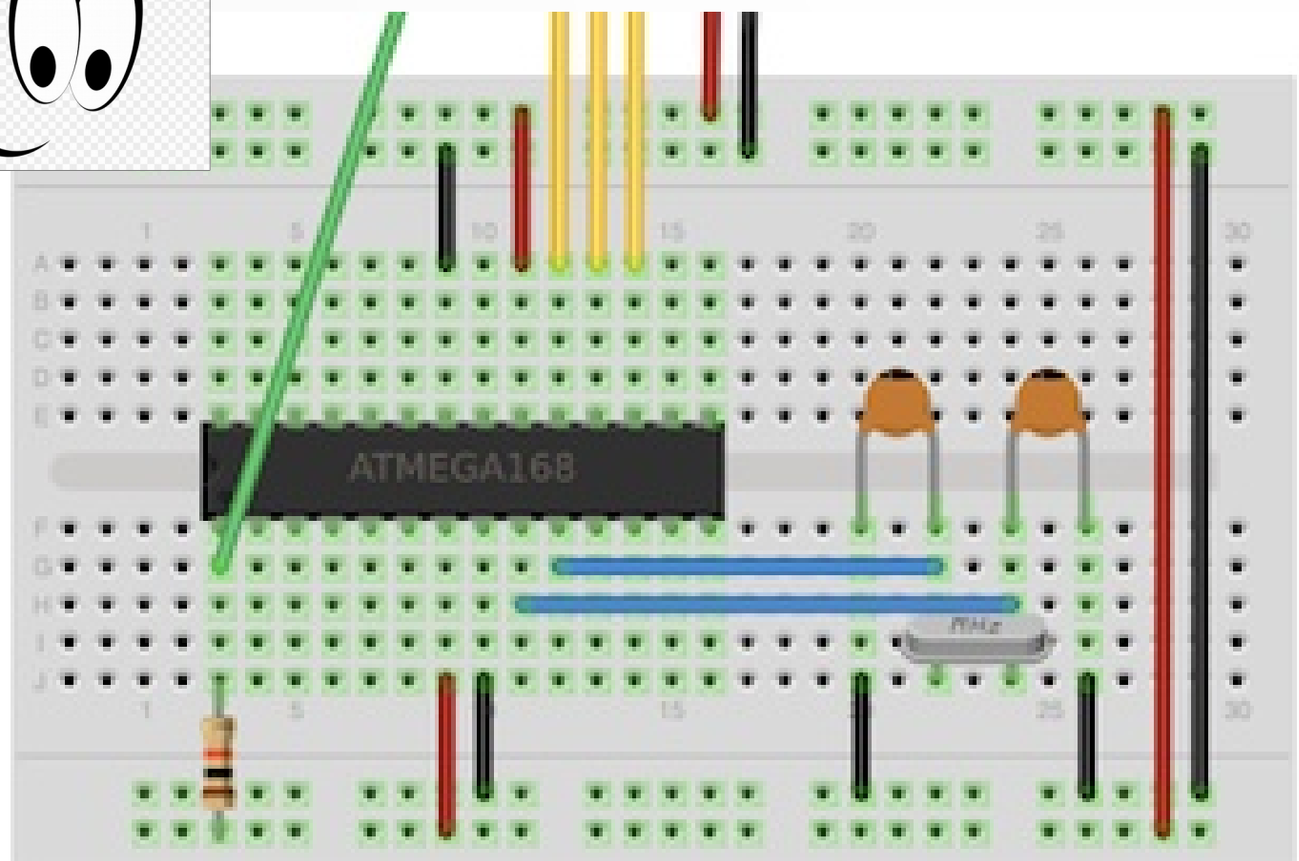
- Im Gegensatz zur CPU, die erst noch durch ein Bussystem und den Peripherie-Bausteinen zu einem **Computer** zusammengesetzt werden muss,

- Ist dies bereits alles in der **MCU** integriert:
  - **CPU**
  - **FLASH**, **EEPROM** (**ROMs**)
  - **RAM**
  - **Ein- / Ausgabe-Einheiten**, z.T. mit **AD**- und **DA-Wandlern**
  - **Taktgenerator (Oszillator)**
  - **PWM-** und **Quadraturausgabe** ...



MCU: *Minimal* beschaltung





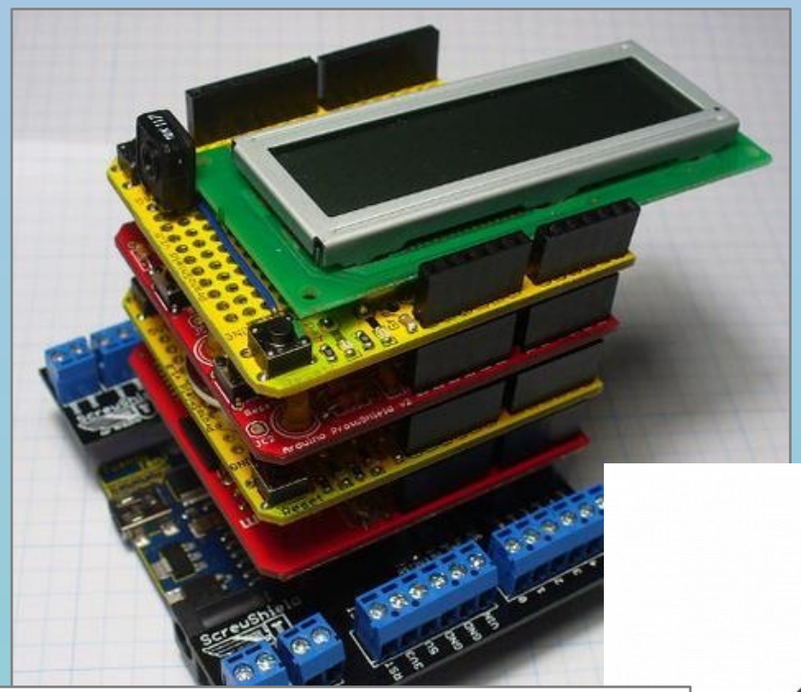
- **MCU (hier: Atmega 168)**
- **Widerstand für RESET-Leitung**
- **Kondensator für Vers-Spng.**
- **Abgebildeter Quarz und beide Kondensatoren können entfallen.**
- **Ca. 10.000x wiederbeschreibar. Stückpreis: unter 2,- €**

