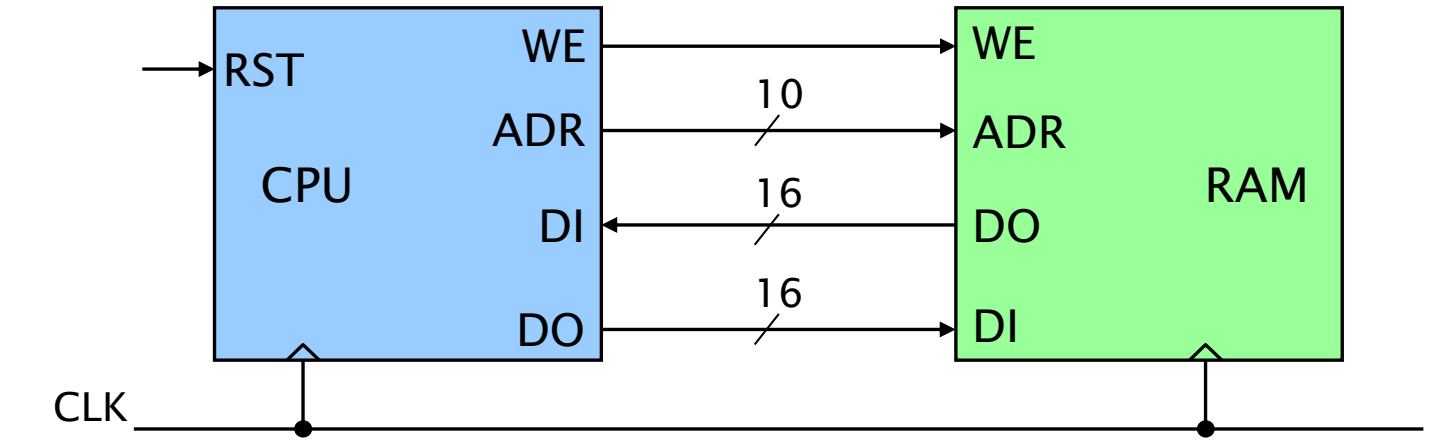
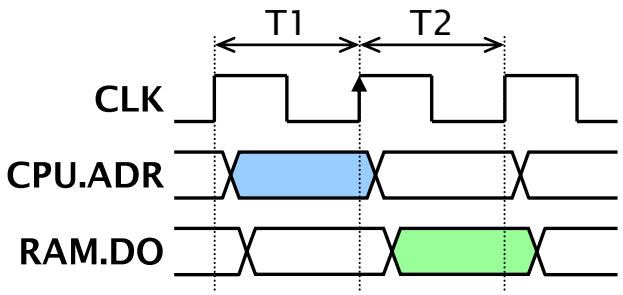
PicoBlaze: Einführung

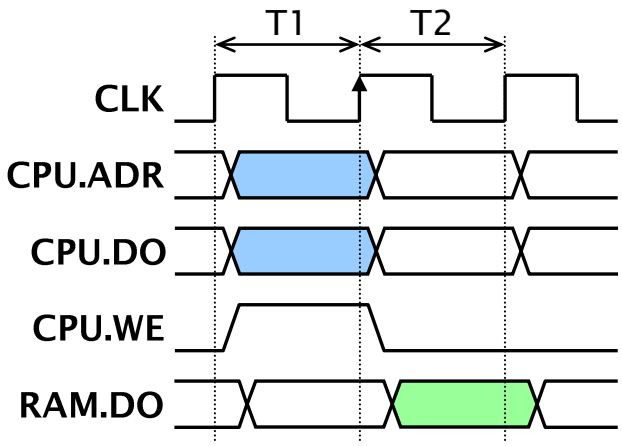
- System-on-Chip
  - Soft-Core-Prozessor
  - Programm-/Datenspeicher
  - Peripheriekomponenten (UART, Multiplizierer, ...)

