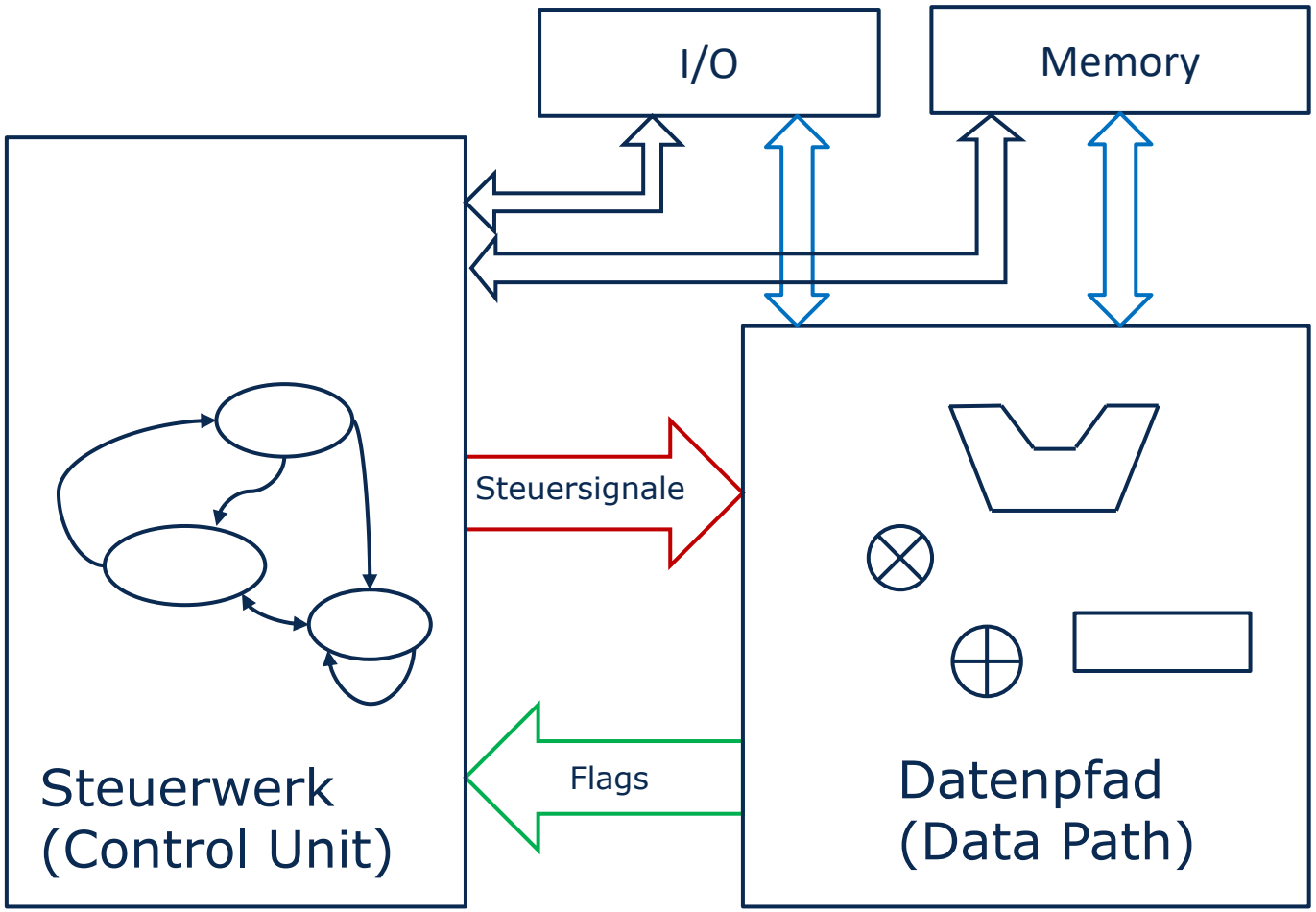
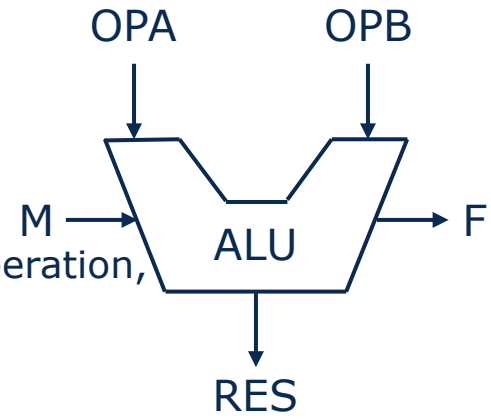
Datenpfade



- Arithmetic Logic Unit (ALU) prozessieren numerische und logische Daten
  - Operanden (OPA, OPB, …)
  - Modus (M): Wahl der Operation, z.B.
    - ADD, SUB, MUL, DIV, SHIFT, AND, OR, …)
  - Ergebnis (RES)
  - Status Flags (F): Zusatzinformation zur durchgeführten Operation, z.B.
    - CARRY, OVERFLOW, SIGN, ZERO



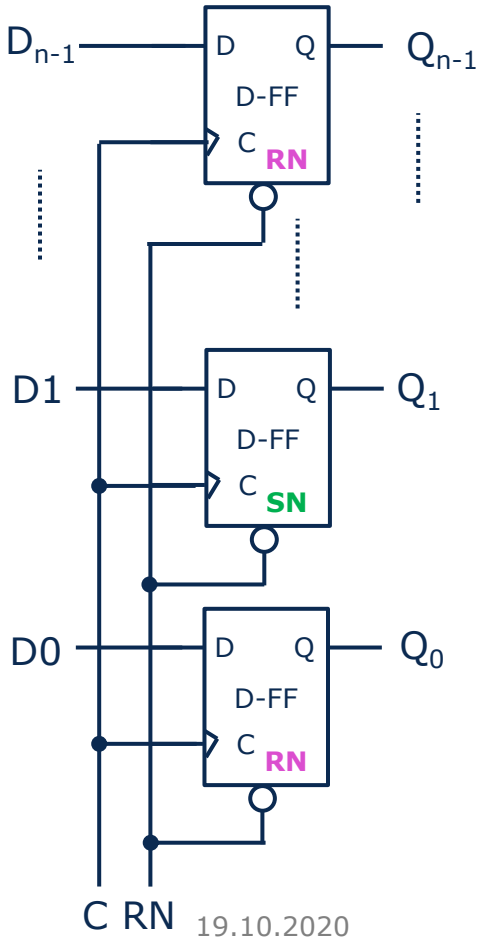
- Datenpfad Baublöcke existieren meist in verschiedenen Varianten (Architekturen)
  - Architekturvarianten (Fläche vs. Schaltgeschwindigkeit)
  - Kombinatorisch oder sequentiell realisiert

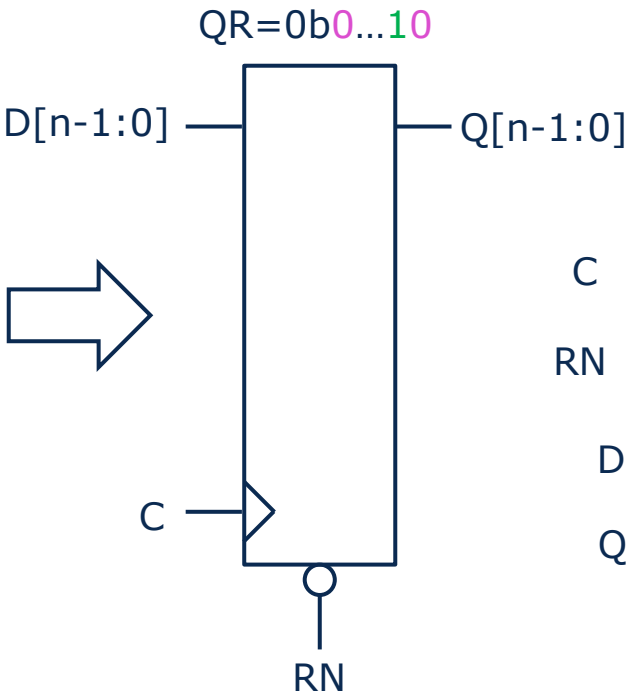
- Baublöcke als Bibliotheken für Synthese-Tools verfügbar (z.B. Synopsys Design Ware)

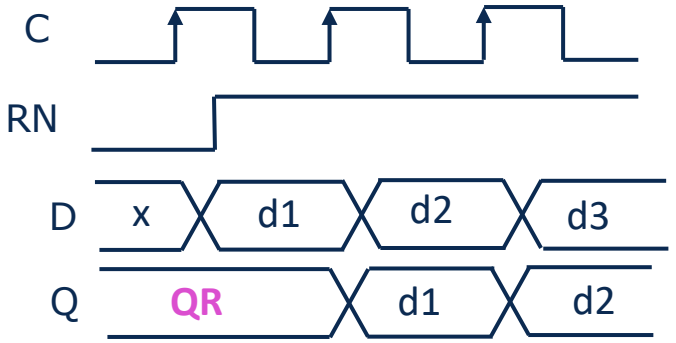
- Im Praktikum stehen dedizierte Baublöcke für die arithmetischen Operationen zu Verfügung

→ Details siehe Anleitung zum Praktikum

- Anordnung aus n-Bit FlipFlops zur Datenspeicherung
- **Reset-Wert** des Registers, festgelegt zur Implementierungszeit durch Auswahl des FlipFlop Typs (**Set**, **Reset**)



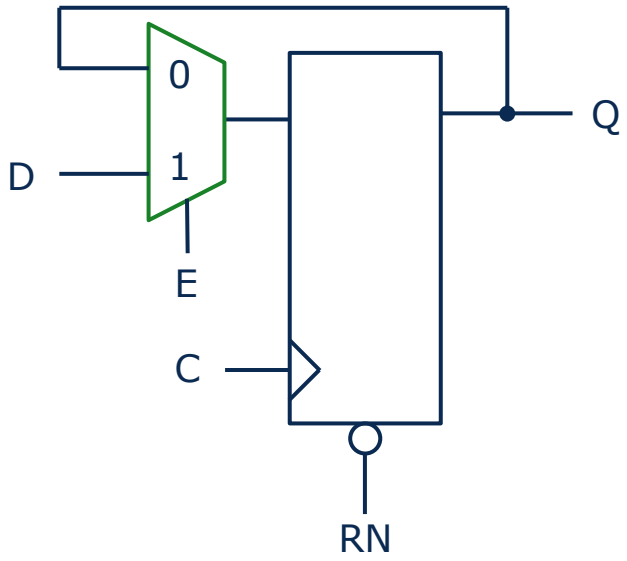


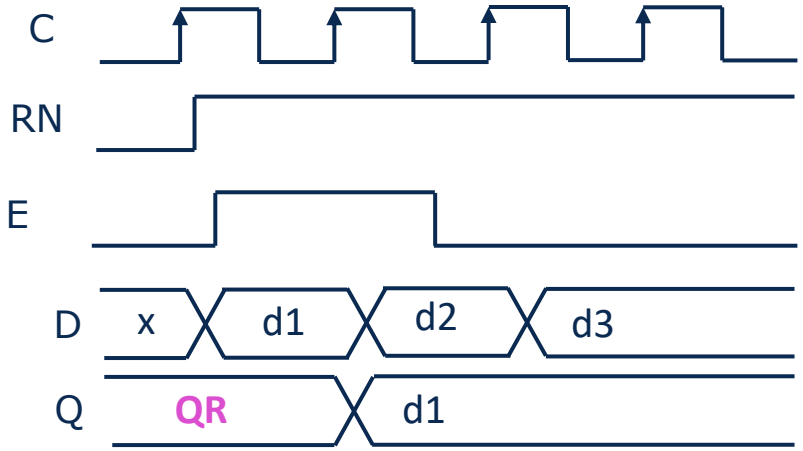


Sehr schneller Zugriff (1 Takt),
sehr kurzes Delay CLK ↑ →Q

Register mit Enable 1

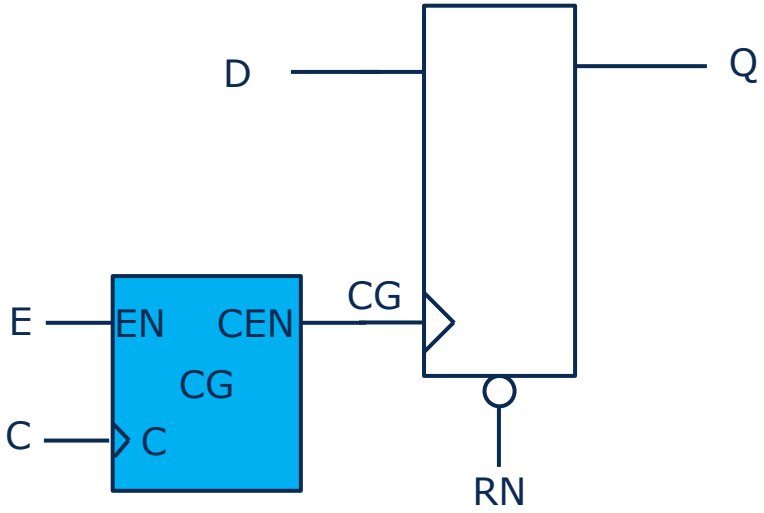
- Bedingtes Schreiben auf das FlipFlop bei E=1
  - $Q_{i+1}=D_i$, wenn E=1
  - $Q_{i+1}=Q_i$, sonst
- Realisierung mit Multiplexer
  - permanentes Schreiben der Daten
  - Auswahl der Daten

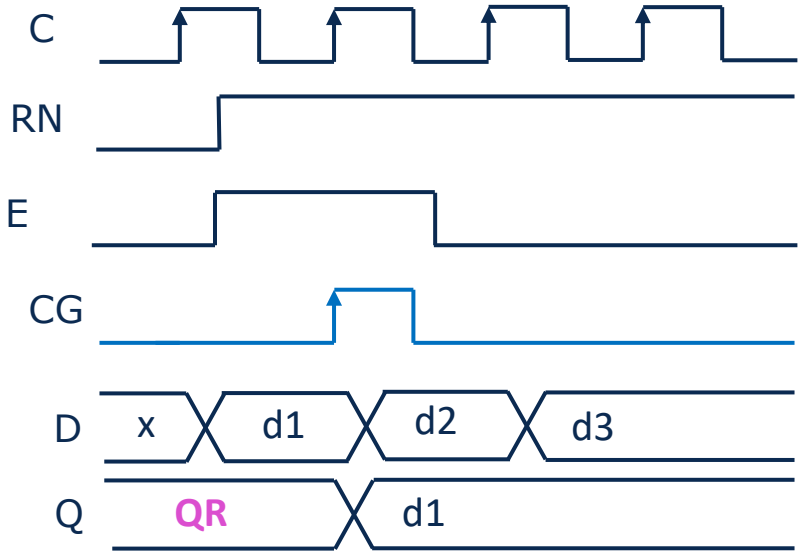




- geringer Schaltungsaufwand
- Kaum Erhöhung der Verlustleistung bei hoher Schreibaktivität