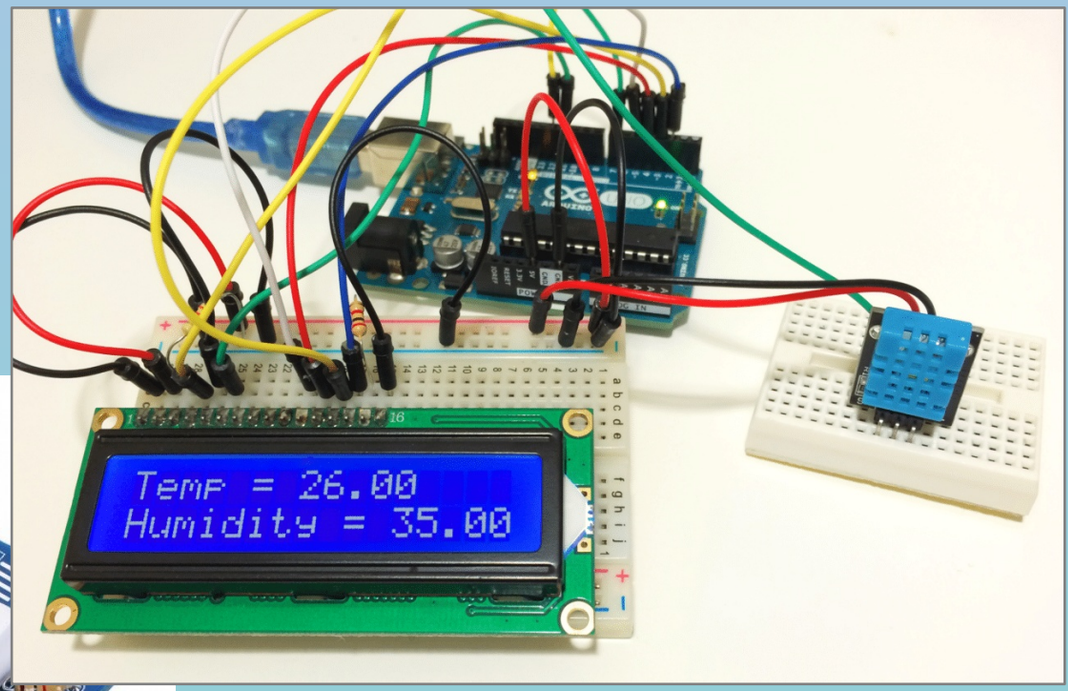
Aktuelle MCUs: Grenzen verschieben †

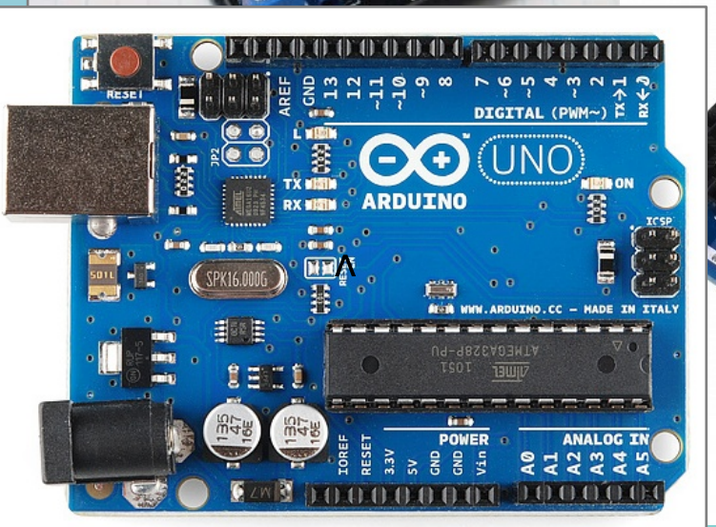
- Immer größere **RAM (328 kB)** / **FLASH (1 MB)** Speicher
- Taktraten in Richtung **100 MHz** und darüber
- Systemreaktionszeiten besser als **1 µs**
- dabei Stromaufnahme unter **1 mA** @ 3,3 Volt auch bei 32 Bit
- Kostenlose und leistungsfähige **Programmierumgebungen**
- **Einfachste Beschaltung** auch für Anfänger geeignet

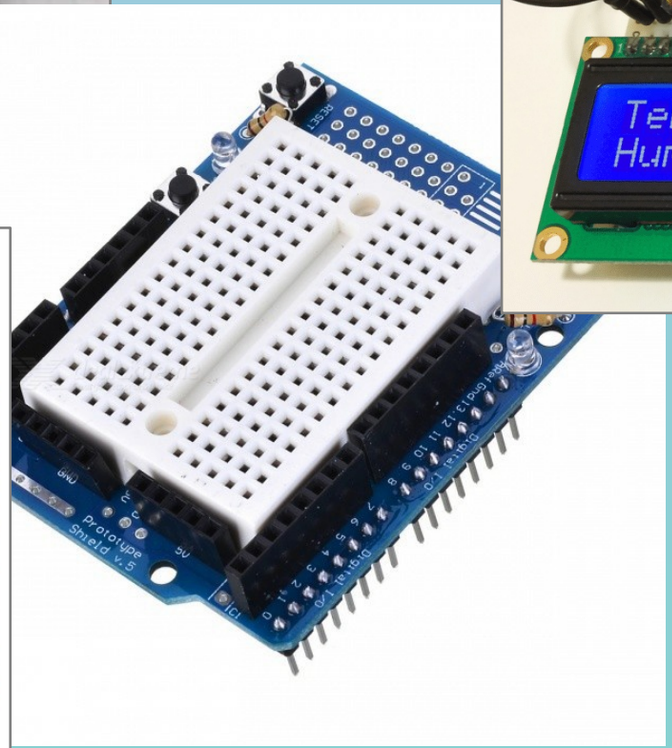
† hier am Beispiel eines STM32L476

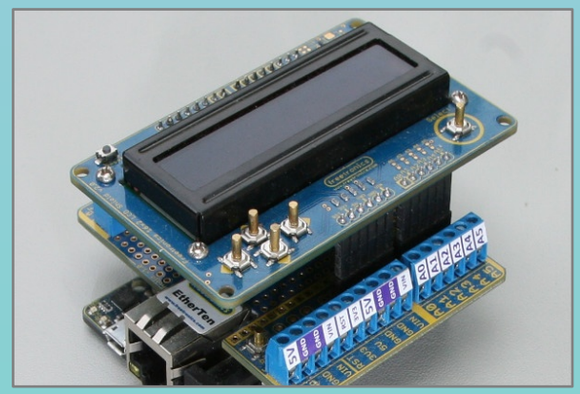
MCU: Arduino Plattform für *Einsteiger* …