- Zieltechnologie
  - FPGA-Baustein vom Typ Spartan3 von Xilinx
  - RAM-Block als Programm-/Datenspeicher
    - Kapazität: 18 Kbit
    - Organisation: 1024 x 16 (+2 Parity-Bit)
    - Zugriffszeit: 5 ns (200 MHz)
    - Zugriffe: taktsynchron
    - Datenbus: unidirektional





Lesezyklus



Schreibzyklus

PicoBlaze: Einführung

- Soft-Core-Prozessor:
  - ein in einer HDL modellierter Prozessor
  - eingebettet in einem FPGA/CPLD

  - technologieunabhängige Soft-Core-Prozessoren:
    - 8-Bit-CPU: 8051, 68HC11, ...
    - 32-Bit-CPU: SPARC, DLX, ...

  - technologieabhängige Soft-Core-Prozessoren:
    - Altera: Nios II (32-Bit-RISC)
    - Xilinx: MicroBlaze (32-Bit-RISC)
    - Actel: ARM (32-Bit-RISC)
    - Lattice: Mico8 (8-Bit-RISC)
    - Xilinx: PicoBlaze (8-Bit-RISC)
    - Cypress: M8C (8-Bit-RISC)

PicoBlaze: Einführung

- PicoBlaze = KCPSM: (K)constant Coded PSM
  - PSM: Programmable State Machine

KCPSM	Version CR	Version 1	Version 2	Version 3
Technologie (FPGA/CPLD)	CoolRunner	Virtex-E Spartan-II/E	Virtex-II	Spartan-3 Virtex-II/Pro Virtex-4
Anzahl der Register	8	16	32	16
Befehls-wortlänge	16	16	18	18
Anzahl der Befehle	49	49	49	57
Programm-speichergröße	256	256	1024	1024

PicoBlaze: Allgemeine Merkmale

- 8-Bit-RISC-Kern:
  - basiert auf 2-Adreßorgnisation
  - reduzierter Befehlssatz mit insgesamt 57 Befehlen:
    - 8 arithmetisch/logische Befehle (ADD, SUB)
    - 10 Schiebe-/Rotationsbefehle (SLA, RL)
    - 4 Transportbefehle (STORE, FETCH)
    - 3 Steuerflussbefehle (JUMP) und
    - 4 Interrupt-Befehle (RETURNI).
  - weitgehend einheitliches Befehlsformat
  - alle Befehle mit fester Länge von 18 Bit
  - konstante Ausführungszeit von zwei Takten für alle Befehle und unter allen Bedingungen
    - Einsatz in harten Echtzeit-Anwendungen möglich
    - z.B. 25 MIPS bei einer Taktfrequenz von 50 MHz

PicoBlaze: Allgemeine Merkmale

- 8-Bit-RISC-Kern:
  - orthogonaler Registersatz mit 16 8-Bit-Registern
  - wenige Adressierungsarten:
    - Registeradressierung (ADD sX, sY)
    - register-indirekte Adressierung (INPUT sX, (sY))
    - unmittelbare Adressierung (ADD sX, kk)
    - direkte Adressierung (INPUT sX, pp)
    - absolute Adressierung (JUMP NZ, aaa) und
    - implizite Adressierung (RETURN)
  - eine Load-/Store-Architektur mit getrennten Speicherbereichen für Daten und Programme
  - einfache und flexibel erweiterbare Ein-/Ausgabe-Schnittstelle
  - ein Einebenen-Interrupts-System

PicoBlaze: Technologische Merkmale

- Strukturbeschreibung
  - in VHDL (kcpsm3.vhd) und Verilog (kcpsm3.v)
  - bestehend aus Makro-Komponenten und aus primitiven FPGA-Komponenten:
    - 109 Look-Up-Tabellen (LUT1..LUT4)
    - 76 Flip-Flops (FD, FDE, ...)
    - Multiplexer und Carry-Logik (MUXCY, XORCY)
    - verteilte Speicherblöcke (RAMxxxx)

LUT1 : 2	FD : 24	MUXCY : 39	RAM16X1D : 8
LUT2 : 6	FDE : 2	MUXF5 : 9	RAM32X1S : 10
LUT3 : 68	FDR : 30	XORCY : 35	RAM64X1S : 8
LUT4 : 33	FDRE : 8		
	FDRSE : 10		
	FDS : 2		