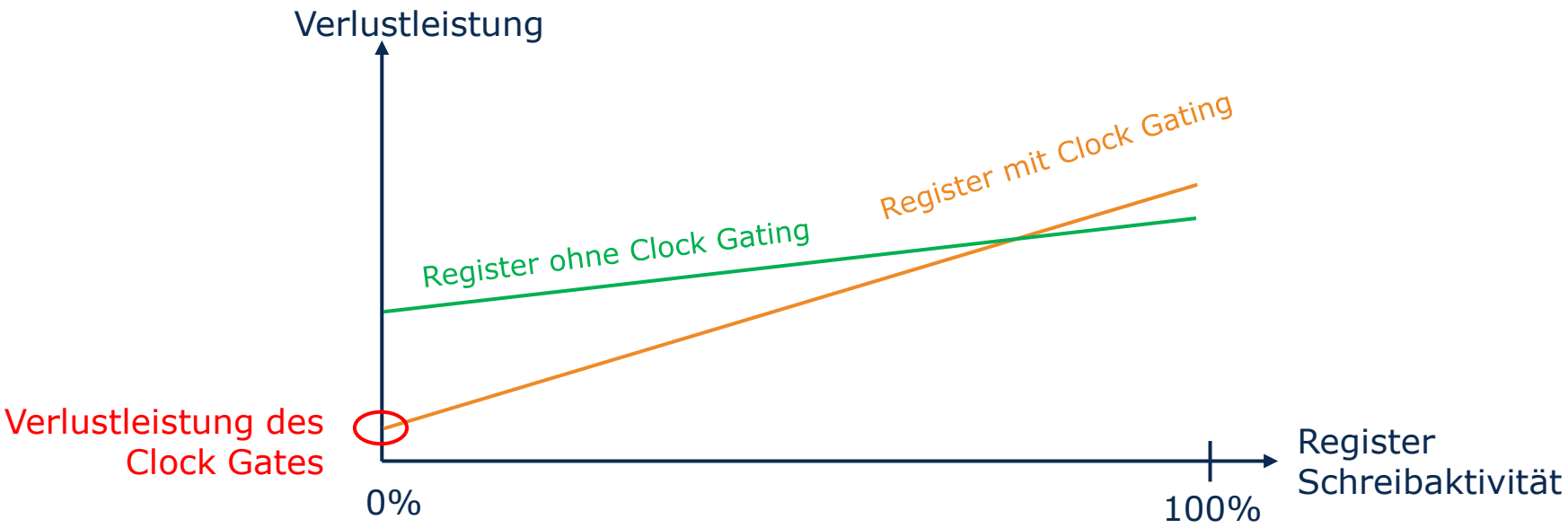
- Realisierung mit Clock Gate
  - Selektives Takten des Registers





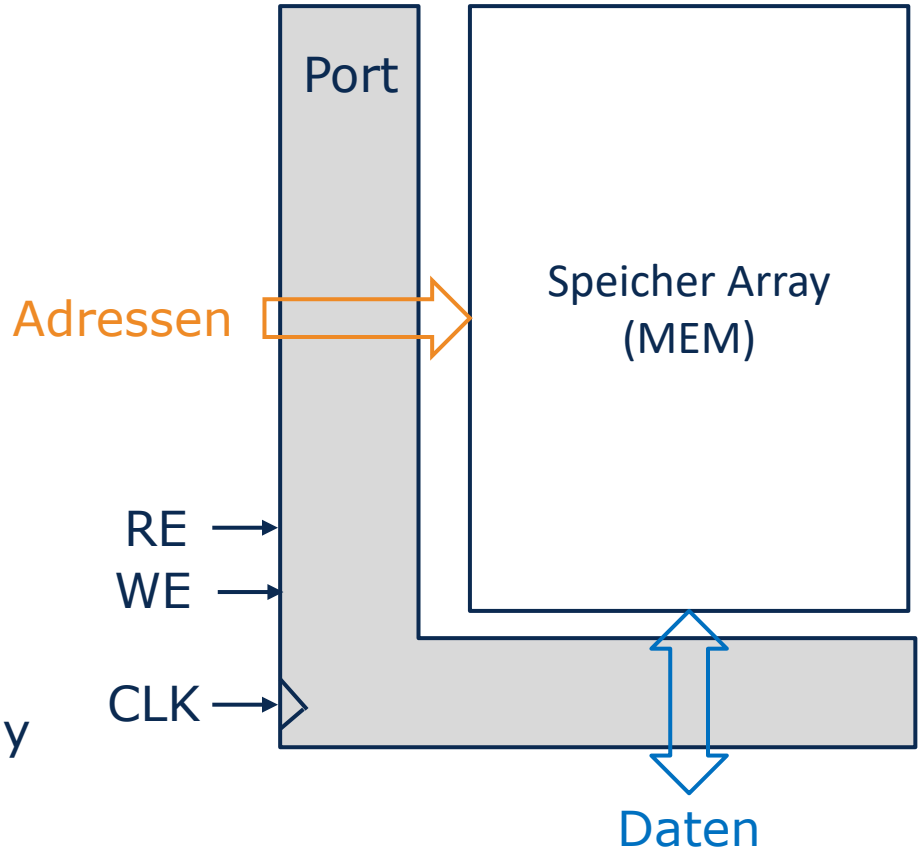
- Höherer Schaltungsaufwand (Fläche)
- Höhere Verlustleistung bei hoher Schreibaktivität
- Geringere Verlustleistung bei nur niedriger Schreibaktivität

Verlustleistungsreduktion durch Clock-Gating



- → Anwendung von Clock Gating abhängig vom Betriebsszenario

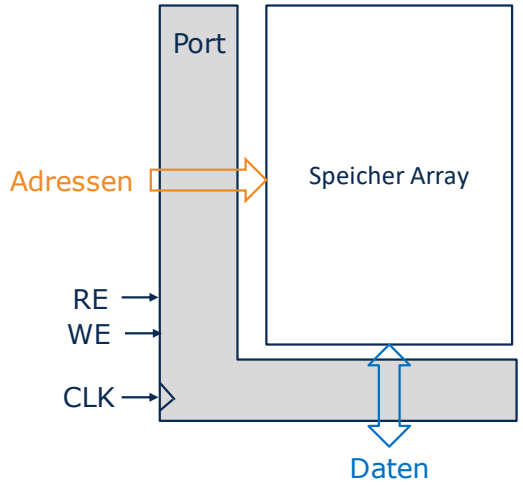
- Adressierbare Speicher
- Speicher Array
  - Datenwortbreite: n_D
  - Adressen: n_A
  - Kapazität $n_A \cdot n_D$
- Zugriff über Ports
  - Write: MEM(addr)=D_{IN}
  - Read: Q=MEM(addr)
- Kriterien:
  - Speicherdichte [Bit/µm²]
  - Zugriffzeit (Anzahl Takte, Delay CLK→Q)
  - Verlustleistung
  - Nicht-flüchtiger Speicher (ja/nein)

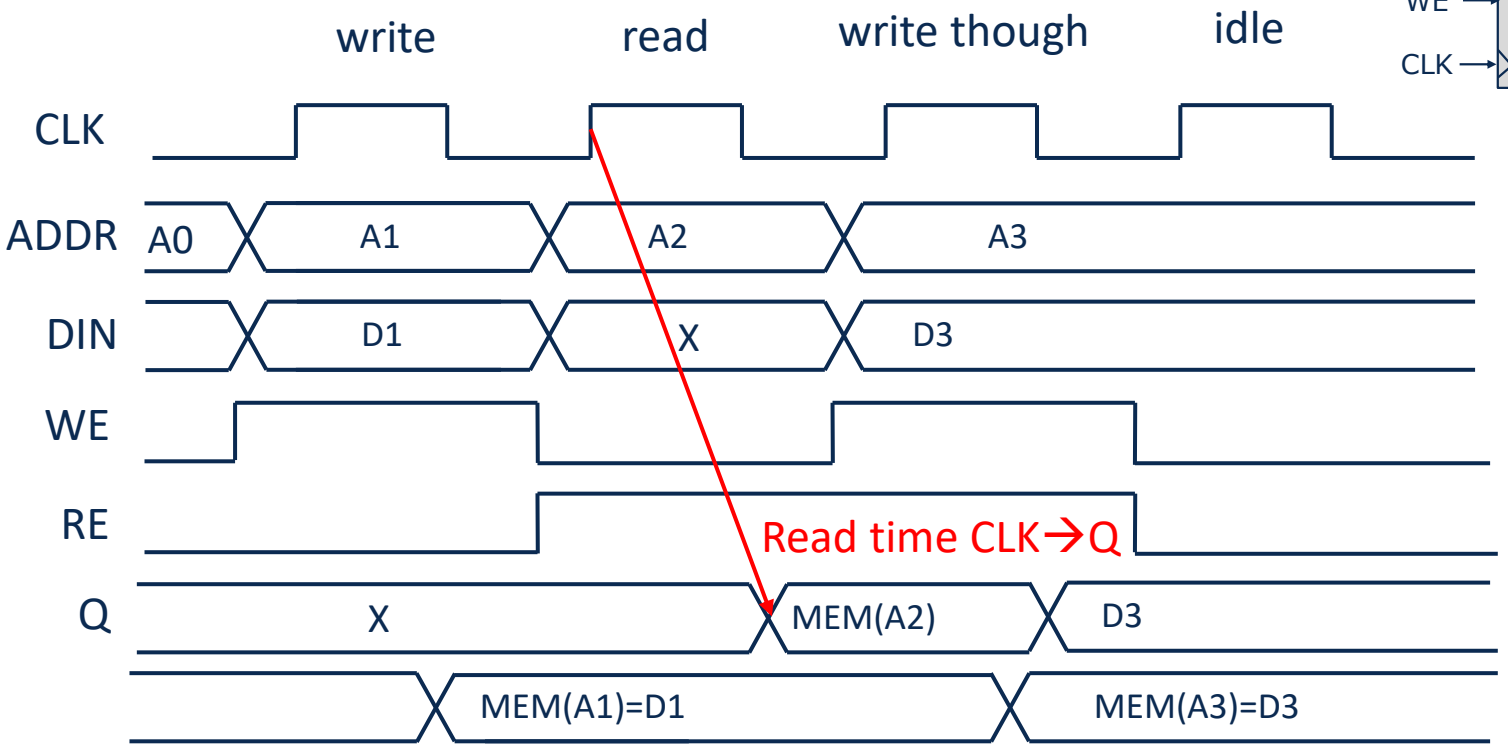


Typen von on-Chip Speicher

Typ	NVM	Bits/µm²	Zugriffzeit	Typische Speichergrößen	Kommentar
Latch	Nein	<4	einige 10ps	einige Bit	Ansteuerschaltung nötig da Level sensitiv!
FlipFlop	Nein	<1.6	einige 10ps	einige Bit	
SRAM	Nein	<35 (Zelle) <25(Makro)	einige 100ps	einige kBit bis MBit	Komplexe Ansteuerschaltung (SRAM Makro)
(embedded) DRAM	Nein	<200	einige 100ps	einige kByte bis Mbyte	Dynamischer Speicher, refresh nötig. Ggf. extra Prozessoptionen (Kosten!)
(embedded) Flash	Ja	<30	einige ns	einige kByte bis Mbyte	Extra Prozessoptionen (Kosten!)
ROM	Ja	<50	einige 100ps	einige kBit bis MBit	„Maskenprogrammierung“ bei der Implementierung
OTP	Ja	<10	einige 100ps	einige kBit bis MBit	„Einmalprogrammierung“ beim Test der Chips

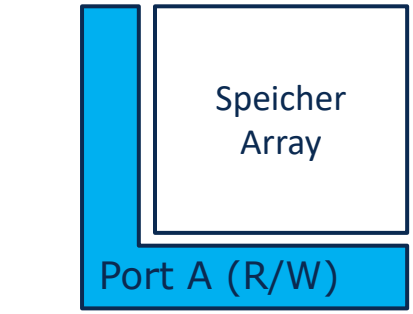
- Beispiel:
  - Synchroner SRAM Port
  - 1 Takt Zugriffszeit
  - Write-Through Funktion

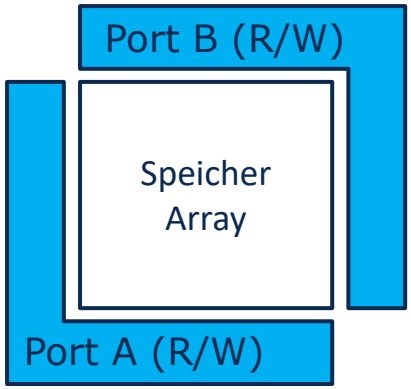


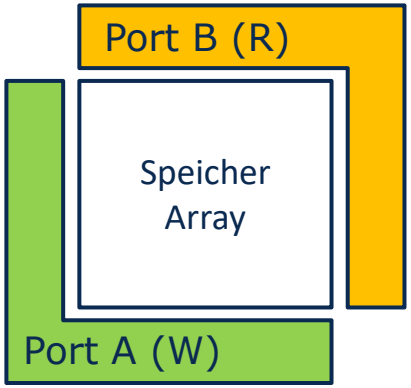


Klassifikation von SRAM Makros

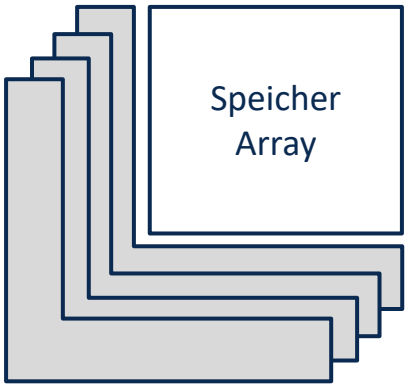
- Single Port (SP)
  - Zugriff von **einem Write/Read** Port
- Dual Port (DP)
  - Zugriff von **2 unabhängigen Write/Read** Ports
  - Synchrone und asynchrone Variante möglich
  - Arbitrierung/Priorisierung von Write Zugriffen auf gleiche Adressen nötig
- Two-Port (TP)
  - Zugriff von **einem Write Port** und **einem separaten Read Port**

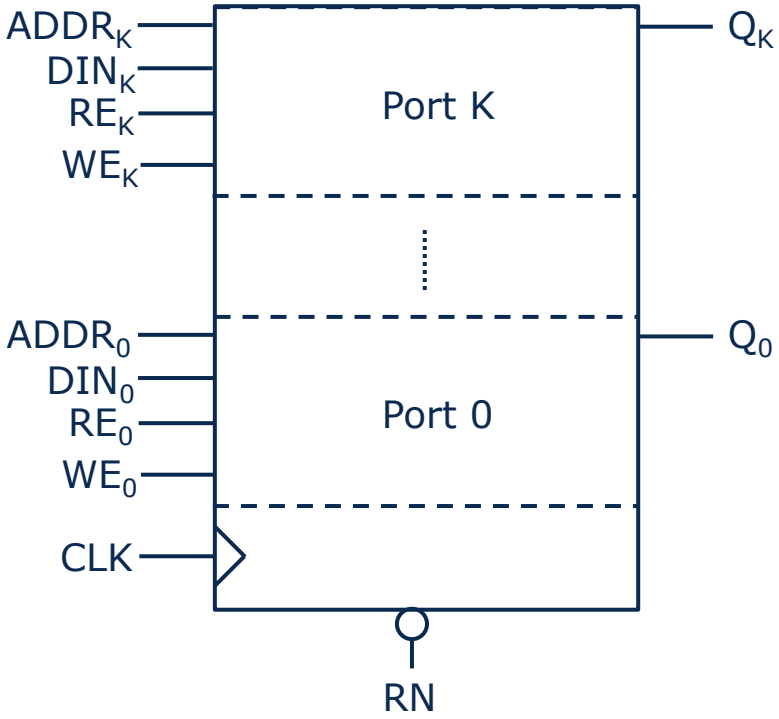






- Arrays aus Registern mit mehren Write- und Read-Ports
- Adressierbarer Zugriff
- Typischer Weise synchrone Realisierung (gleicher Takt für alle Ports)
- Schneller Zugriff (1 Takt), kurzes Delay CLK ↑ →Q