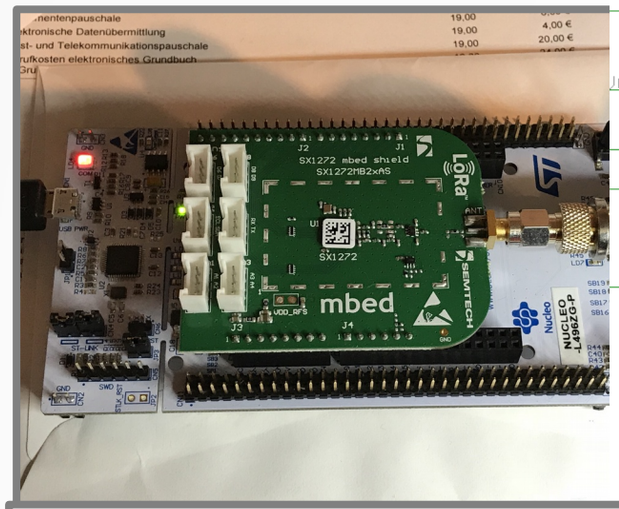


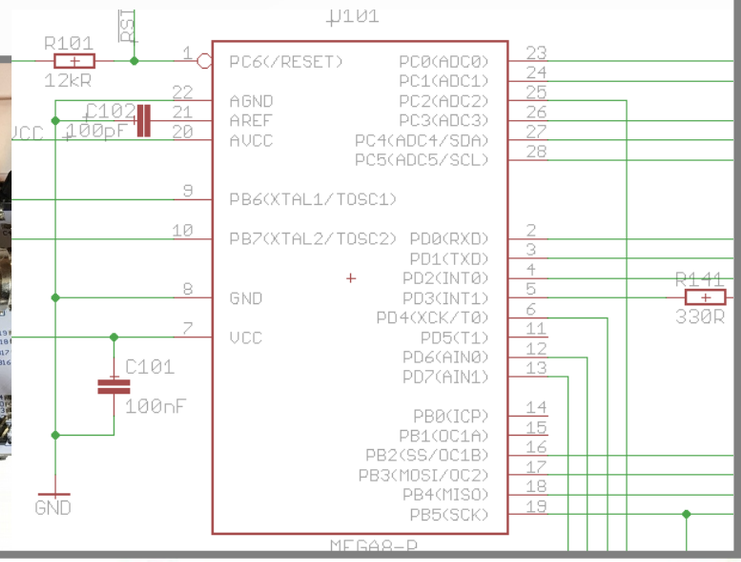


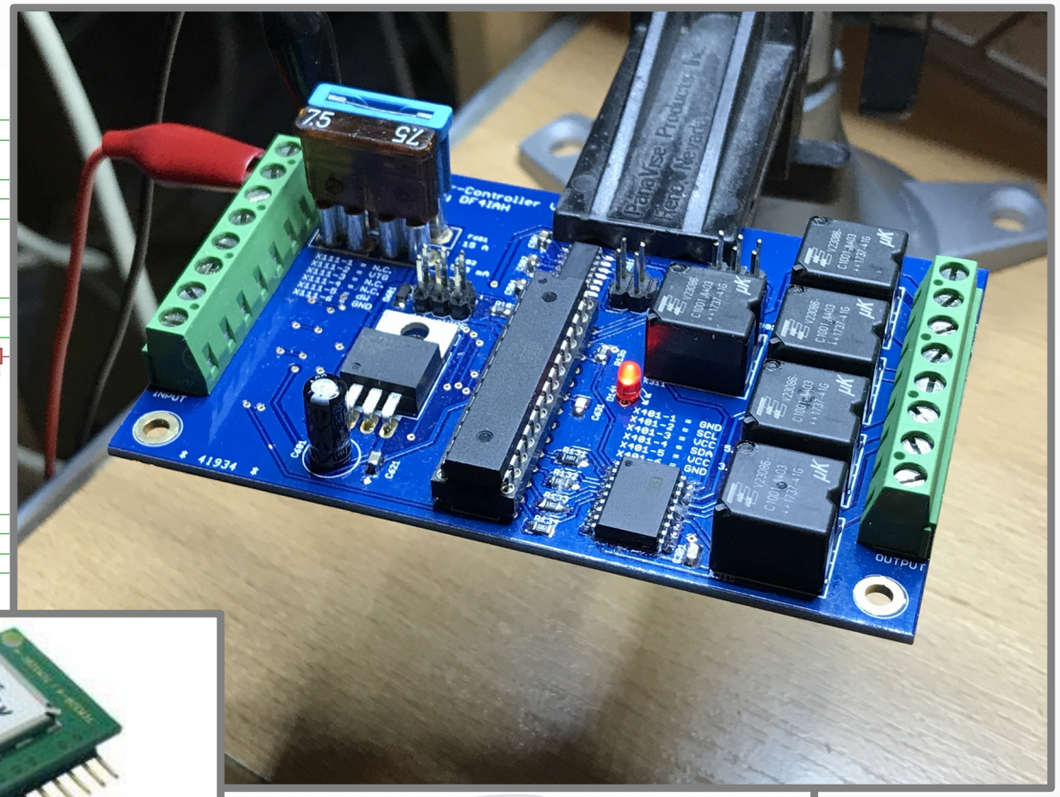


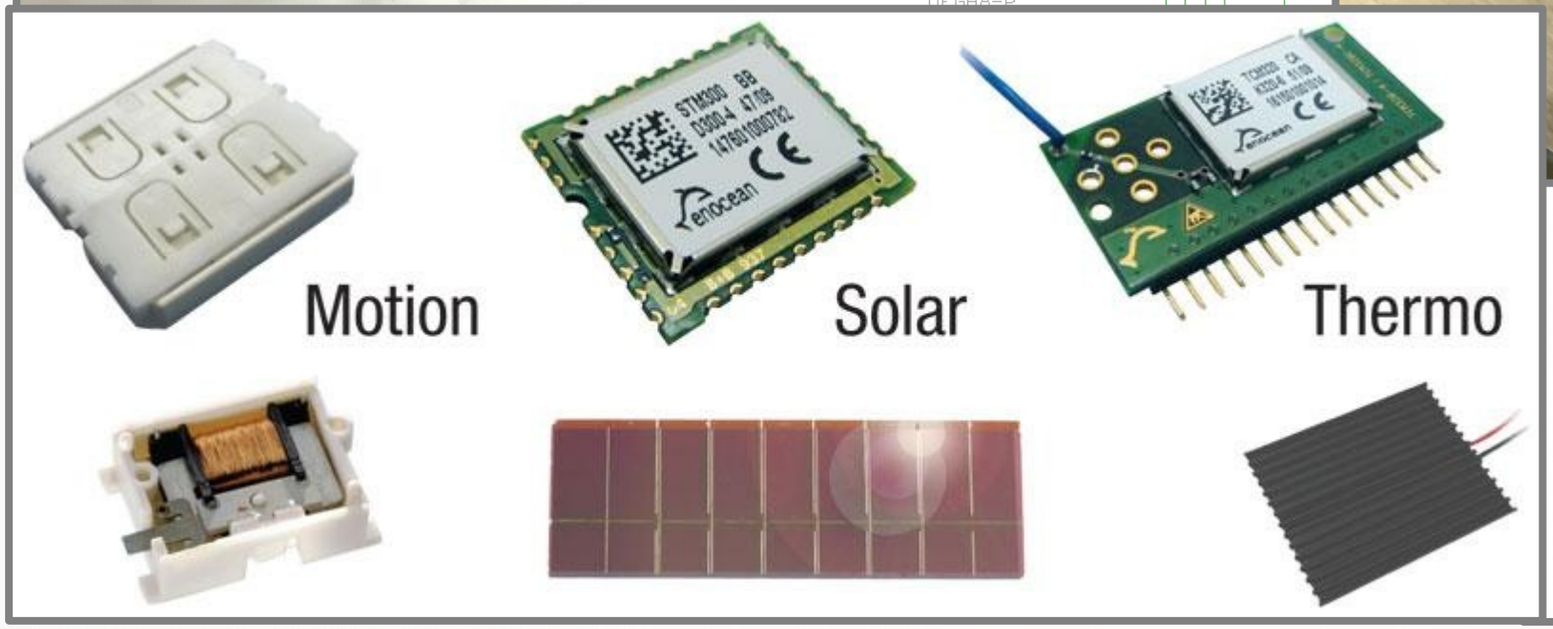


MCU: … und für den *Profi*







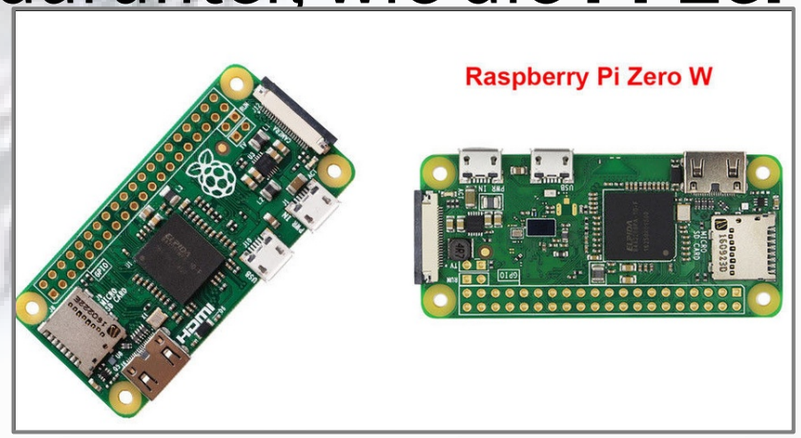


SoC: System-on-Chip

- Vergleichbar der Idee mit einer MCU einen (kleinen) Prozessor und seiner zugehörigen Peripherie **in einem IC zu vereinen**,
- … ist der **SoC** eine **kompatible PC-Architektur in einem IC-Gehäuse**.
- Im Gegensatz zum einem eigenständigen Personal-Computer, der zwingend eine INTEL / AMD-CPU vorsieht, ist bei einem **SoC** viel eher eine **energiesparende ARM Cortex CPU** integraler Bestand-teil eines SoC.
- Dafür sind bei einem SoC der **USB-**, der **PCI-Express-** und der **Ethernet-**Bus, **Audio-** und **Video-Schnittstellen** in aller Regel gemeinsamer Bestandteil.

SoC: *Raspberry & Banana-Pi*