PicoBlaze: Technologische Merkmale

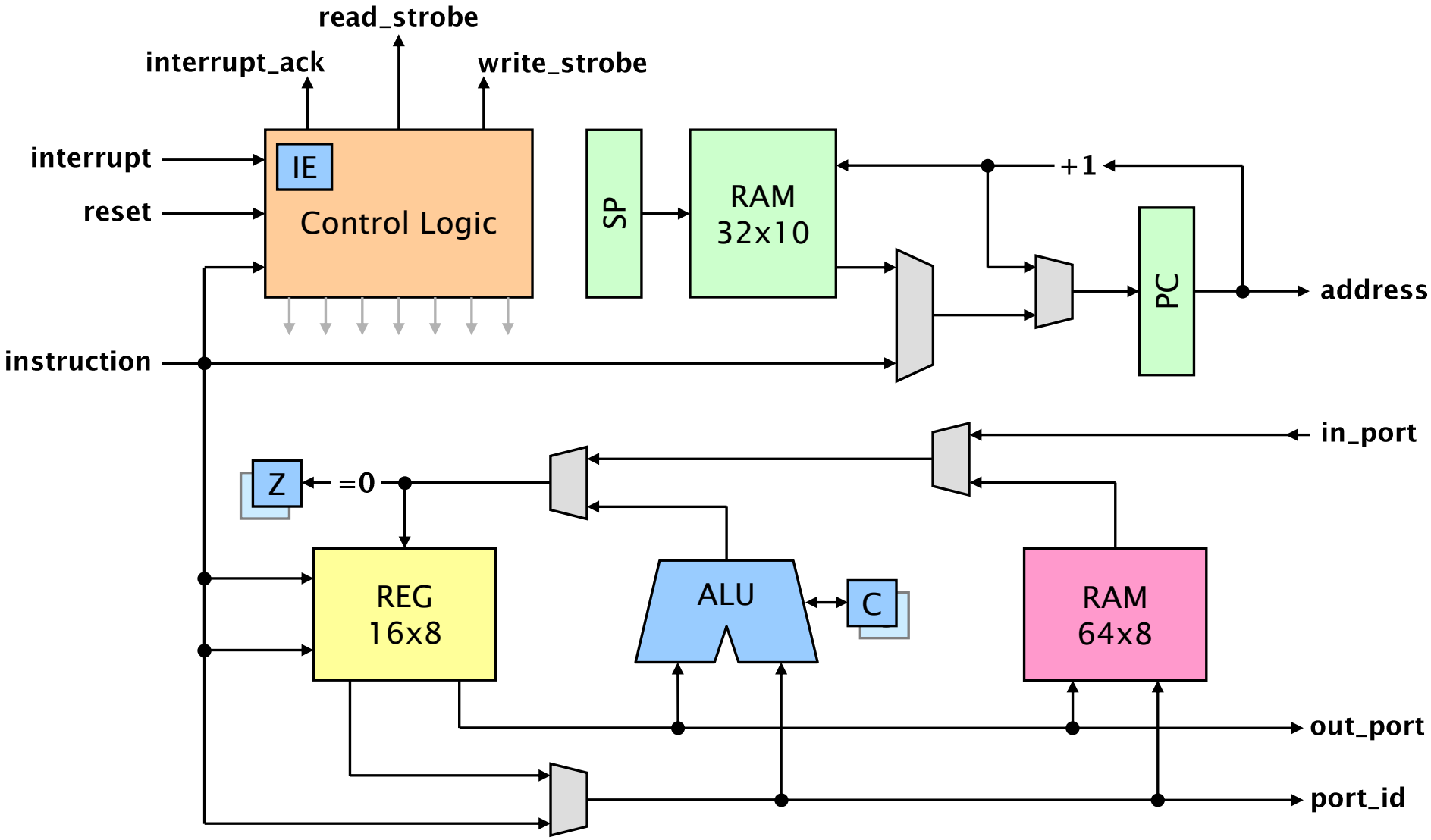
- Rechenwerk und Steuerwerk sind auf Laufzeit und Ressourcen optimiert
- zwei Takte pro Befehl
- feste Größe: 96 Slices
  - ca. 5% eines XC3S200-Bausteins
  - + 1 RAM-Block (1k x 18) als Programmspeicher
- schwer portierbar auf andere FPGA-Architekturen
- relativ leicht erweiterbare Architektur
  - Größe des Scrachtpad-Speichers
  - Tiefe des Call/Return-Stacks
  - Bedingungen für das ZERO-Flag
  - Befehlssatz