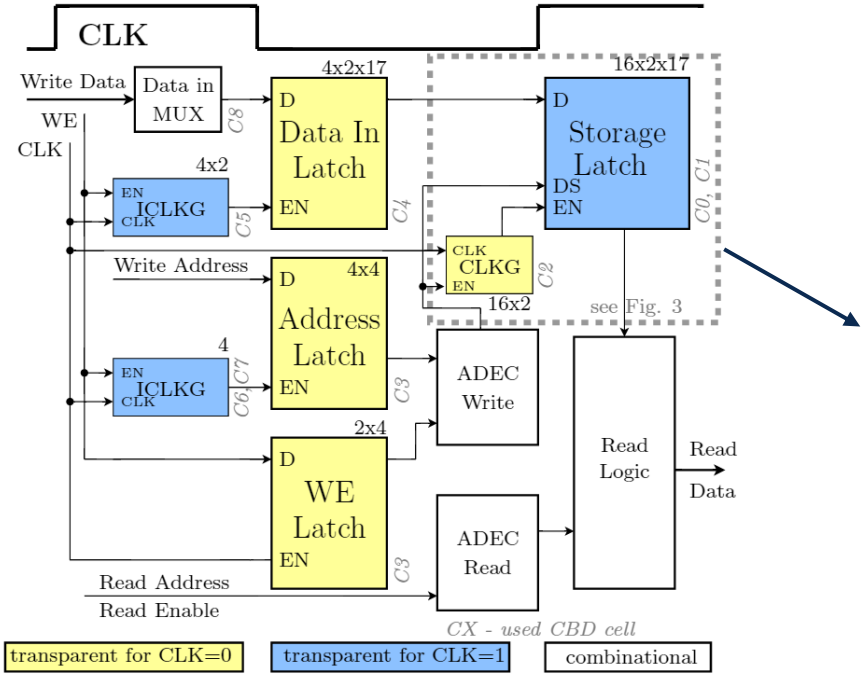


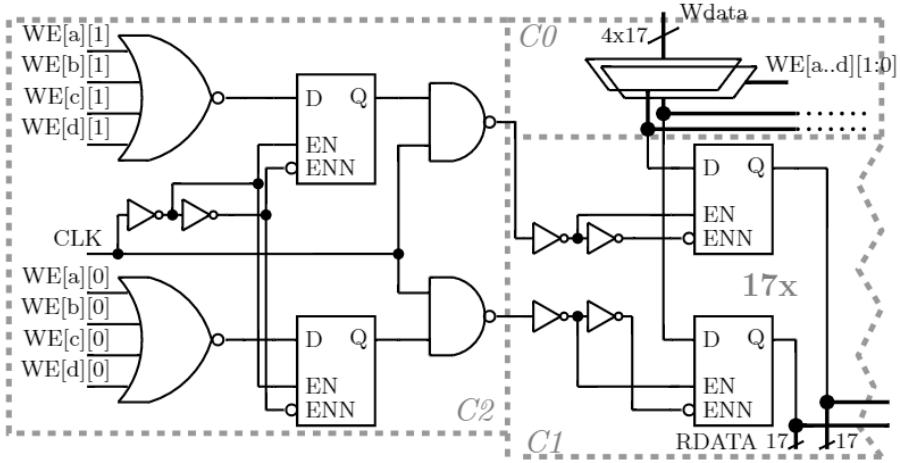
- Anwendung in Systemen mit mehreren Rechenkernen

Beispiel: Register File für SDR Prozessor

- Register File für Infineon X-GOLD SDR™ 20 Prozessor
- Entwickelt an der Stiftungsprofessur HPSN (2008/2009)
- 16-Worte zu je 34 Bit, 4-Write Ports, 6 Read Ports
- Cell-Based Design → Optimierung auf Transistorlevel

Quelle: infineon.com

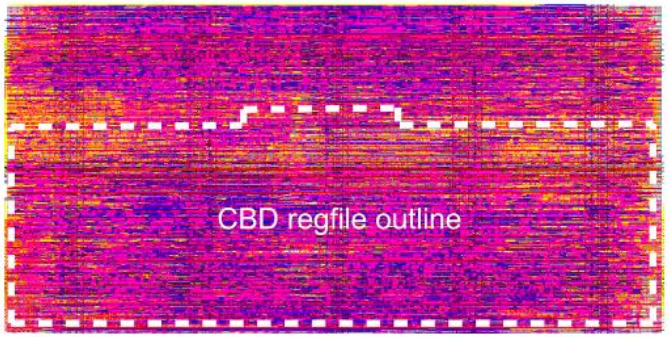


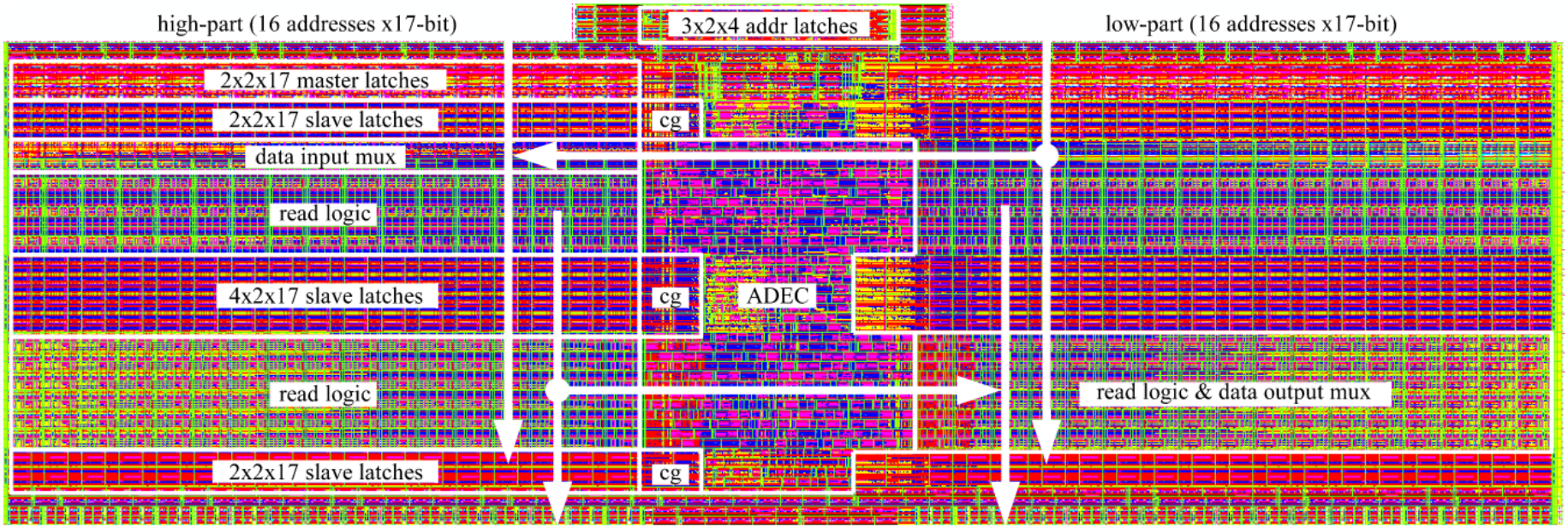


Johannes Uhlig, Sebastian Höppner, Georg Ellguth, and René Schüffny, A Low-Power Cell-Based-Design Multi-Port Register File in 65nm CMOS Technology, IEEE International Symposium on Circuits and Systems ISCAS 2010, 2010, p. 313-316,

Beispiel: Register File für SDR Prozessor

- Implementierung in 65nm CMOS Technologie
- ≈0,026mm² Chipfläche (0,02Bit/µm²)
- 300MHz Taktfrequenz
- Effizienzsteigerung verglichen mit einer Synthese und Place&Route Implementierung:
  - 44% Flächenersparnis und
  - 30% Verlustleistungsreduktion



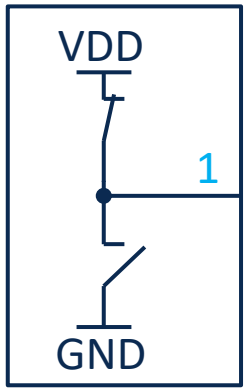


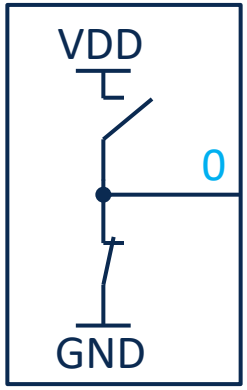
Johannes Uhlig, Sebastian Höppner, Georg Ellguth, and René Schüffny, A Low-Power Cell-Based-Design Multi-Port Register File in 65nm CMOS Technology, IEEE International Symposium on Circuits and Systems ISCAS 2010, 2010, p. 313-316,

Bussysteme

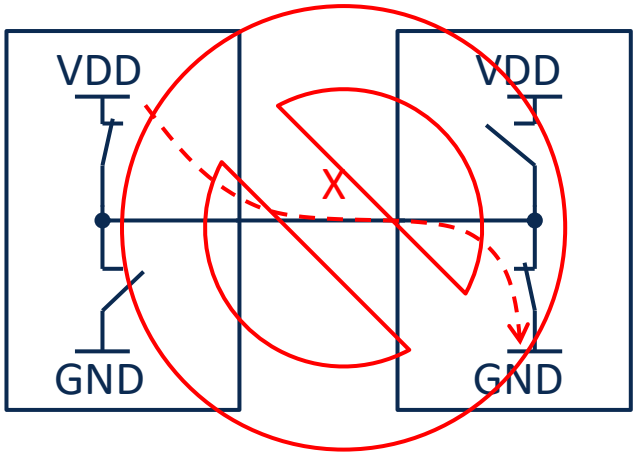
- Die Verschaltung von Datenpfadbaublöcken soll flexibel und re-konfigurierbar erfolgen
- Bussysteme dienen der Vernetzung von Datenpfadelementen
- Vorstellung von 3 Ansätzen:
  - Tri-state Bus
  - Multiplexer Bus
  - Komplexe Bussysteme und Network-on-Chip

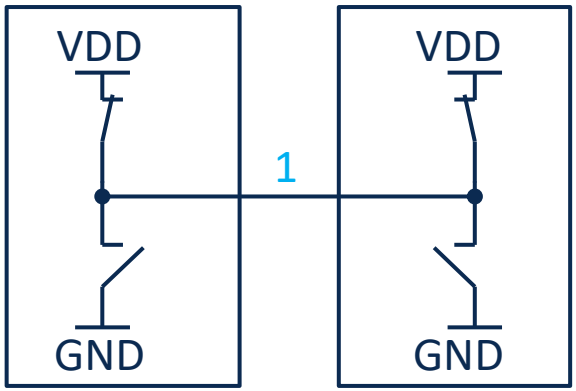
- Gatter**ausgänge** erzeugen 2 Logikpegel (0,1)

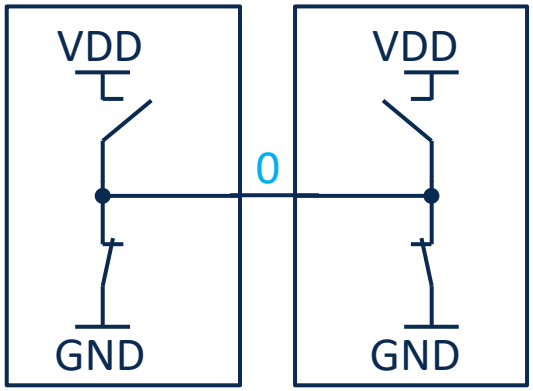




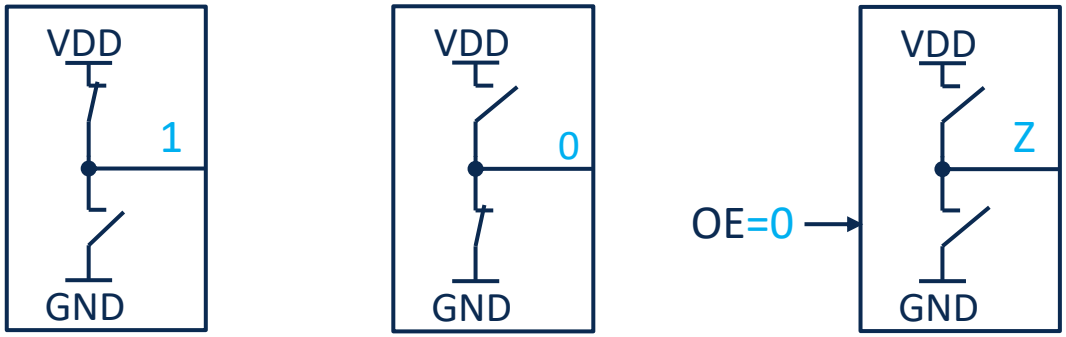
- Das Zusammenschalten mehrerer Gatter**ausgänge** ist nur möglich wenn sie die **gleichen** Pegel treiben



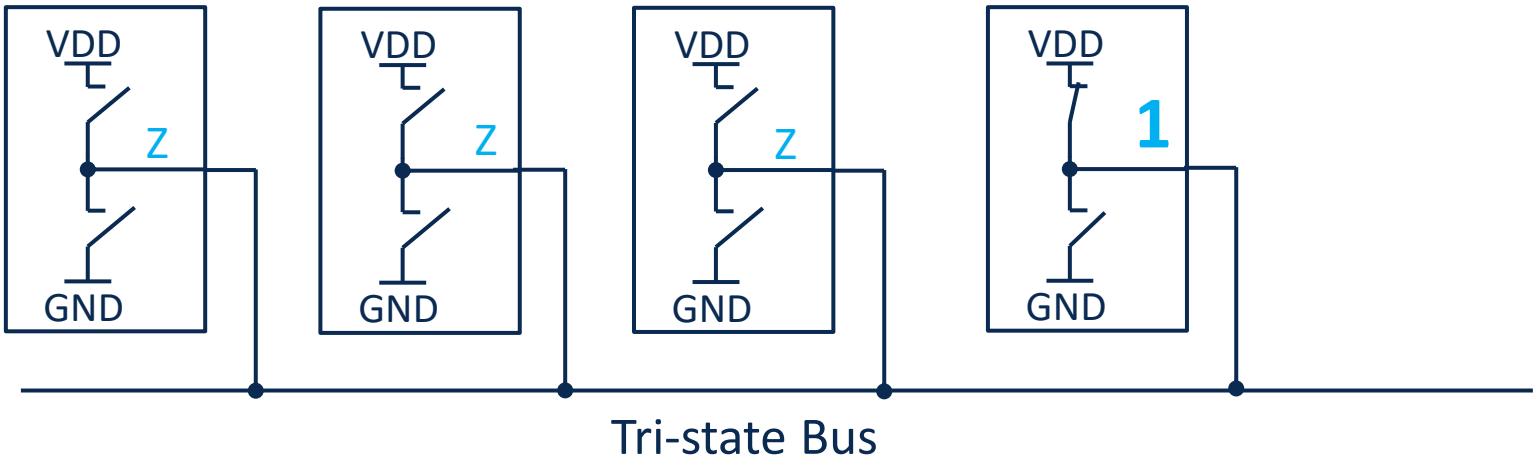




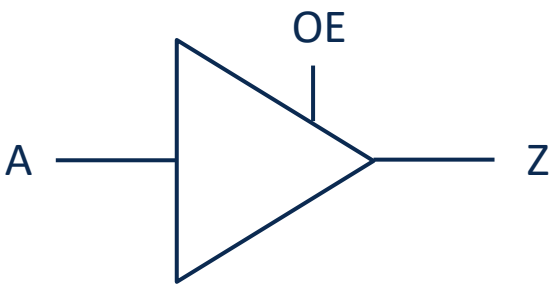
- Einführung eines **Dritten** Ausgangszustandes (Tri-State) **Z**
- Ausgangstreiber wird hochohmig geschalten, durch separates Steuersignal (OE)



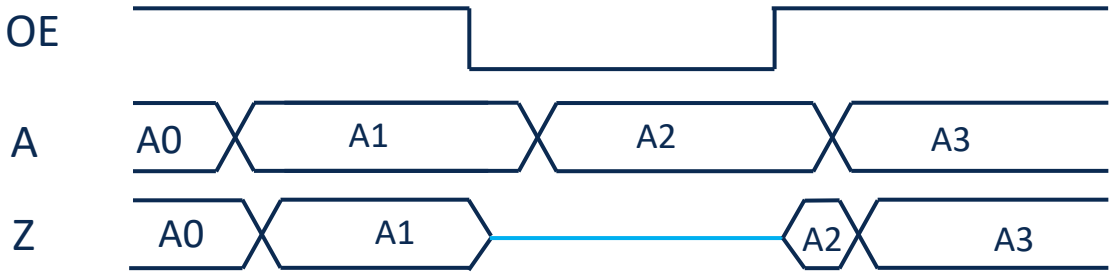
- Das Zusammenschalten mehrerer Gatter**ausgänge** im Tri-state ist möglich.
- Es darf nur ein Treiber aktiv sein!