- Was der ***Arduino*** bei der MCU ist, ist der ***Raspberry-Pi*** bei den PC-**SoC**.
- Aufgrund hoher Stückzahlen und preiswerter Fertigung werden **Raspberry-Pi 3 B** Systeme für rund **50,- €** gehandelt.
- Je nach Ausstattung aber auch weit darunter, wie die **Pi-Zero** Modelle.



Verschiebung der Abgrenzung zwischen TTL-Logik und der µP-Architekturen

	Logikgatter	µP
Komplexität	gering	hoch integriert
Signallaufzeit	gering	Taktverzögerung
Architektur	parallel	sequentiell
Verwendung	mäßig / aufwändig	einfach
Stückkosten	mittel	gering
Laborkosten	mittel / hoch	einfach / mittel
Fehlersuche	mäßig	einfach / mittel
Fehlerkorrektur	aufwändig	einfach
Einarbeitungszeit	einfach / mittel	einfach / mittel

2. Betrachtung: TTL ↔ Rekonf. Logik (CPLD, FPGA)

- Manchmal sind diese **Eigenschaften** für eine Zielanwendung nötig:

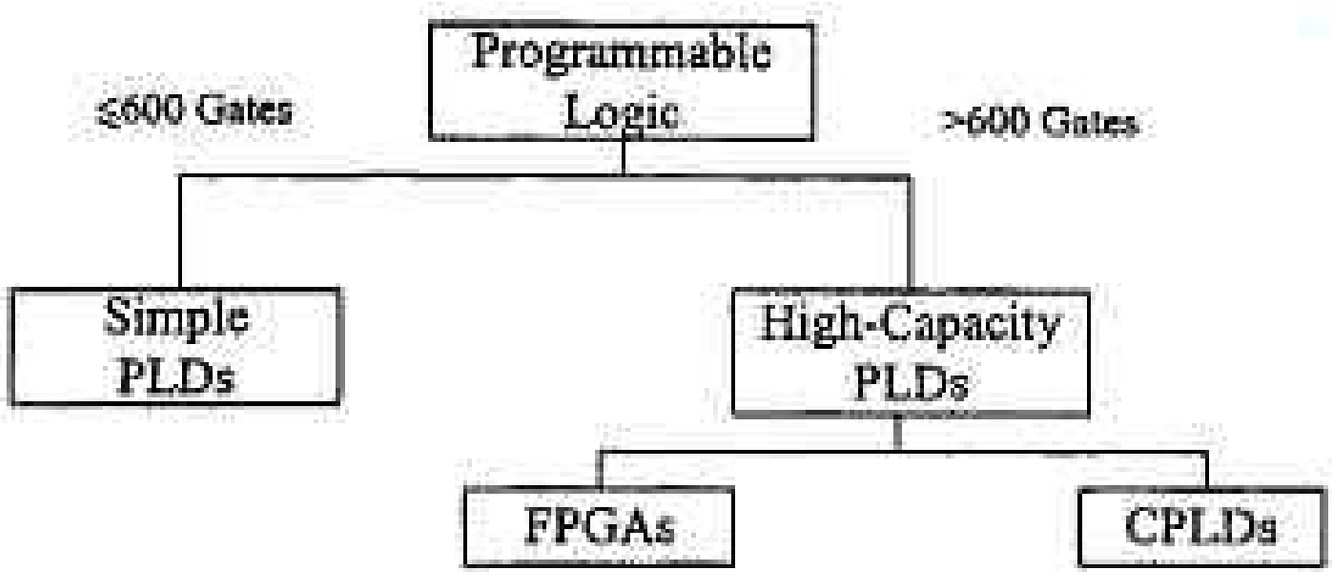
… dann heißt die Lösung: **Rekonfigurierbare Logik**