PicoBlaze: Technologische Merkmale

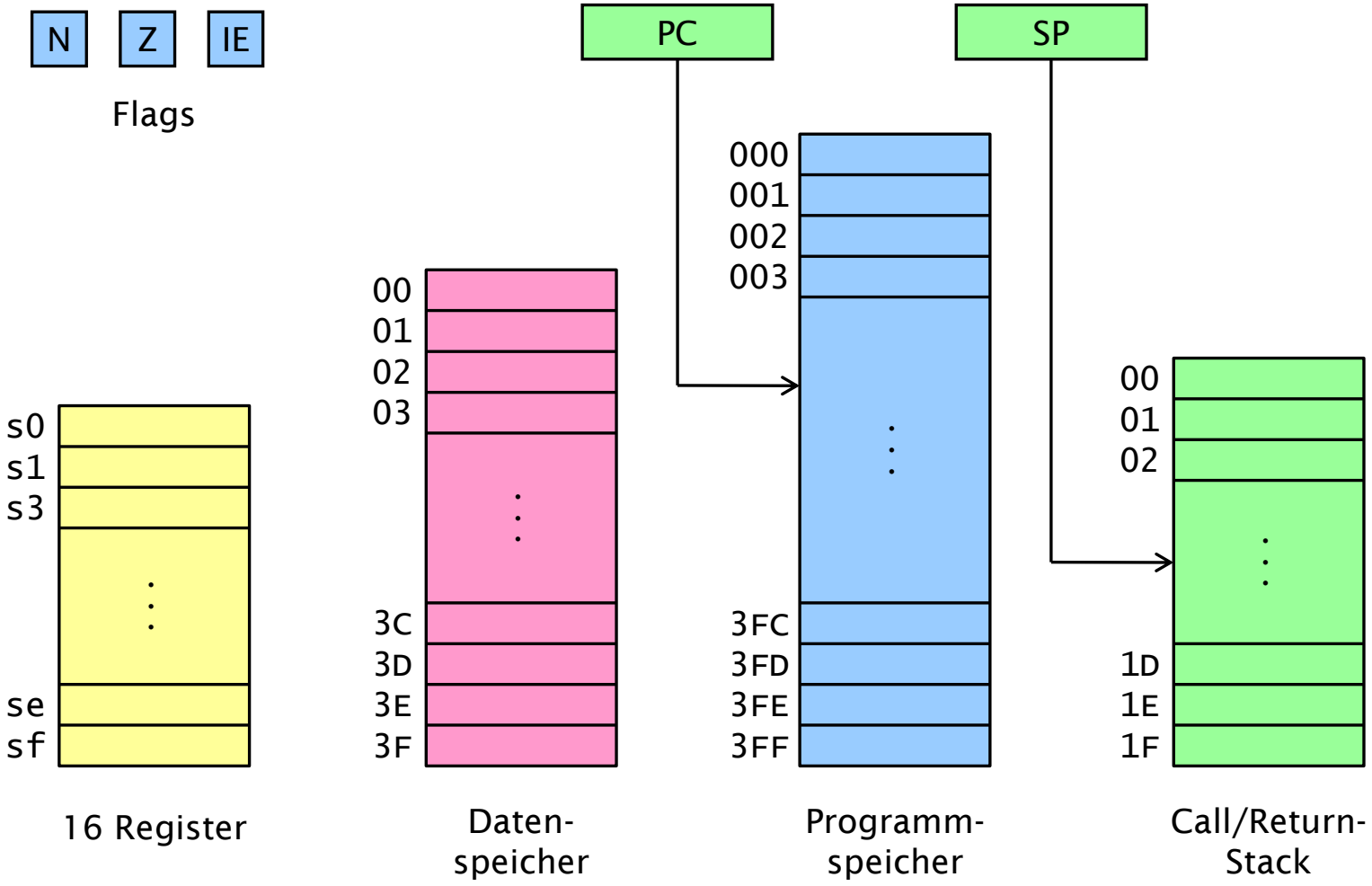
- Schnittstelle mit Port-Deklaration:

```
entity kcpsm3 is
    port (        reset : in  std_logic;
                    clk : in  std_logic;

                address : out std_logic_vector( 9 downto 0);
            instruction : in  std_logic_vector(17 downto 0);

                port_id : out std_logic_vector( 7 downto 0);
               out_port : out std_logic_vector( 7 downto 0);
                in_port : in  std_logic_vector( 7 downto 0);
           write_strobe : out std_logic;
            read_strobe : out std_logic;

              interrupt : in  std_logic;
          interrupt_ack : out std_logic);
end kcpsm3;
```