- Buffer mit Tri-state Ausgangstreiber

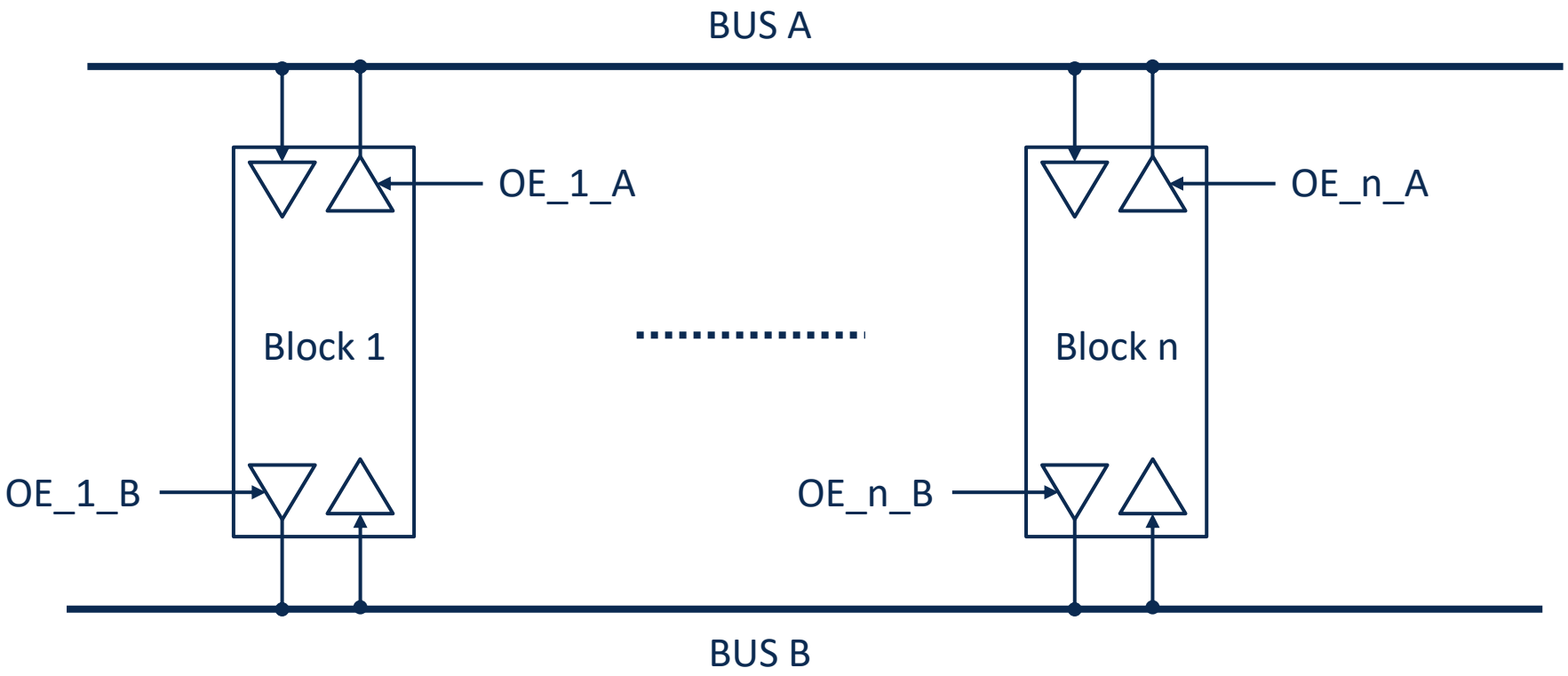


OE	A	Z
0	X	Z
1	0	0
1	1	1



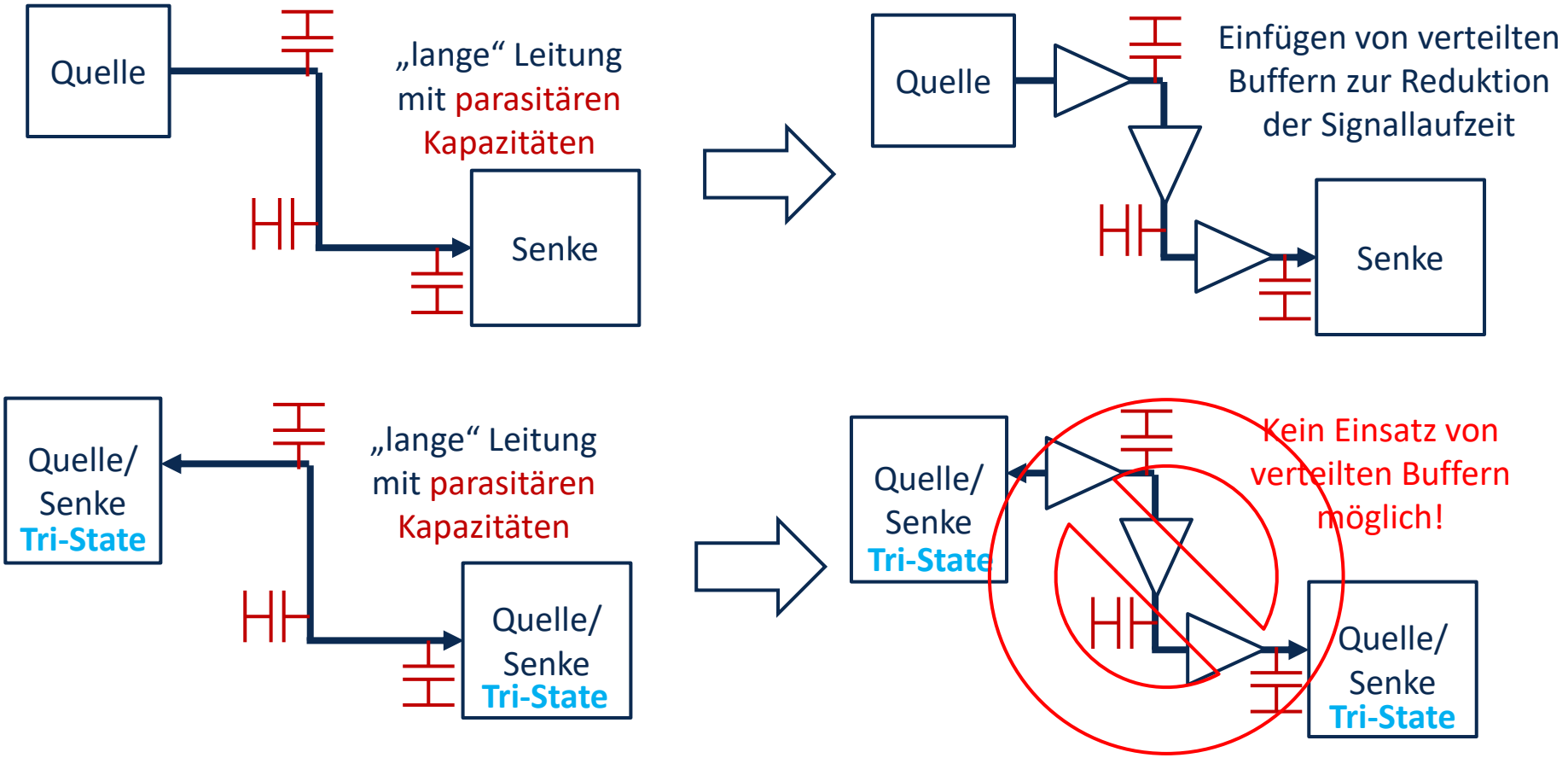
- Kombinatorische Gatter und sequentielle Baublöcke (z.B. Register) können mit Tri-state Ausgängen versehen werden.

Datenpfade mit Tri-State-Bussen



- Mehrere Baublöcke können an **einen** Tri-State Bus angeschlossen werden
- Pro Tri-State Bus darf nur **ein** Treiber aktiv sein → Steuerwerk!
- Mehrere Busse möglich für **parallelen** Datentransfer zwischen Baublöcken

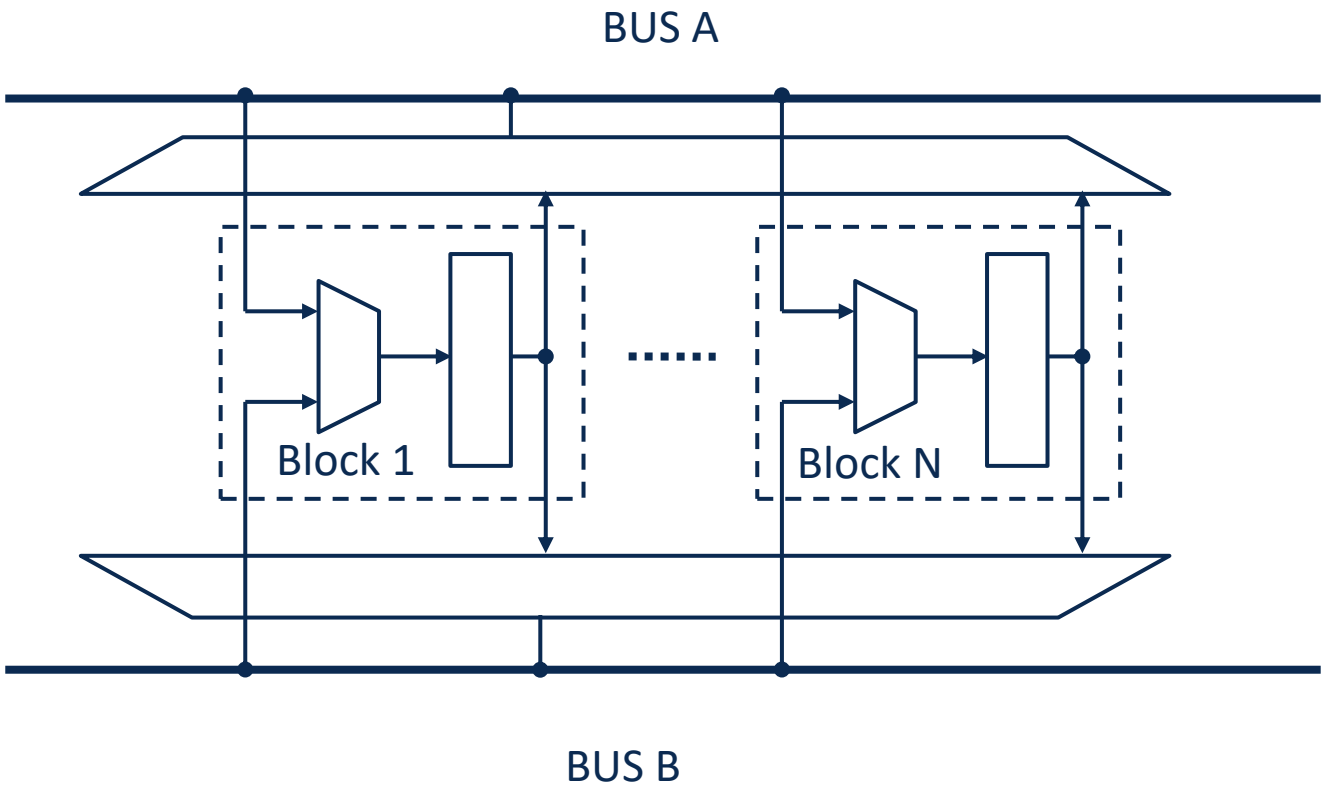
Nachteil von Tri-State-Bussen



- In aktuellen Entwürfen mit automatischem Platzieren und Verdrahten (P&R), werden Tri-State Signale vermieden!

Multiplexer Busse

- Auswahl der Datenquellen für den Bus durch Multiplexer
- Auswahl des Busses für den Eingang des Datenpfad Elements durch Multiplexer
- Setzen der Multiplexer Select Signale durch das Steuerwerk



Multiplexer Bus

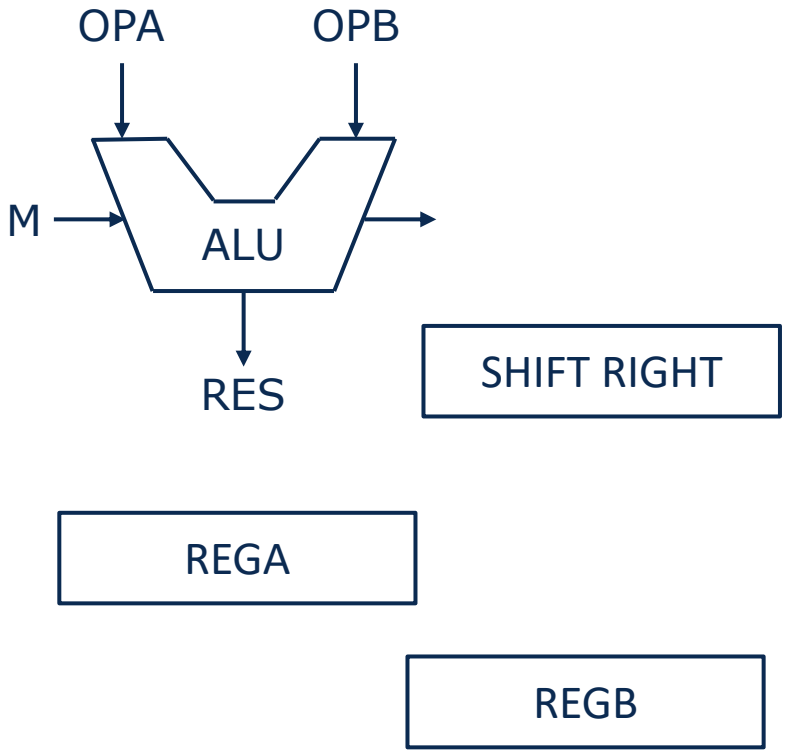
- Vorteil: Keine Tri-State Treiber

- Anzahl der Multiplexer Eingänge abhängig von notwendigen Datenverbindungen
- Ziel: Minimierung der notwendigen Multiplexer Eingänge
- Möglichkeit der hierarchischen Realisierung von Multiplexern
  - Bus Multiplexer: Auswahl der Datenquelle für den Bus
  - Eingangs Multiplexer: Auswahl der Datenquelle für den Eingang des Datenpfadelementes

Beispiel: Bus-Implementierung

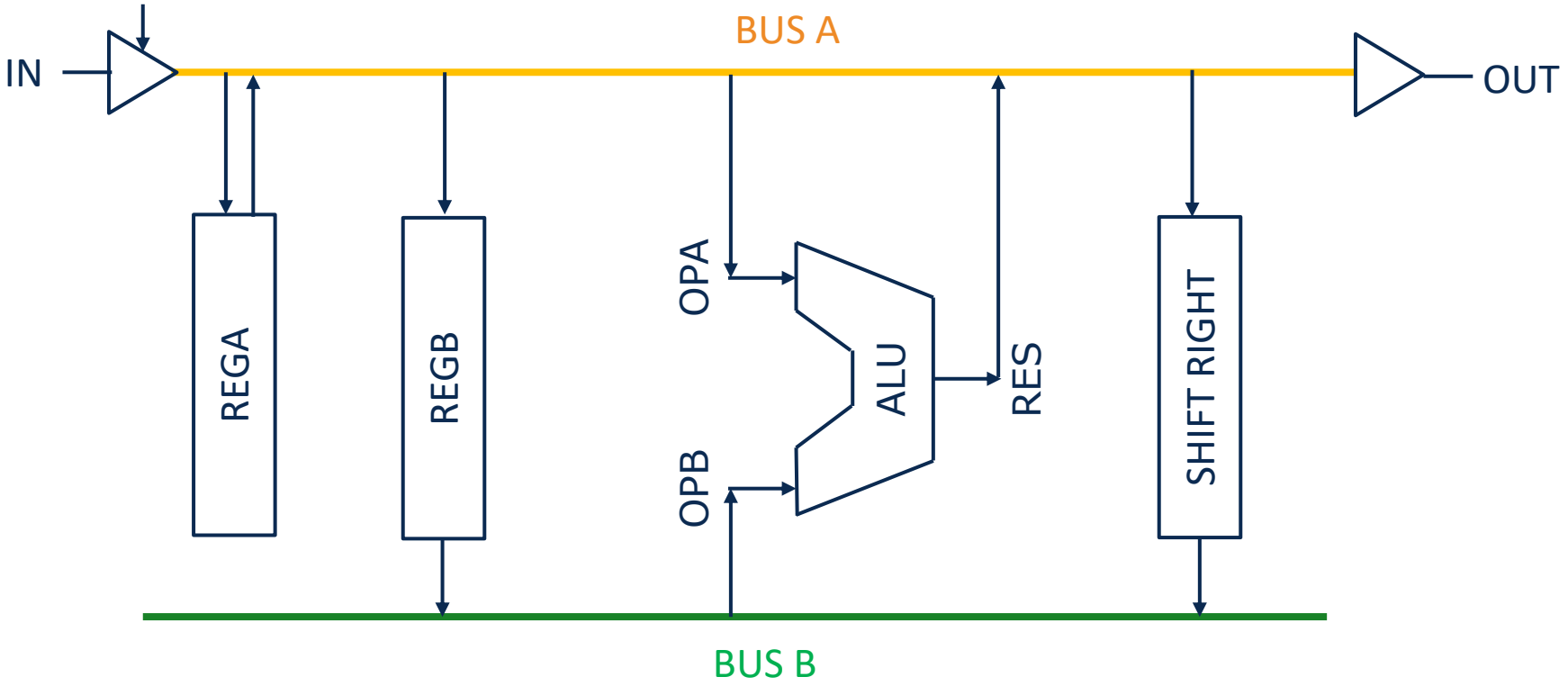
- Beispiel: C=(A+B)+A/2

Zustand	Datenquelle	Anzahl Busse
LOAD A	IN→ REGA	1
LOAD B	IN→ REGB	1
COMP1	REGA → ALU(OPA) REGB → ALU(OPB) REGA → SHIFT	2
COMP2	ALU(RES)→ ALU(OPA) SHIFT→ ALU(OPB)	2
STORE	ALU(RES) → OUT	1