- **Durchlaufzeiten** müssen **kurz** sein
- **Durchlaufzeiten** müssen **deterministisch** sein
- Die Anwendung ist **erheblich** zu **parallelisieren**
- Die Zielschaltung soll in ein **ASIC**† überführt werden

† ASIC: Application Specific Integrated Circuit

Der große Plan: Rekonfigurierbare Logik

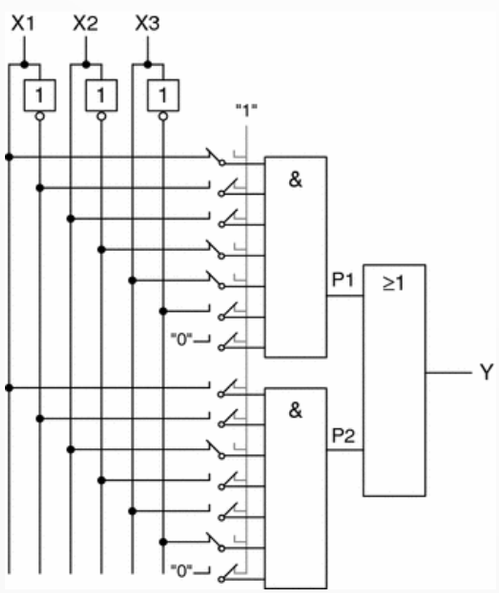
CPLD Complex programmable Logic Device
Vom Grundkonzept ein vergrößertes PAL mit weitaus mehr Register-Transistor-Logik (RTL)
Vorteile: Rekonfigurierbarkeit und schnelle Taktdurchlaufzeiten. Unterschiedliche Routing-varianten beeinflussen Durchlaufzeiten nur gering, da innerer Aufbau sehr gleichförmig ist.
Bis heute im lebhaften Wettbewerb mit FPGA und ASIC.

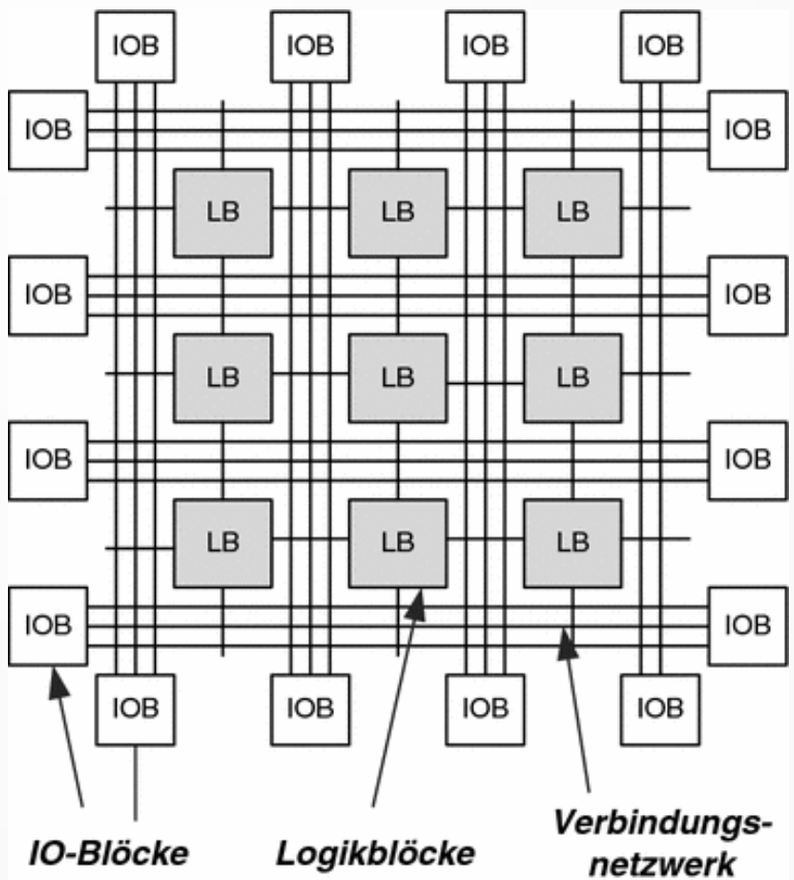


FPGA Field programmable Gate Arrays
In Felder aufgeteilte Logik- und Spezial-anwendungen wie DSP-Units, RAM/ROM-Units, Schieberegister, PLLs und Clock-Drivers, um nur einige zu nennen.
Vorteile: Auf Grund der Vielzahl von speziallierter Units ist die Packungsdichte enorm hoch. Die nutzbaren Frequenzen können um die 500 MHz liegen. Das dickste Plus dieser Architektur ist die (beliebige) Parallelisierungs-möglichkeit und Rekonfigurierbarkeit - auch zur Laufzeit. Beliebige Soft-CPUs lassen sich integrieren. Ob ein 6502, ein Z80, 8080 oder ARM, alles kann genutzt werden. Oder ein klassischer FSM-Zustandsautomt realisieren.
Nachteile: Der Preis, die Leistungsaufnahme sind schlechter als beim ASIC. Ansonsten ist das FPGA das Universal-Pferd für alle Anwendungen mit hoher benöigten Performance.
Ist ein Design im FPGA erfolgreich in Betrieb genommen worden, so kann leicht eine Adaption auf ein ASIC erfolgen, welches alle unbenutzen Einheiten wegrationalisiert und somit preiswert macht.