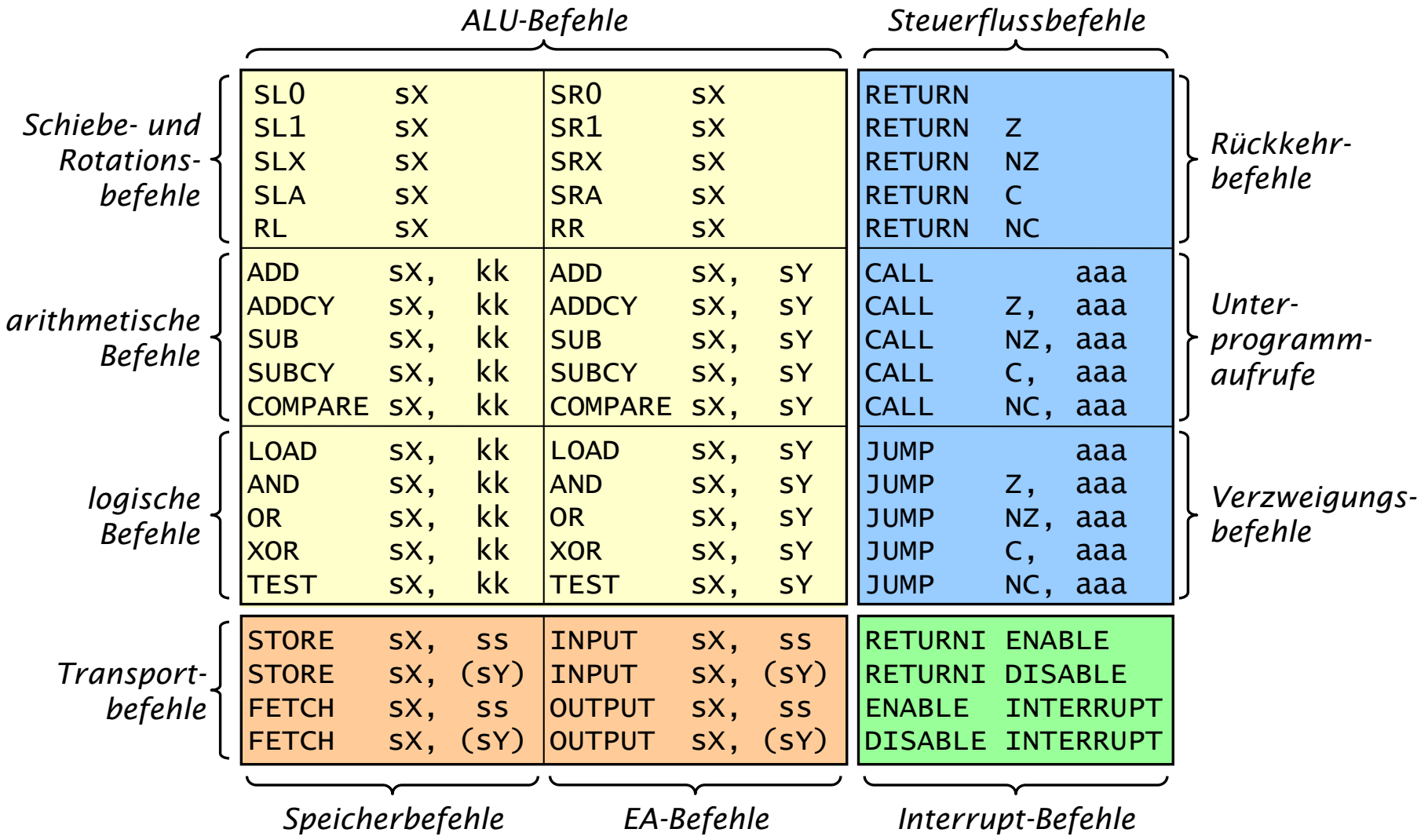
PicoBlaze: Prozessorstruktur