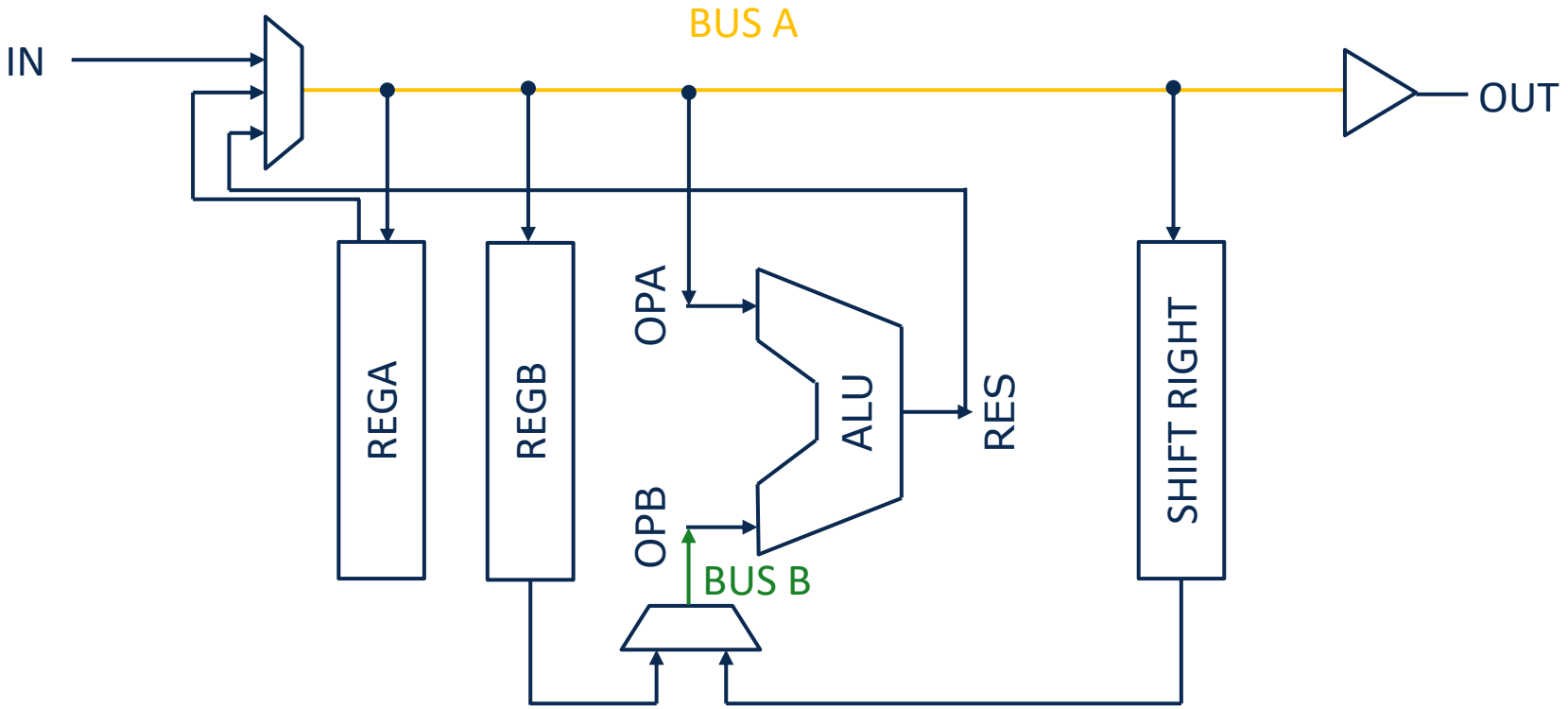


- → 2 Datenbusse nötig (BUS A, BUS B)

Beispiel: Bus-Implementierung Tri-State Bus

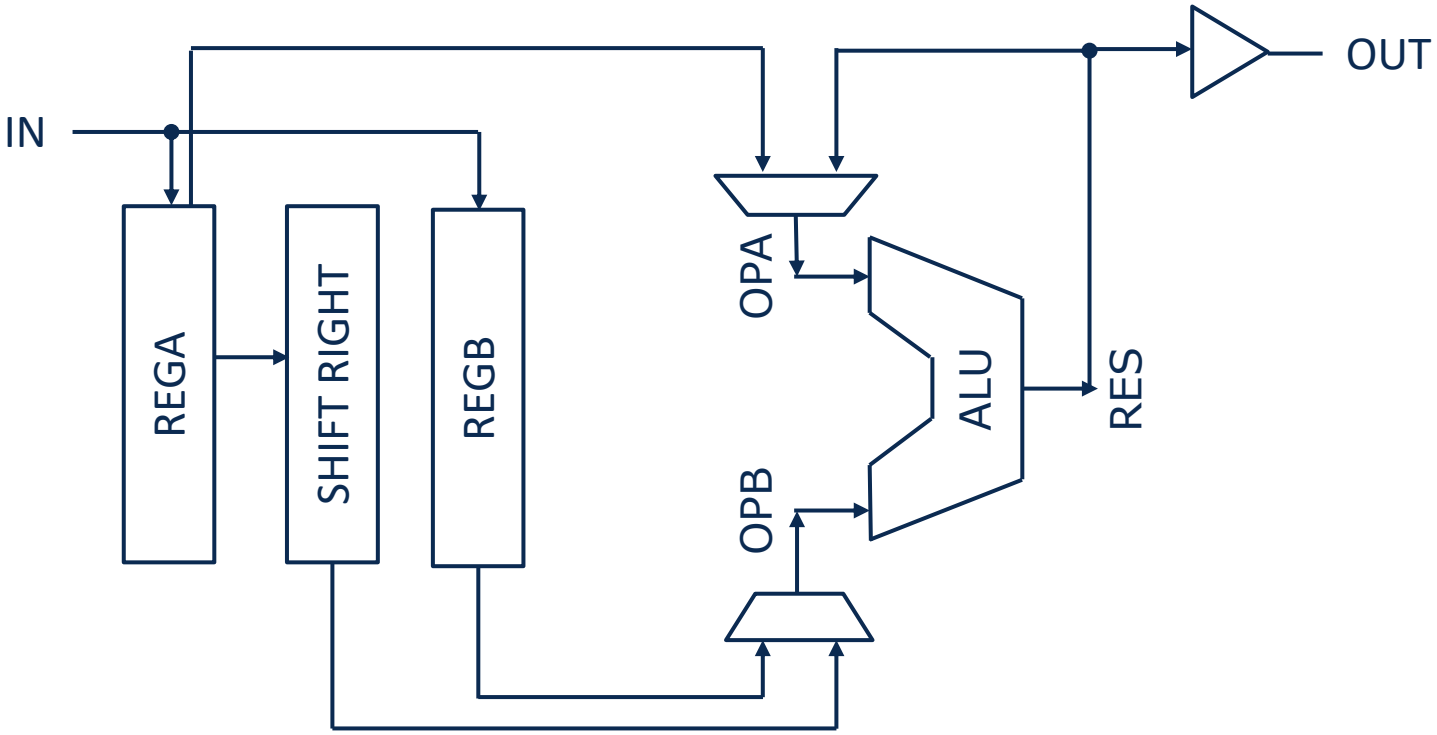


Beispiel: Bus-Implementierung Multiplexer Bus Variante 1

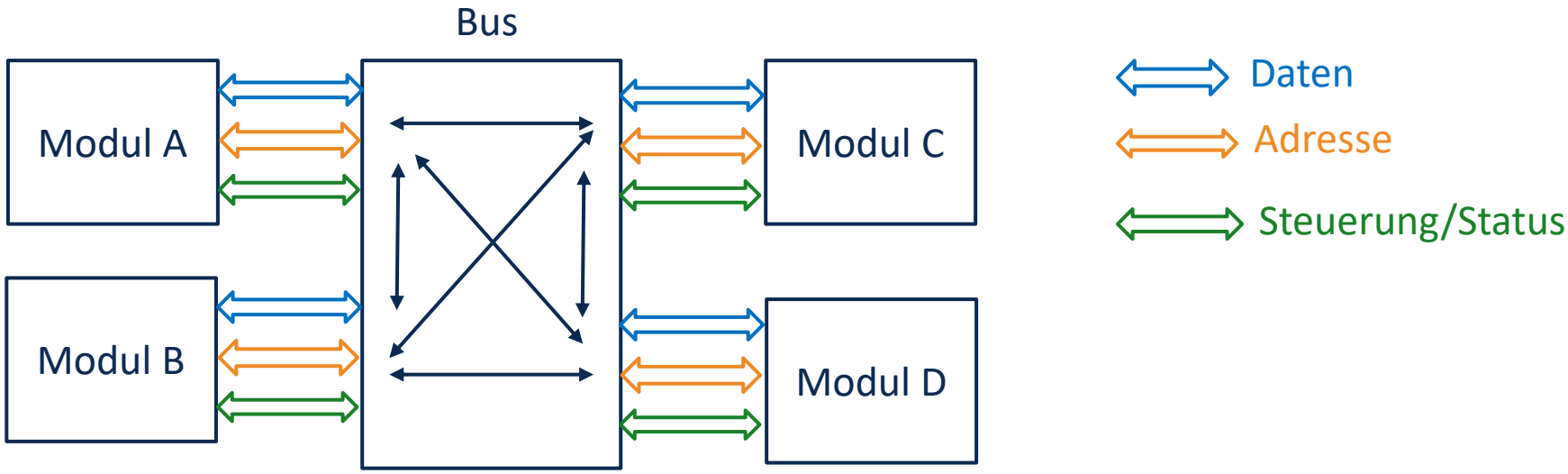


Beispiel: Bus-Implementierung Multiplexer Bus Variante 2

- Reduktion der Multiplexer

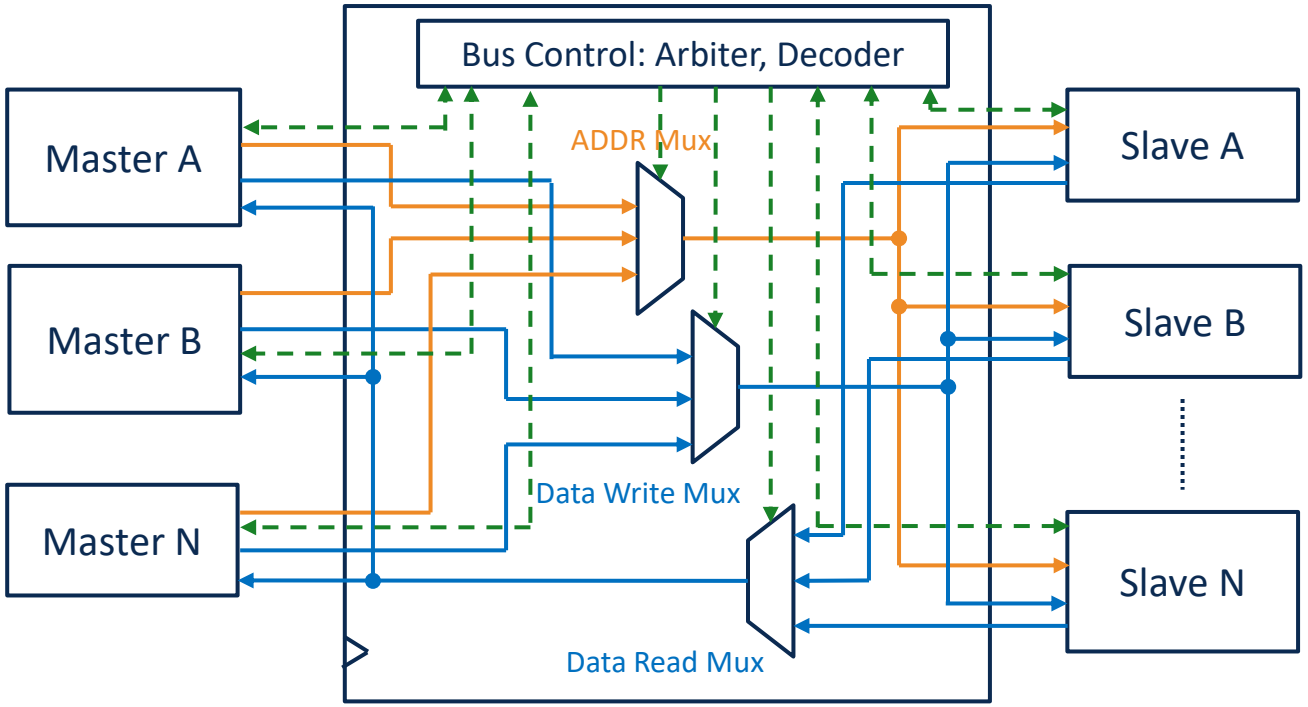


- Die bisher vorgestellten Bus-Systeme erfordern **Anwendungsspezifische** Kontrolle durch ein Steuerwerk
- Sie sind nicht standardisiert → Einbinden vorgefertigter Baublöcke (IP) schwierig.
- Komplexe digitale Systeme erfordern flexible, standardisierte Bussysteme
- Grundkonzept:
  - Standardisierte Busschnittstelle (Daten, Adresse, Steuer- und Statussignale)
  - Adressierung der Busteilnehmer (ID)
  - Zugriff über Ports (ähnlich Memory)
  - Businterne Steuerlogik realisiert den Datentransfer und Überwacht den Zugriff



Komplexe Bussysteme – Crossbar Bus

- „Circuit Switched Network" → Multiplexer Bus
- Master und Slave Komponenten mit standardisiertem Bus-Interface
- Arbitrierung und Priorisierung des Zugriffs durch Bus Controller
- Synchrone Realisierung (CLK)
- Sequentieller Zugriff (Adress-Cycle, Data Cycle, Wartezyklen wenn Bus „busy", Burst)
- Beispiele: APB, AHB, AXI, WishBone, …