CPLD: Complex Programmable Logic Device

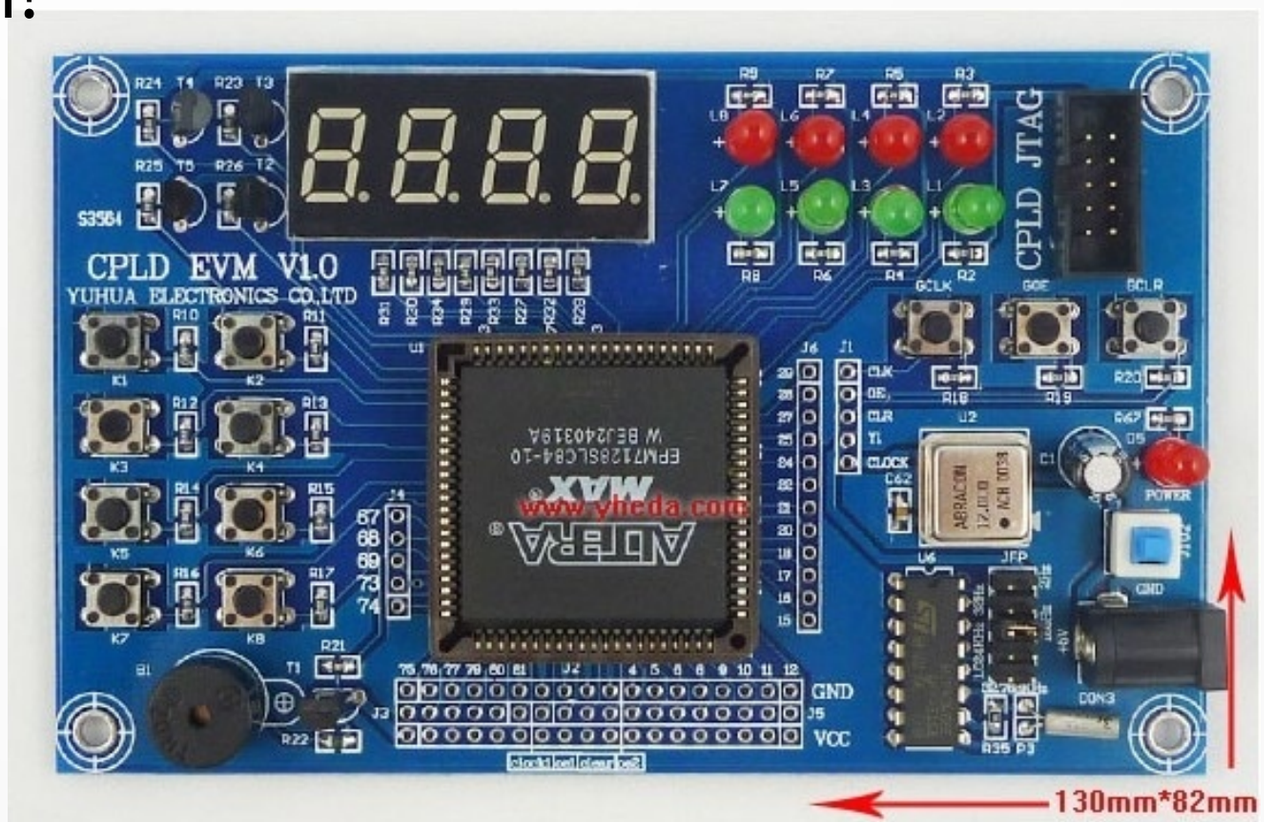
- Enthält Logikstrukturen, die entweder aktiviert oder deaktivert werden können. Konkret: ver-**oder**te **Und**-Terme.
- Alles wird auf diese Logikterme abgebildet und über Cross-bars zu den I/O-Blöcken geführt:
- Rückkopplung ermöglicht Kippglieder (Flip-Flops).





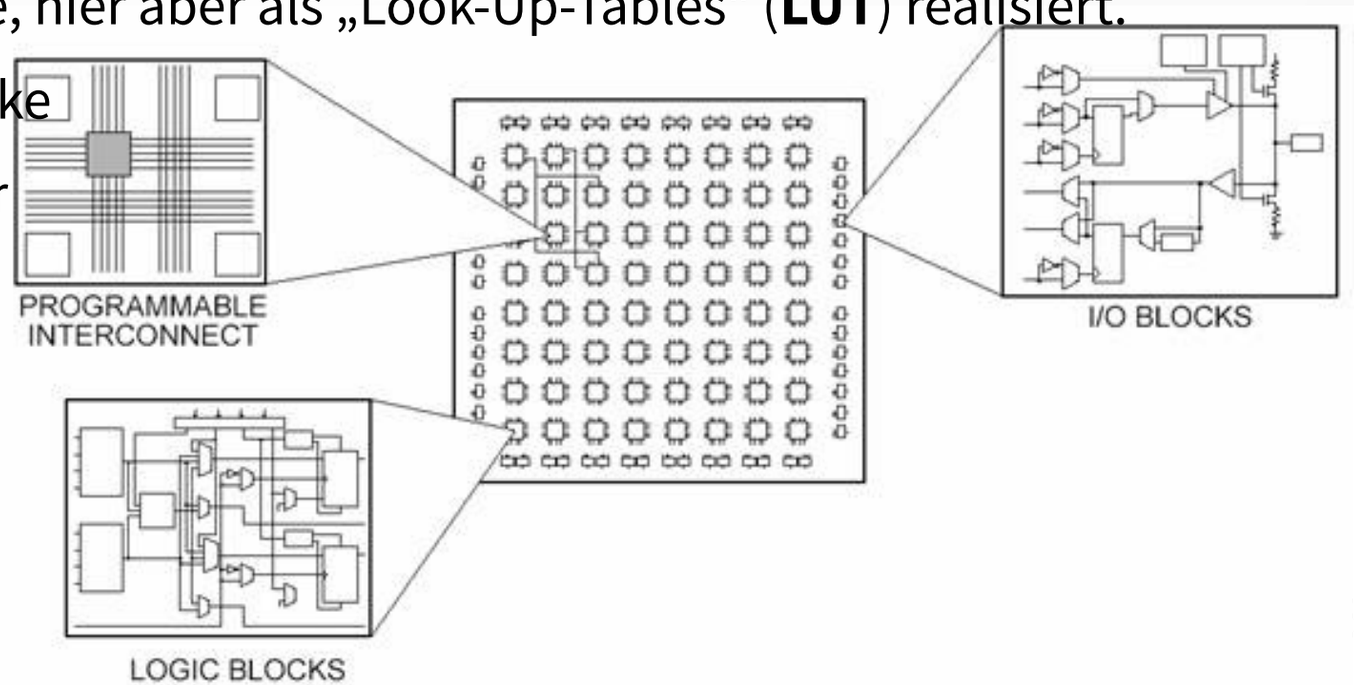
CPLD: Entwicklungsboards

- Selbt mit **CPLDs** ist eine erstaunlich **hohe Komplexität** der Aufgabenerfüllung möglich, obwohl keine spezialisierte Module darin vorliegen!
- Alle Durchlaufzeiten sind streng **deterministisch**, zudem auch **minimal** gegenüber anderen Lösungen.



FPGA: Field Programmable Gate Array

- Von der Bezeichnung nicht irritieren lassen, auch ein CPLD ist „im Feld programmierbar“.
- Was jedoch ein FPGA auszeichnet sind seine **spezialisierten Funktionsblöcke**:
  - „Natürlich“ die Logik-Blöcke, hier aber als „Look-Up-Tables“ (**LUT**) realisiert.
  - DSP Recheneinheiten / Blöcke
  - ROM / RAM / Schieberegister
  - PLL und Phasenschieber
  - AD-Wandler, Bussysteme
  - etc ...



FPGA: höchste Integration, größte Flexibillität

- Praktisch besteht ein FPGA aus einem großen RAM-Speicher, welcher beim Booten (meist) von einem externen seriellen Speicher eingelesen wird.
- Ein FPGA lädt in ca. 1 Sekunde seinen RAM und ist dann funktionsfähig. Das Bitmuster stellt dann entsprechende Funktionsblöcke aktiv.

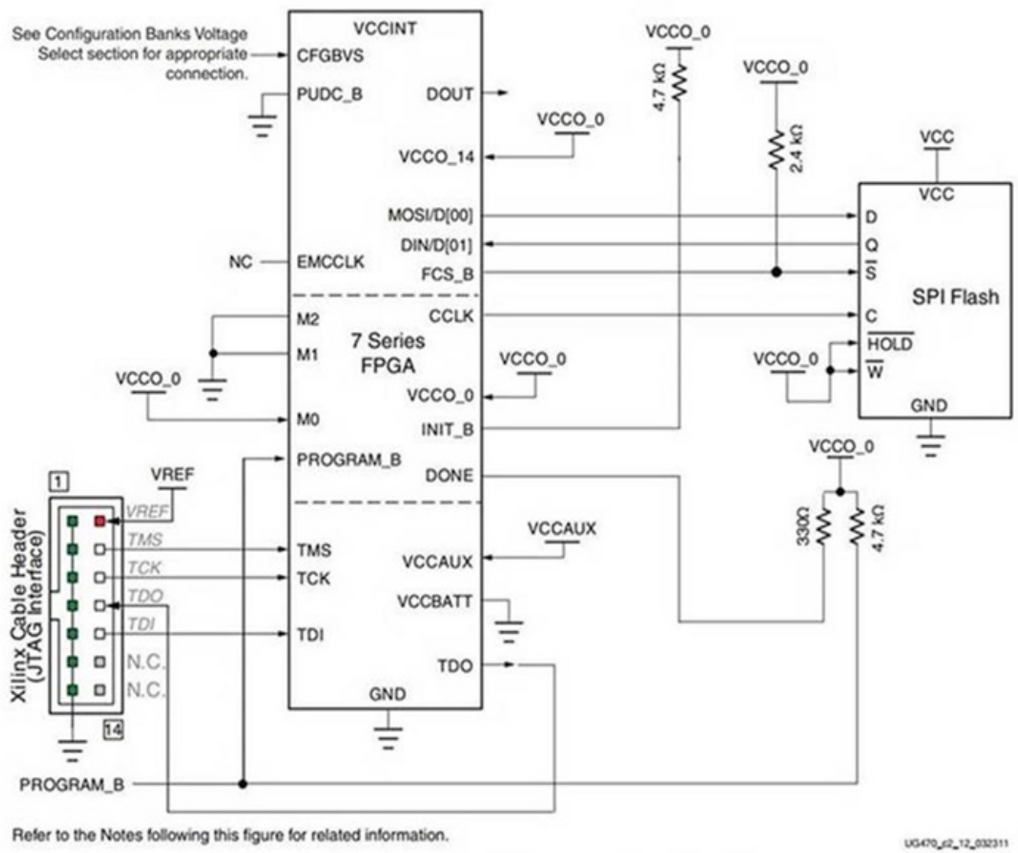


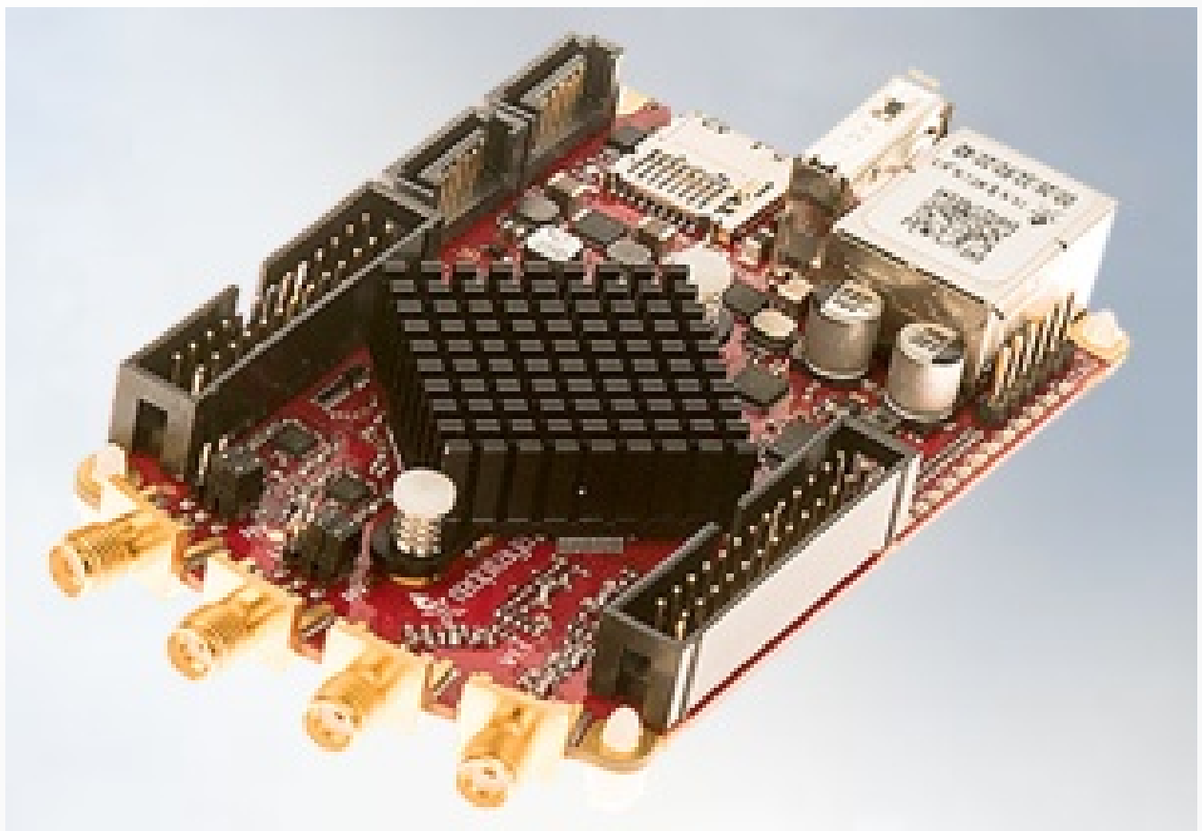
Figure 2-12: **7 Series FPGA SPI x1/x2 Configuration Interface**

FPGA & CPU in einem IC: geht das?