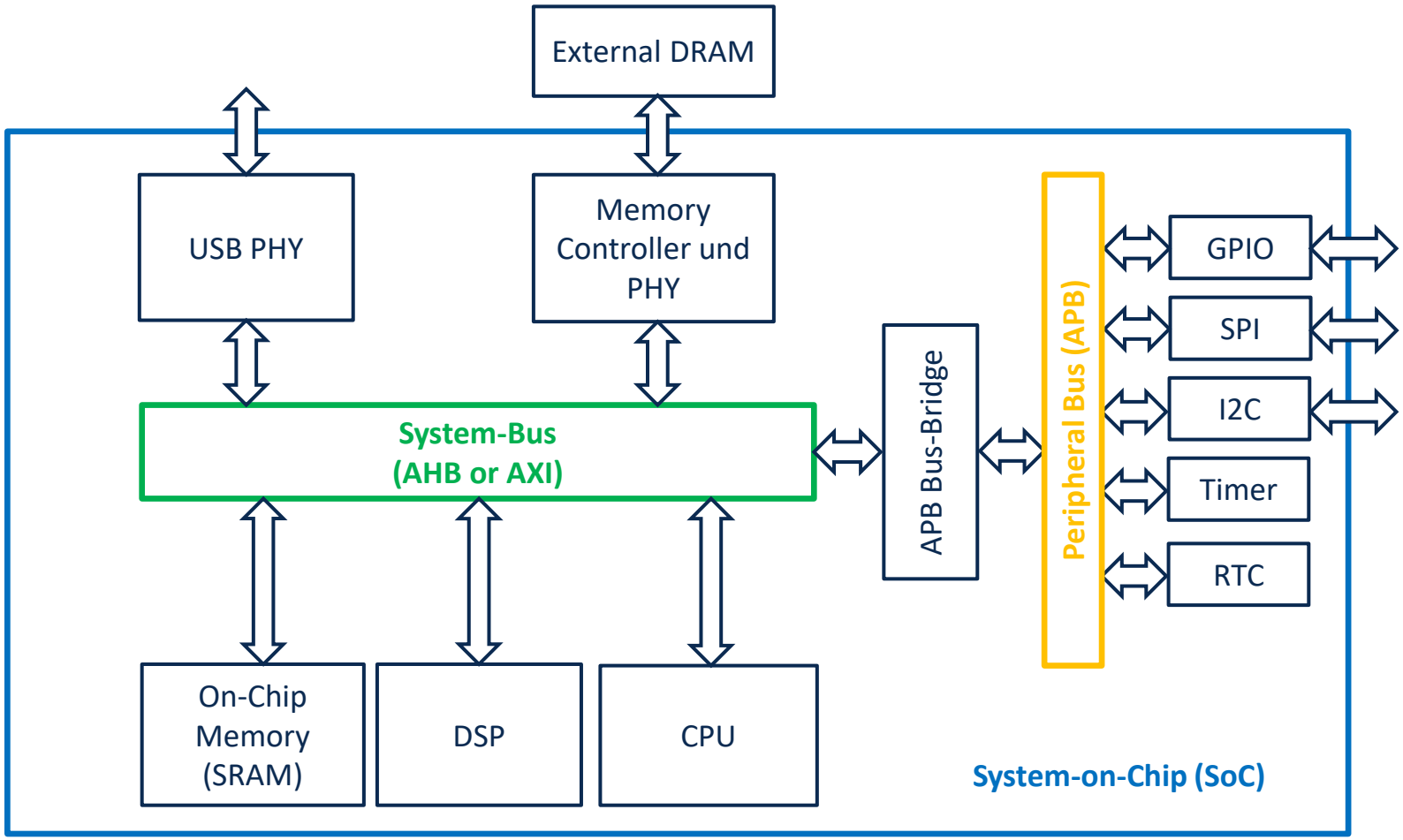


- Advanced Microcontroller Bus Architecture (AMBA)
- Entwickelt von von Advanced RISC Machines Ltd. (Arm)
- AMBA ist ein freier, offener Standadard zur Verbindung von System-on-Chip (SoC) Baublöcken.
- Die AMBA Spezifikationen sind weit verbreitet als
  - Standard für on-Chip Kommunikation
  - Standard Interface für IP re-use.
- Vereinfacht die Implementierung komplexer SoCs mit vielen Baublöcken

- AMBA definierte Bus-Systeme:
  - **APB** (Advanced Peripheral Bus)
  - **AHB** (Advanced High-performance Bus)
  - **AXI** (Advanced eXtensible Interface Bus)

- Für Trace und Debug-Anwendungen
  - **ATB** (Advanced Trace Bus)

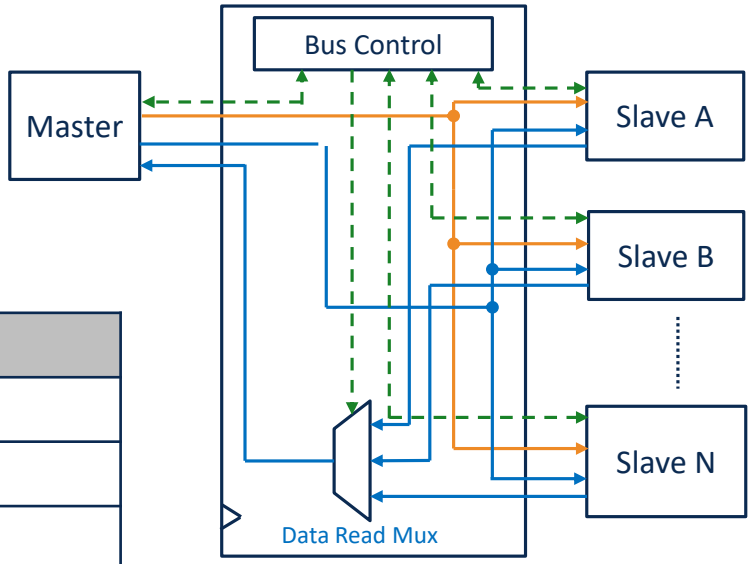
Beispiel SoC mit AMBA Bussen



APB (Advanced Peripheral Bus)

- Einfaches Bus-System mit geringer Komplexität
- Geringe Bandbreite, kein Pipelining
- Single Master, multiple Slaves
- Spezifikation siehe: https://developer.arm.com/documentation/ihi0024/c/

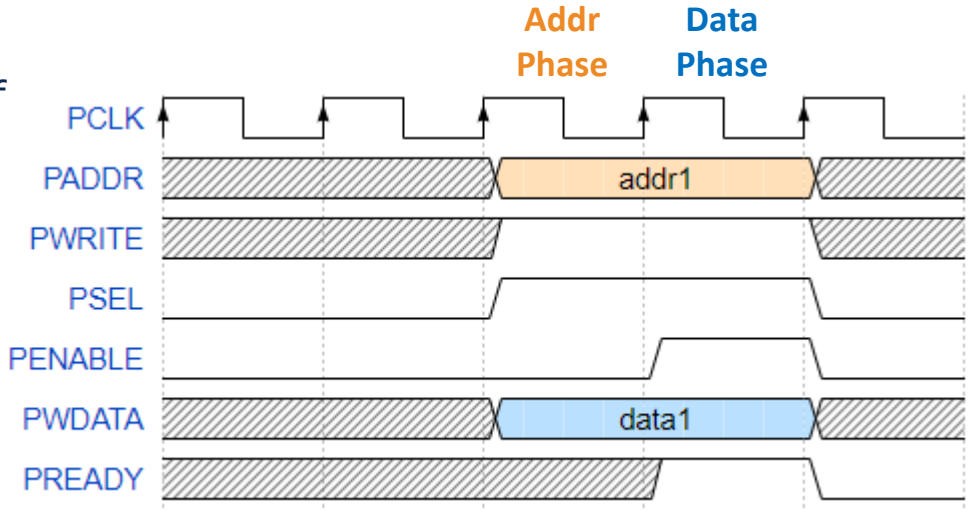
- APB Signal-Definition:



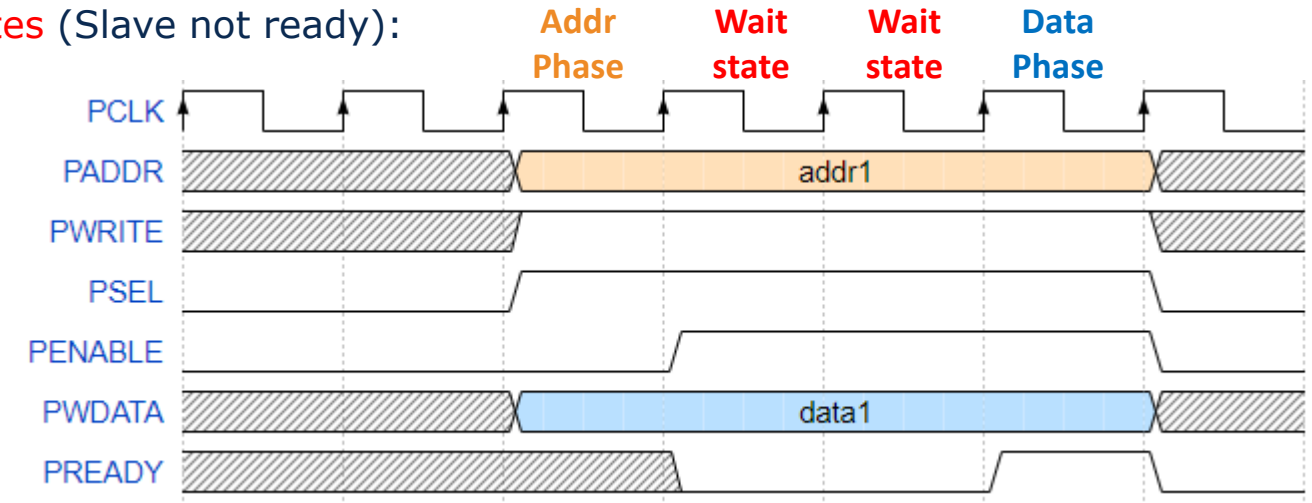
Signal (Master)	Signalrichtung	Beschreibung
PCLK	→ Master, Slaves	Taktsignal
PRESET	→ Master Slaves	Reset Signal
PADDR[a-1:0]	Master → Slaves	Adresse, bis 32-Bit Breite
PSEL<n-1:0>	Master → Slaves	Select Signal für Device (Slave)
PENABLE	Master → Slaves	Zeigt Daten-Transfer-Zyklus an
PWRITE	Master → Slaves	1: Write, 0: Read
PWDATA[d-1:0]	Master → Slaves	Write-Data, bis 32-Bit Breite
PREADY	Slaves → Master	Getrieben vom Slave, Kann Transfer verlängern
PRDATA[d-1:0]	Slaves → Master	Read-Data, bis 32-Bit Breite
PSLVERR	Slaves → Master	Zeigt Transfer Error an

APB Waveforms: Schreiben auf den Bus

- APB Write:
- 2 Takte pro Bus-Zugriff

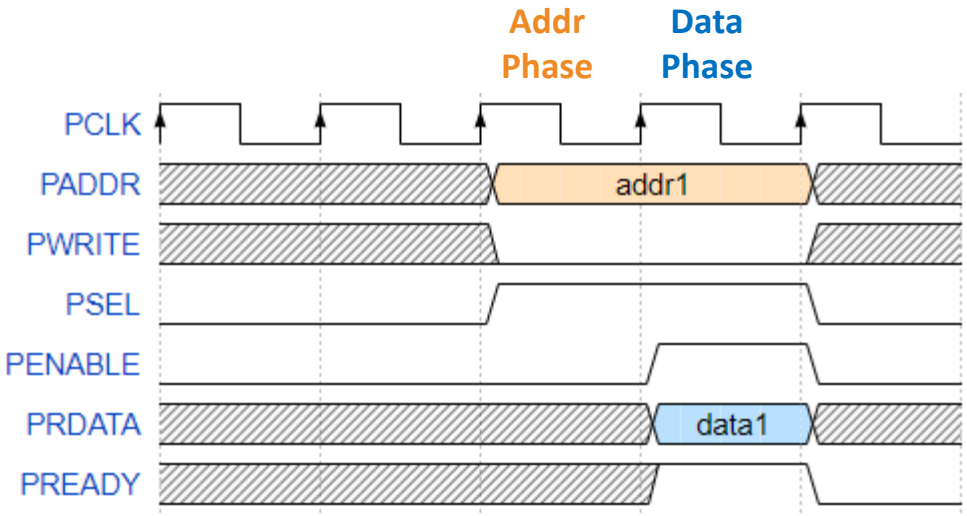


- APB Write mit Wait States (Slave not ready):

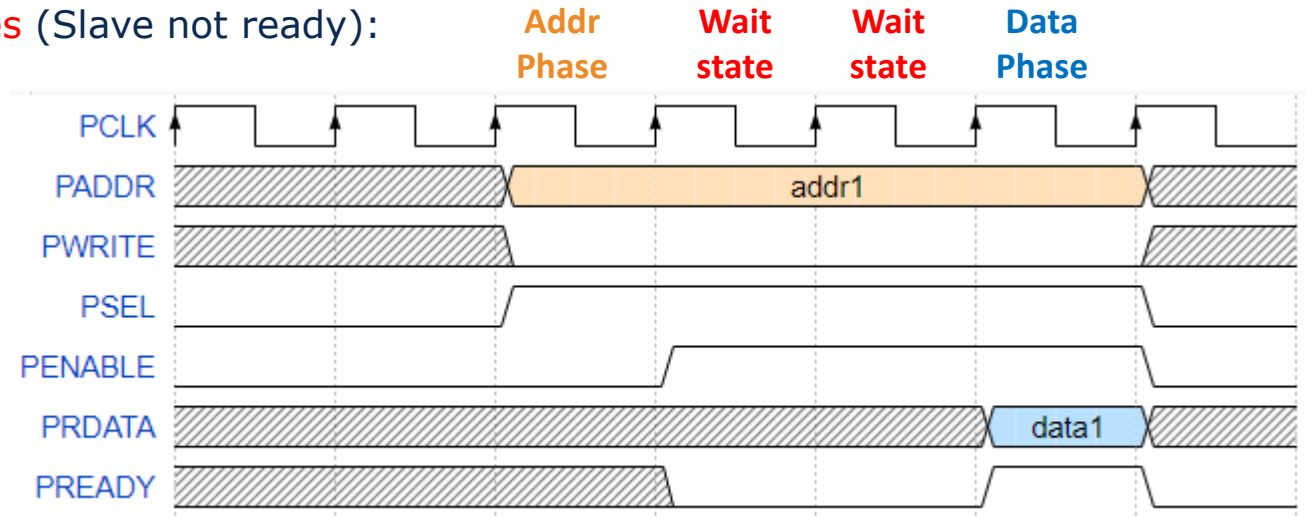


APB Waveforms: Lesen vom Bus

- APB Read
- 2 Takte pro Bus-Zugriff

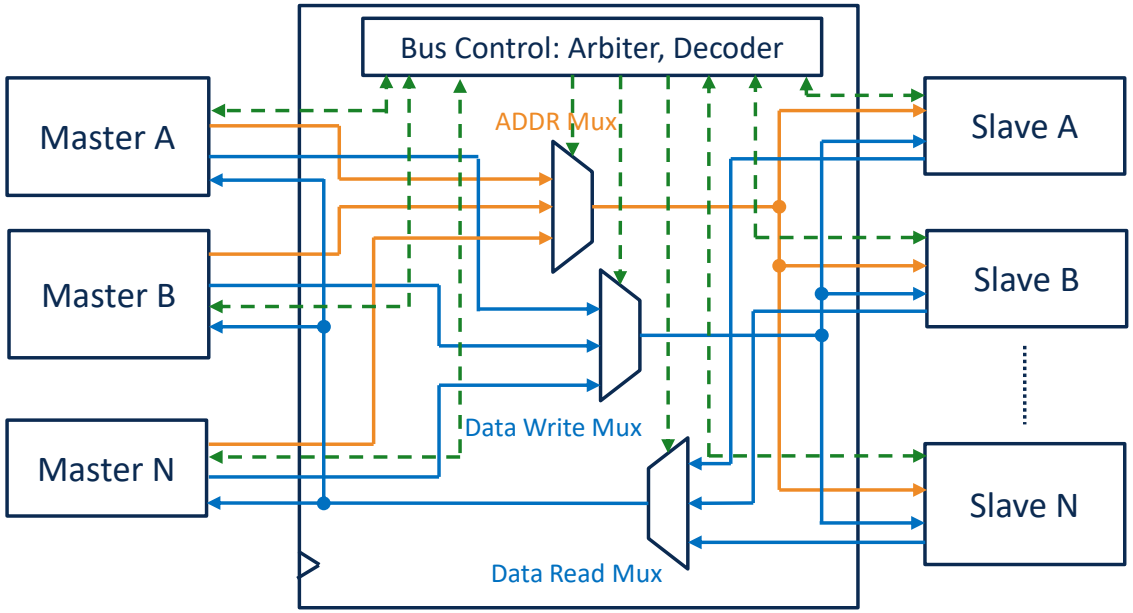


- APB Read mit Wait States (Slave not ready):