- Der Marktführer XILINX bietet beispielsweise eine solche Lösung an: „**Zynq**“

- Die Aufgabe wird aufgeteilt in einen **sequentiellen Ablauf → CPU**, alles andere macht das **FPGA**. Die CPU spricht per **Registerschreiben** und **-lesen** mit dem FPGA.
- Einziger Nachteil: Zynq > 50€

- Zynq-Implementierung:



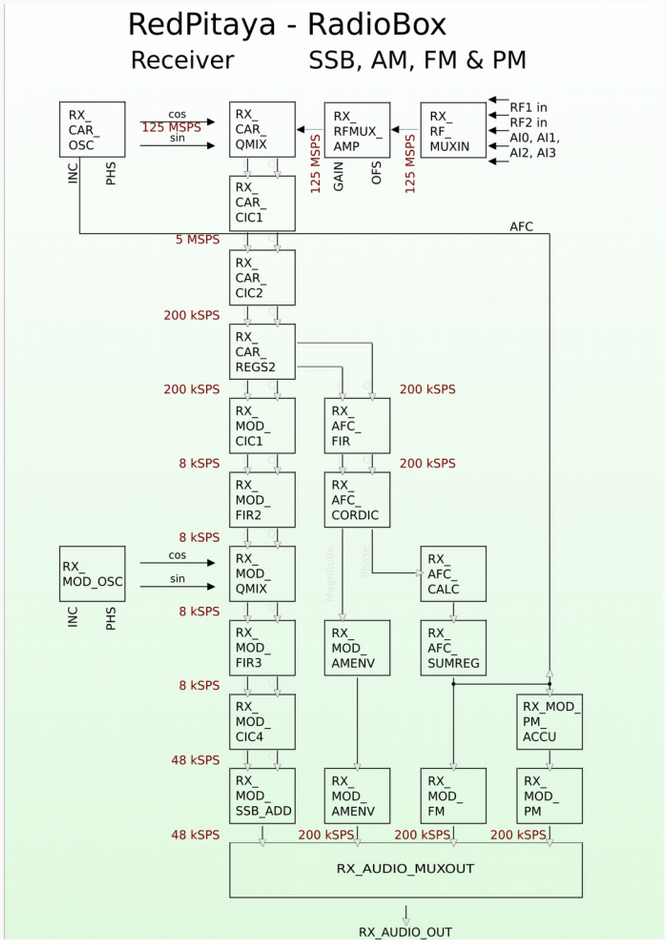
Beispiel: RedPitaya

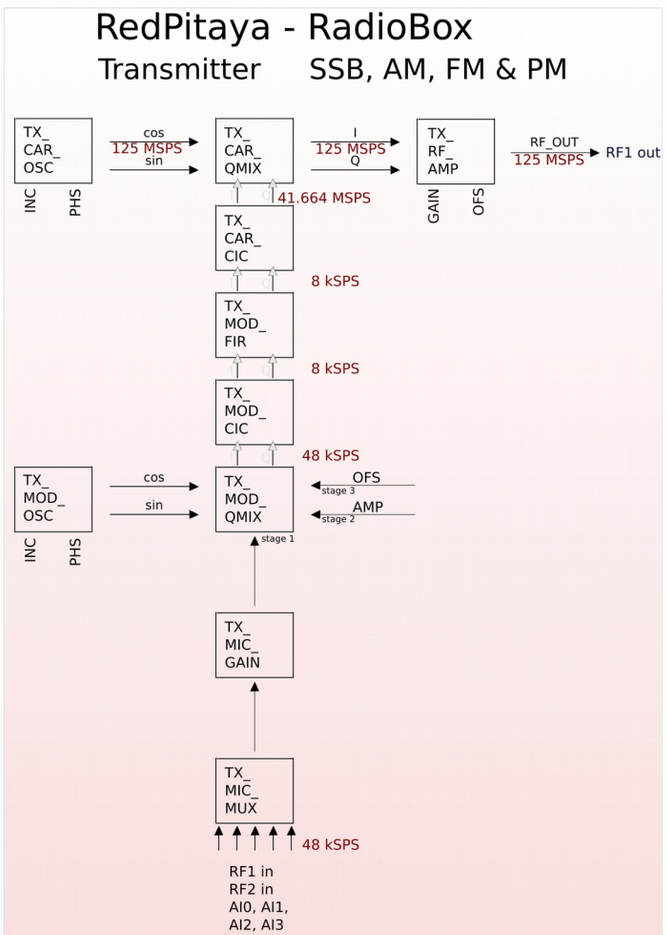
Was macht ein **FPGA / Zynq** so begehrt?

- Die Design-Eingabe kann auf verschiedene Arten erfolgen. Ob als VHDL / Verilog / Chisel – also per **Hochsprache**,
- Oder per MATLAB/Simulink Realisierung des **mathematischen Modells**. Wie auch viele weitere Konfigurationshilfen (DSP für FIR/IIR).

- Die Entwicklungsumgebungen sind inzwischen bedingt kostenfrei nutzbar. Viele Tutorials sind dazu verfügbar. Eval-Boards sind kostengünstig erhältlich.
- Parallelisierung bis die Resourcen ausgehen: da lassen sich etliche SDRs in einem kleinen FPGA realisieren. Mit einem großen FPGA könnte man den CW-Funkverkehr aller Bänder zeitgleich dekodieren ...

FPGA-Beispiel: SDR Sender und Empfänger





- RadioBox ist eine SDR-Implementierung auf einem Xilinx **Zynq Z-7010** (Red Pitaya).
- Der **Zynq Z-7010** ist der kleinste der Serie:
  - 28.000 Logik-Zellen
  - 2,1 MBit Block RAM
  - 80 DSP Slices
  - Max. 100 I/O Pins verfügbar
- … und? Es waren erst ca. 60% des FPGAs ausgenutzt!
- Sobald per Software (ARM-CPU) die Register gesetzt sind, arbeitet das FPGA völlig autark die SDR-Anwendung ab.

Mit dem **FPGA** spielen?

- Es existieren zahlreiche Eval-Boards und Lern-Boards
- auch **Adurino-Header** oder **Raspberry-Header** kompatible
- Nutze die **FPGA-Leistung** mit vielen erhältlichen **Shields!**
- AVNET und Trendz-electronic bieten solche Boards an.

Zynq Soft Propeller

ZynqBerry