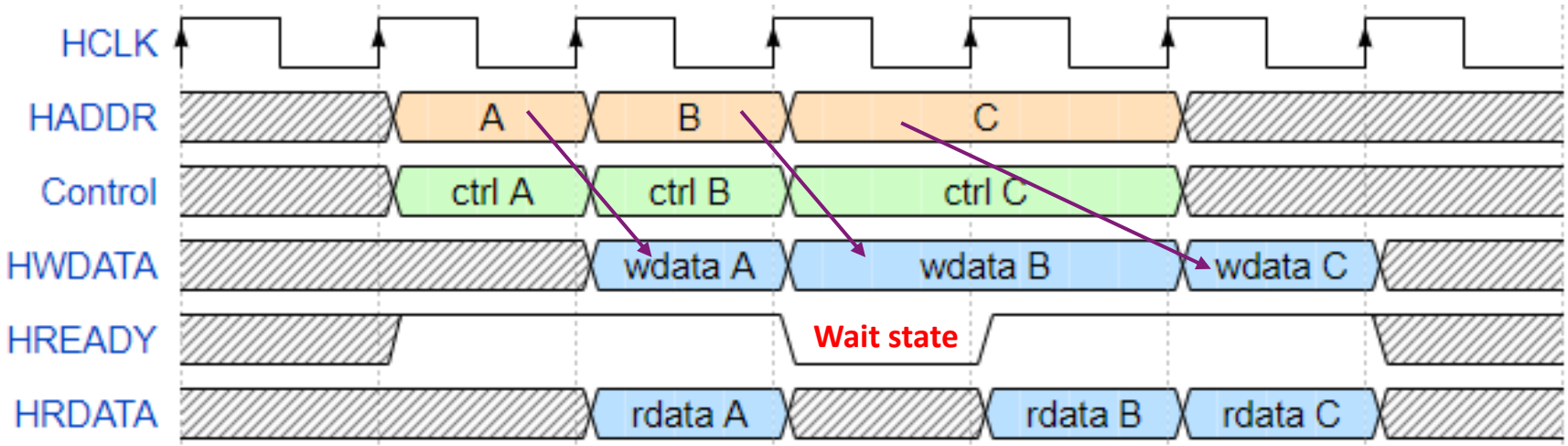
AHB (Advanced High-performance Bus)

- Ein uni-direktionaler Adress-Bus
- Zwei- uni-direktionale Datenbusse
- Hoher Datendurchsatz mit Pipelining
- Multi-Master Fähigkeit mit Arbitierung
- Burst-Transfers (4, 8, 16 nacheinanderfolgende Transfers **eines** Masters oder Slaves)
- Spezifikation siehe: https://developer.arm.com/documentation/ihi0033/a/



Beispiel: AHB Pipelining

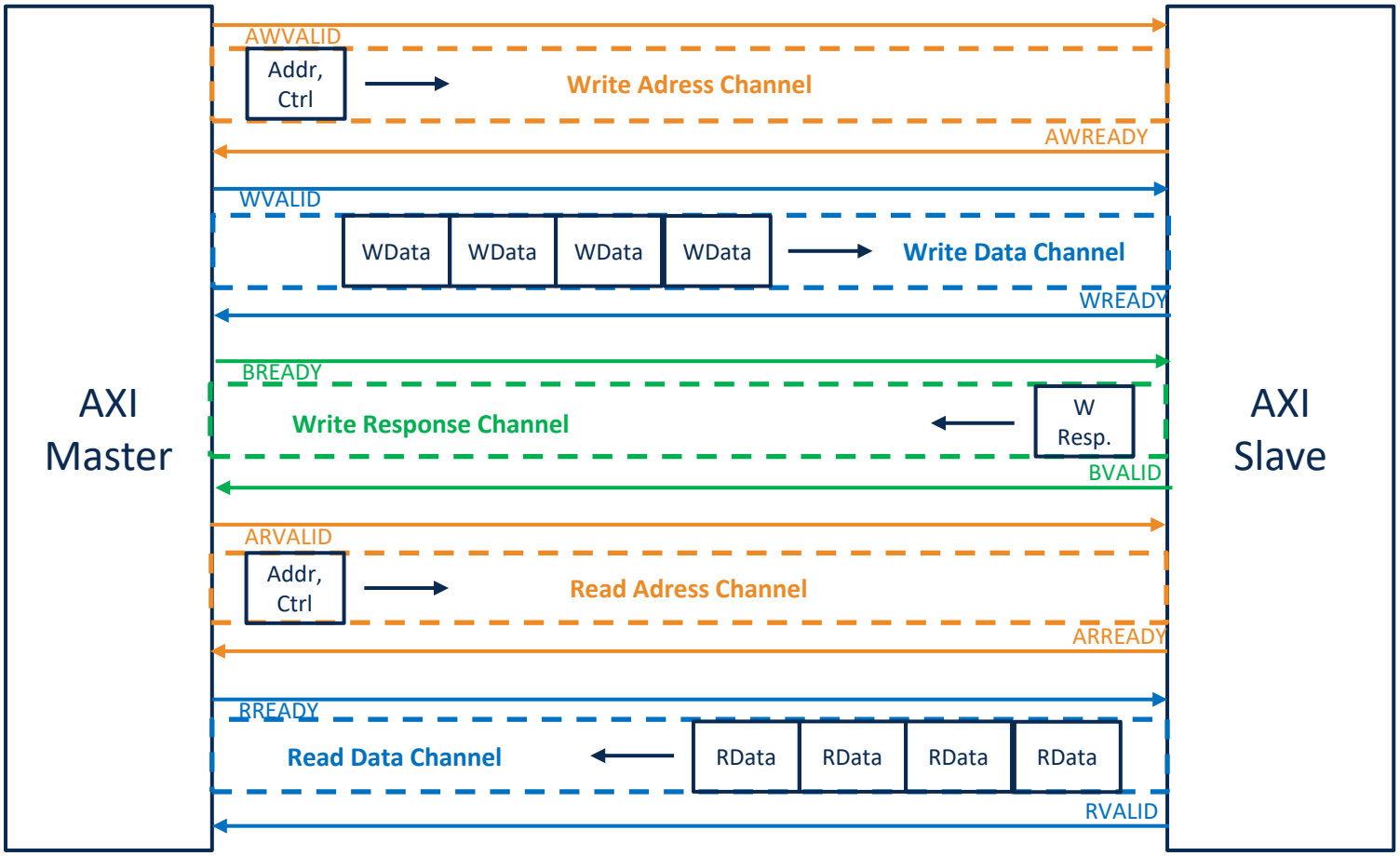
- Pipelined AHB Transaktionen
- Bis zu effektiv eine Bus-Transaktion pro Takt für hohen Datendurchsatz



AXI (Advanced eXtensible Interface Bus)

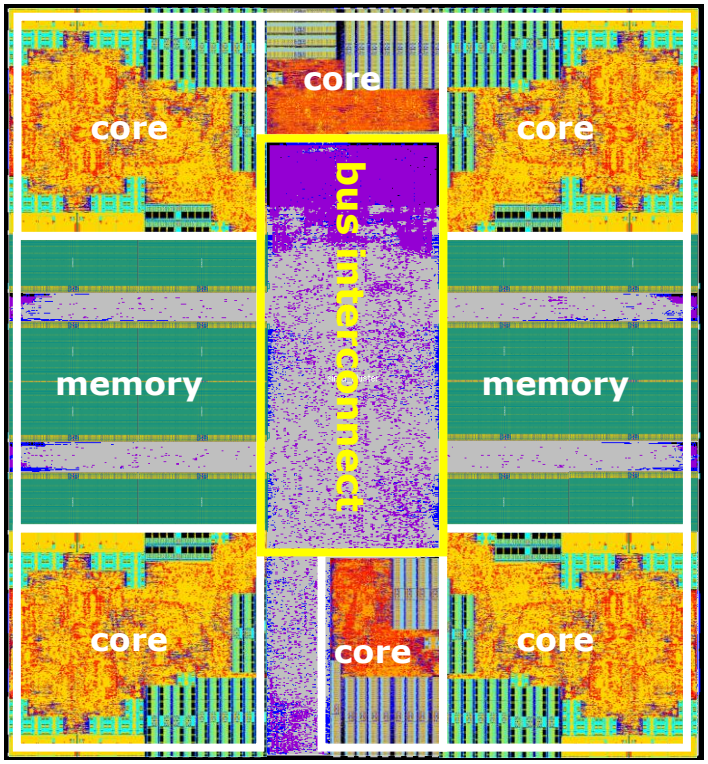
- Flexibles aber komplexes Bus-System für high-performance Interconnects
- Features (u.a.):
  - Separate Unabhängige Channels für Read Adresse, Write-Adresse, Read-Daten, Schreib-Daten, Write-Response
    - Information fließt nur, wenn **Quelle gültig** und **Ziel ready** ist
    - In jedem Channel kann Master oder Slave den Datendurchsatz limitieren
  - Bursts nur mit Übertragung der Startadresse
  - Out-of Order Completion von Transaktionen

- Unabhängige, Daten und Control (Ready, Valid) Signale pro Channel
- Synchronisation der Channels mit Transaction-IDs (Tags)



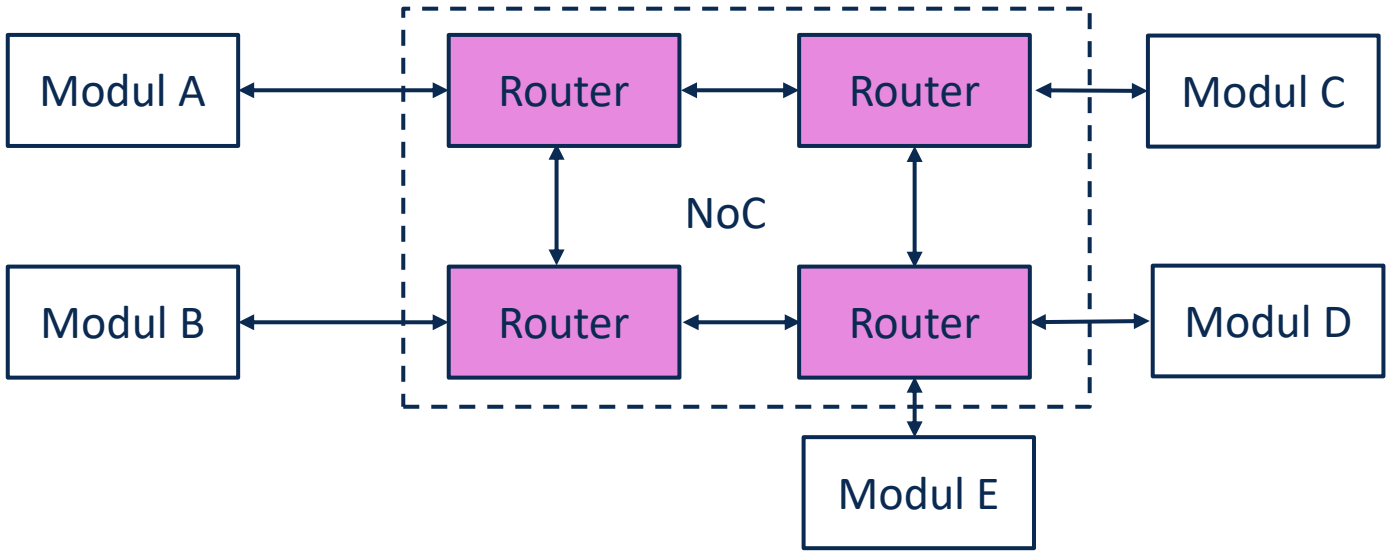
Vor- und Nachteile von Crossbar Bussen

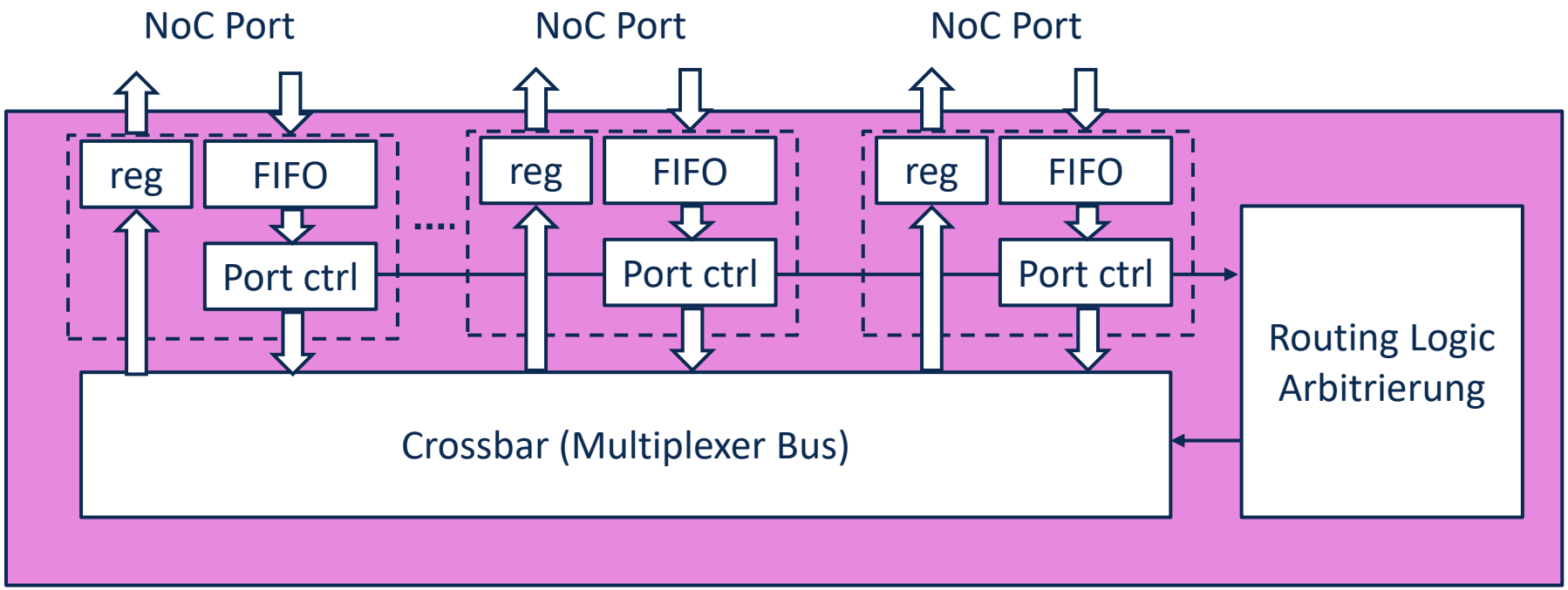
- Vorteile:
  - Standardisierte Busschnittstelle
  - Geringer Latenz (wenige Takte)
  - Deterministische Latenz
  - Einfach zu implementieren (synchrones Design)
  - Flexibel auf kleinen Chips
- Nachteile:
  - Nur logische Zuordnung, keine Berücksichtigung des Floorplans
  - Lokal synchrone Taktung für „globale Verdrahtung“ auf dem Chip
  - Hoher Aufwand an Chipfläche und Verlustleistung
  - Kann die maximale Taktfrequenz limitieren.
- Beispiel: Music 2 SIMD Cluster



Infineon/IMC MuSiC2 SIMD Cluster (conv. interconnect)

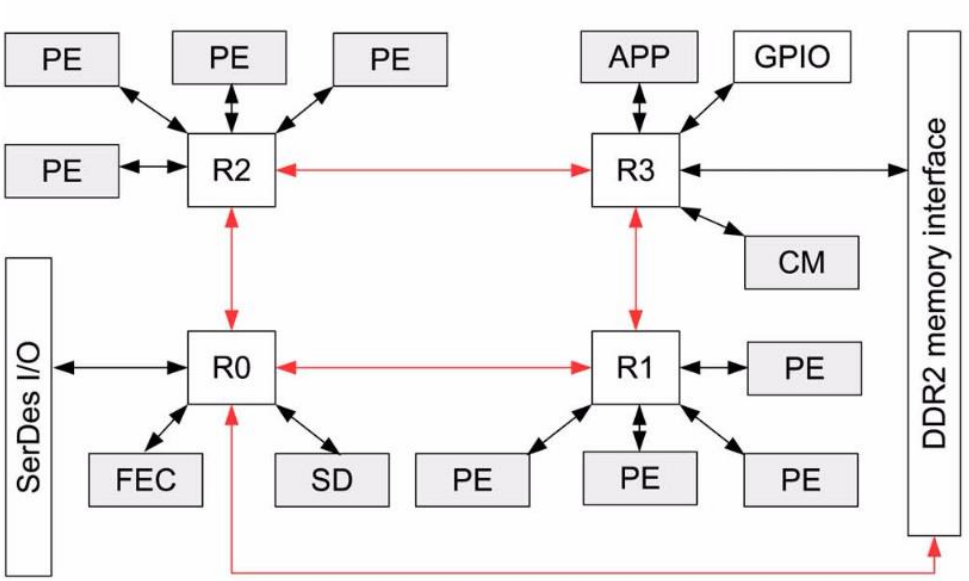
- „Packet Switched Network“ → Routing von Paketen durch das Netzwerk (Router)
- Flexible Topologien möglich
- Standardisiertes Paketformat und Interface der Komponenten
- Hohe Performanz (Datendurchsatz, Latenz) in komplexen Systemen durch Parallelität
- Global asynchrone Realisierung möglich, individuelle Takte der einzelnen Komponenten

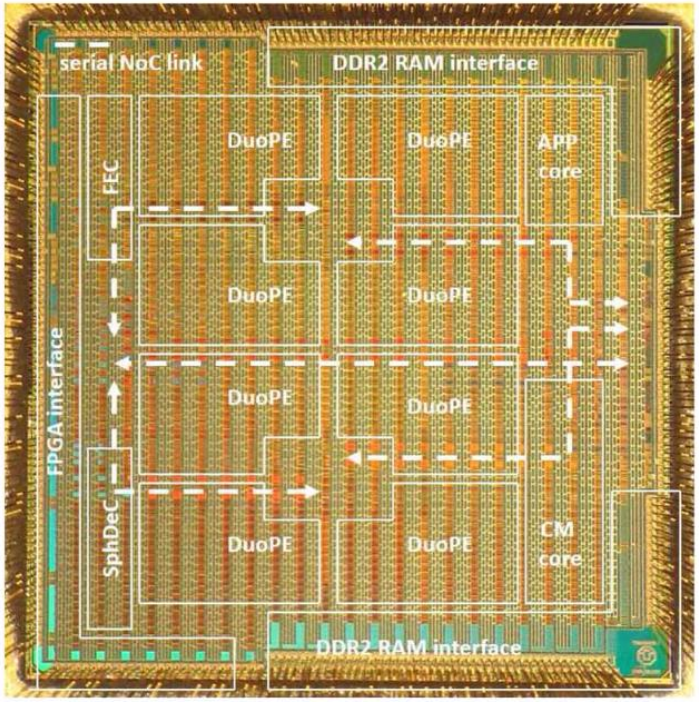




Beispiel: Tomahawk 2 NoC

- Tomahawk2: Software-Defined Radio Basisbandprozessor, entwickelt von TUD-MNS und TUD-HPSN
- Vernetzung der Systemkomponenten in einem NoC
- Punkt-zu-Punkt Verbindungen mit schnellen seriellen Links → kompaktes Layout



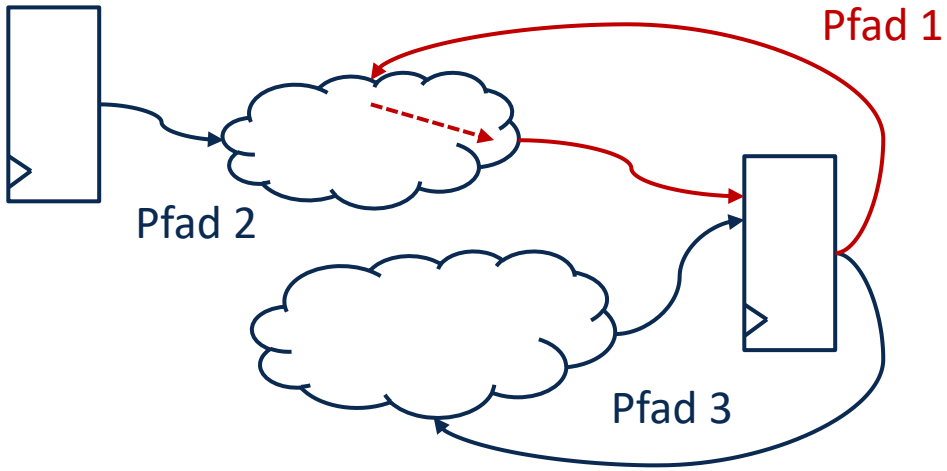


Höppner, Sebastian and Walter, Dennis and Hocker, Thomas and Henker, Stephan and Hänzsche, Stefan and Sausner, Daniel and Ellguth, Georg and Schlüssler, Jens-Uwe and Eisenreich, Holger and Schüffny, René, An Energy Efficient Multi-Gbit/s NoC Transceiver Architecture With Combined AC/DC Drivers and Stoppable Clocking in 65 nm and 28 nm CMOS, IEEE Journal of Solid State Circuits 50 (2015), no. 3, 749-762,

Optimierung von Datenpfaden

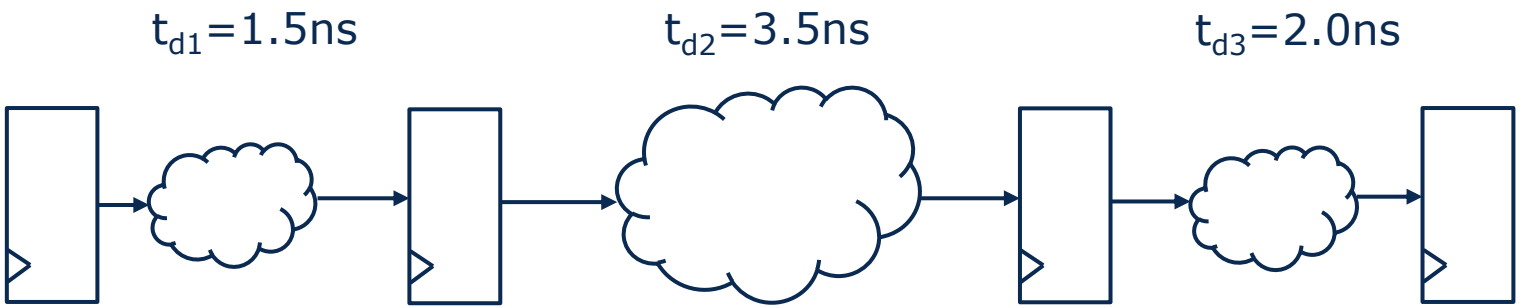
Anforderungen an Datenpfade

- Taktfrequenz und Datendurchsatz, gemessen in
  - [GHz]
  - Operationen pro Sekunde [GOPS]
- Chipfläche, gemessen in
  - [µm²]
  - NAND2 Äquivalenten (technologieunabhängige Normierung)
- Energie pro Rechenschritt, gemessen in
  - [pJ]



- Datenpfade besitzen viele Timing-Pfade mit Verzögerung t_{delay}
  - Startpunkt: Register Ausgang
  - Endpunkt: Register Eingang
- Taktfrequenz limitiert durch den **kritischen Pfad** ($f_{max} \approx 1/max(t_{delay})$)
- Oft sind nur wenige Datenpfadelemente kritisch für das Timing

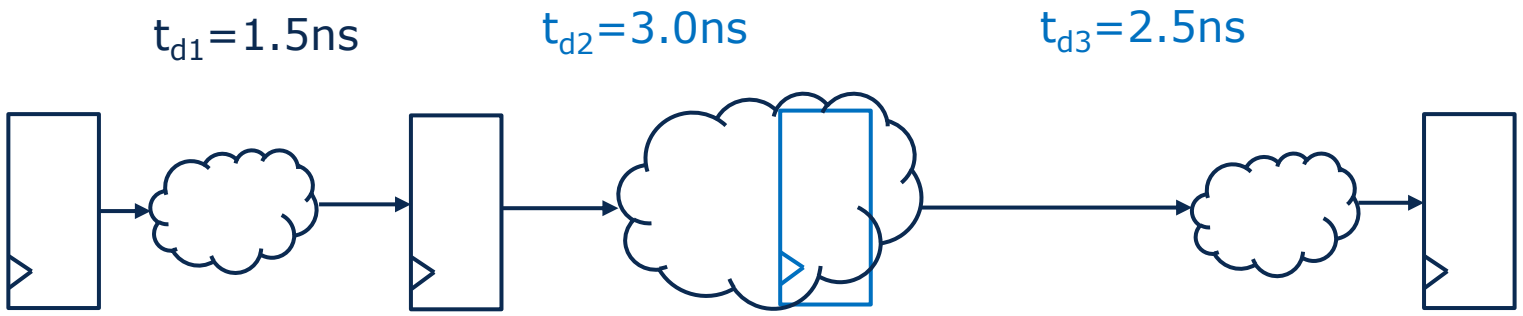
Taktfrequenz und Durchsatz



- $f_{max}=1/max(t_{delay})$ → 285MHz
- Latenz: 4 Taktzyklen
- Datendurchsatz: 285MHz/4=**71 MOP/s** (ohne Pipelining)

Optimierung von Taktfrequenz und Durchsatz

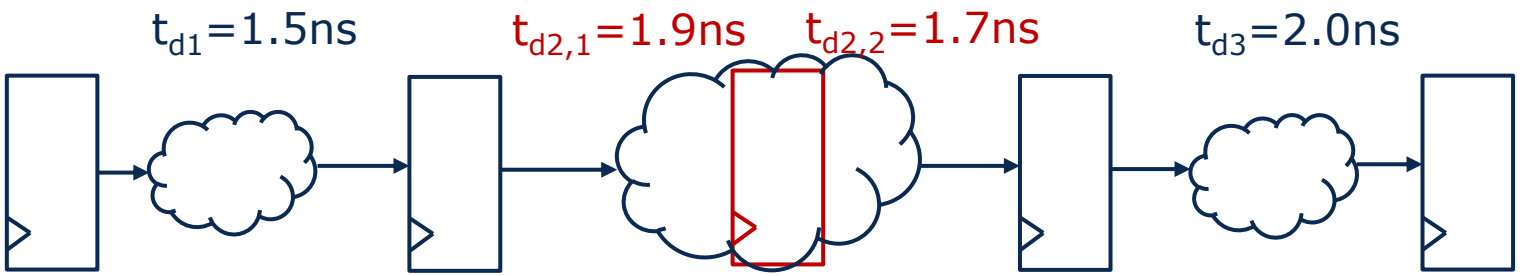
- Logic Re-Timing:



  - $f_{max}=1/max(t_{delay})$ → 333MHz
  - Latenz: 4 Taktzyklen
  - Datendurchsatz: 333MHz/4=**83 MOP/s** (ohne Pipelining)
- Kann automatisch von Synthesetools durchgeführt werden

Optimierung von Taktfrequenz und Durchsatz

- Einfügen von Registerstufen:

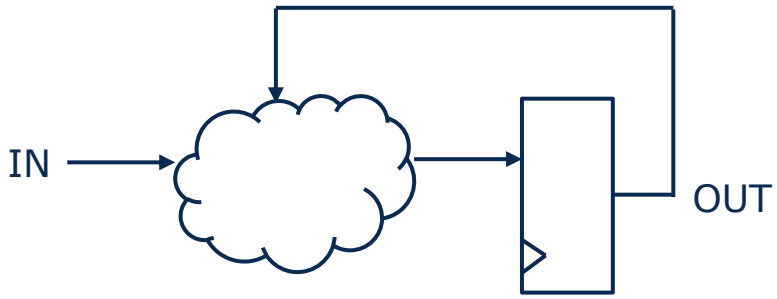


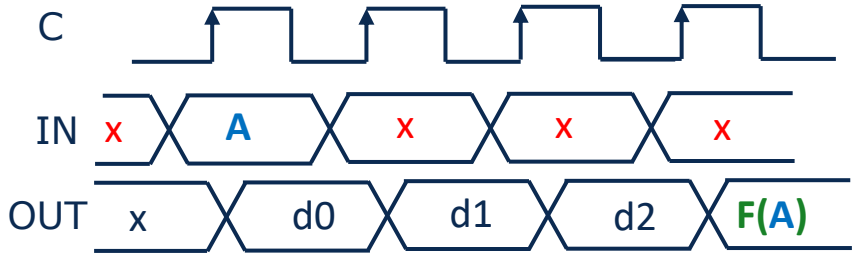
- $f_{max}=1/max(t_{delay})$ → 500MHz
- Latenz: 5 Taktzyklen
- Datendurchsatz: 500MHz/5=**100 MOP/s** (ohne Pipelining)

- Arithmetische Schaltungs-IP (RTL) oft konfigurierbar hinsichtlich der Anzahl benötigter Takte (z.B. Synopsys Design Ware)

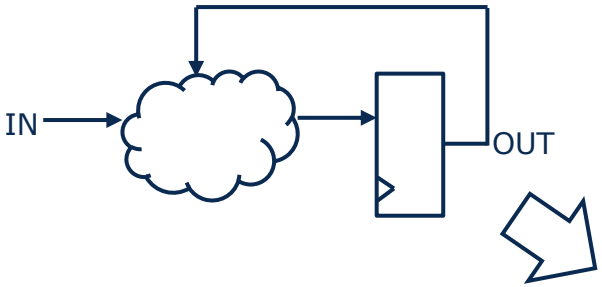
Sequentielle Datenpfad Baublöcke

- Realisierung von sequentiellen Abläufen
- Register Werte in Zeitschritt i sind Grundlage für Berechnung in Schritt i+1
- Vorteile:
  - Kürzere Logiklaufzeiten (Berechnung in mehreren Taktzyklen)
  - Geringer Hardware Aufwand (Wiederverwendung von Baublöcken in unterschiedlichen Taktzyklen)
- Nachteile:
  - Längere Berechnungsdauer

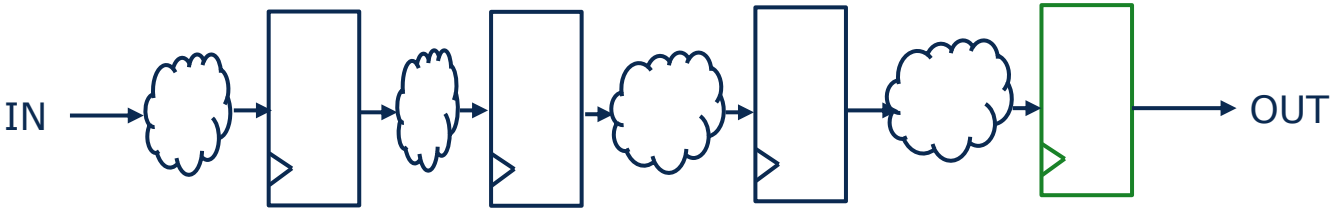


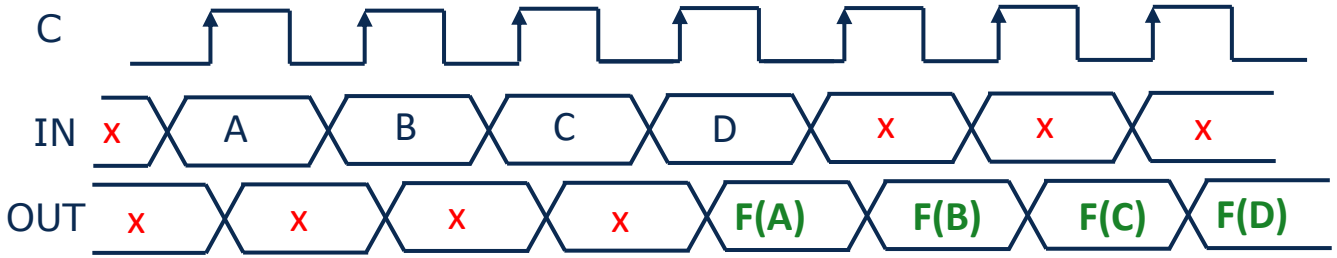


- Latenz N Takte
- Datenrate 1/N Operationen pro Takt



- **Parallele** Abarbeitung von Teilaufgaben (Pipelining), durch:
  - Aufspalten der Logik pro Rechenschritt in **separate** Schaltungen („Loop Unrolling“)
  - Oder: Einfügen von Registerstufen in kombinatorische Arithmetische Baublöcke





- Latenz N Takte
- Datenrate 1 Operation pro Takt

Zusammenfassung Datenpfade

- Vorstellung von
  - Datenpfadbaublöcken
  - Speicherarchitekturen
  - Bussystemen
- Timing Optimierung von sequentiellen Datenpfadbaublöcken
  - Einfügen von Registerstufen
  - Sequential Re-Timing
  - Pipelining