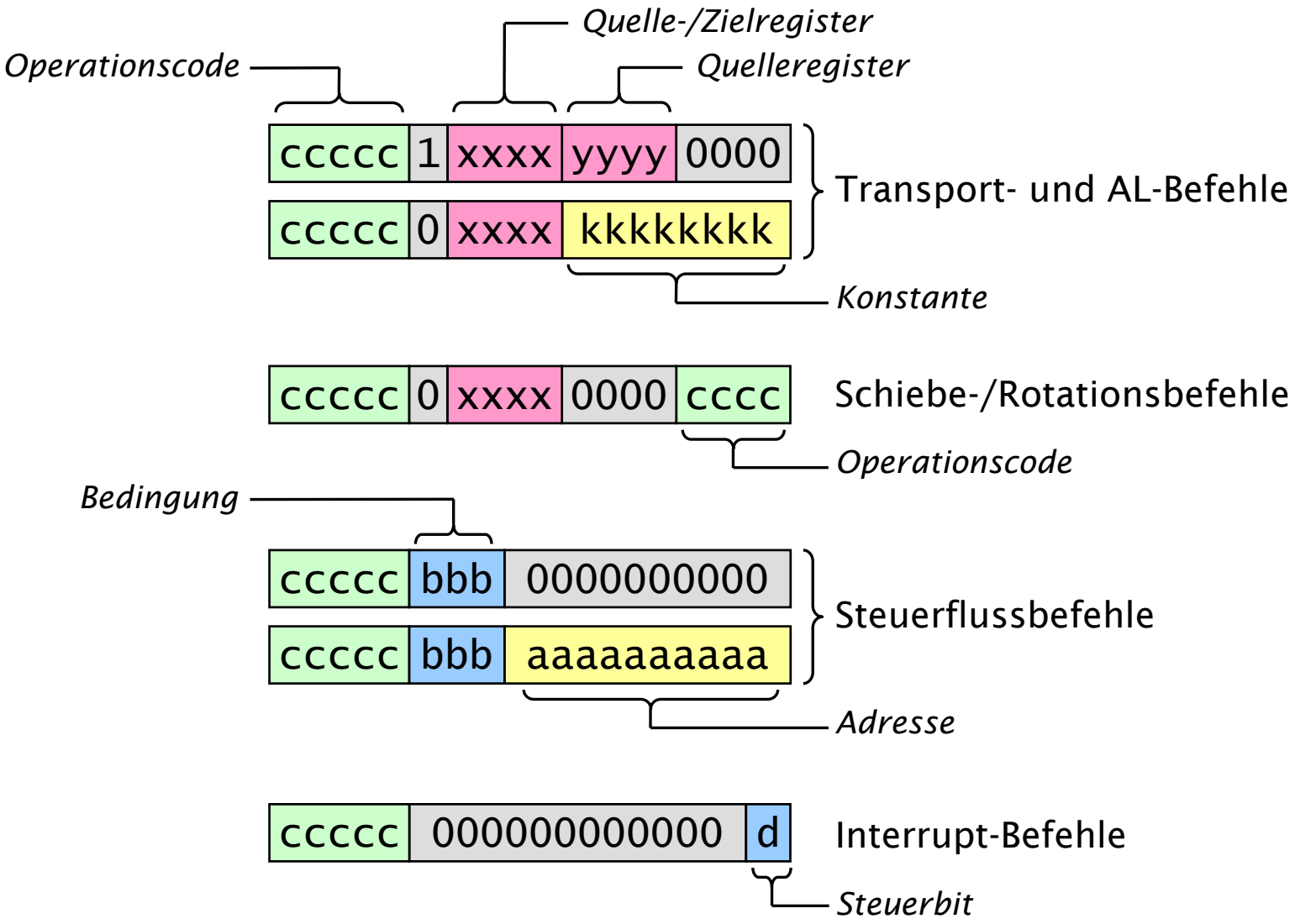
PicoBlaze: Programmiermodell



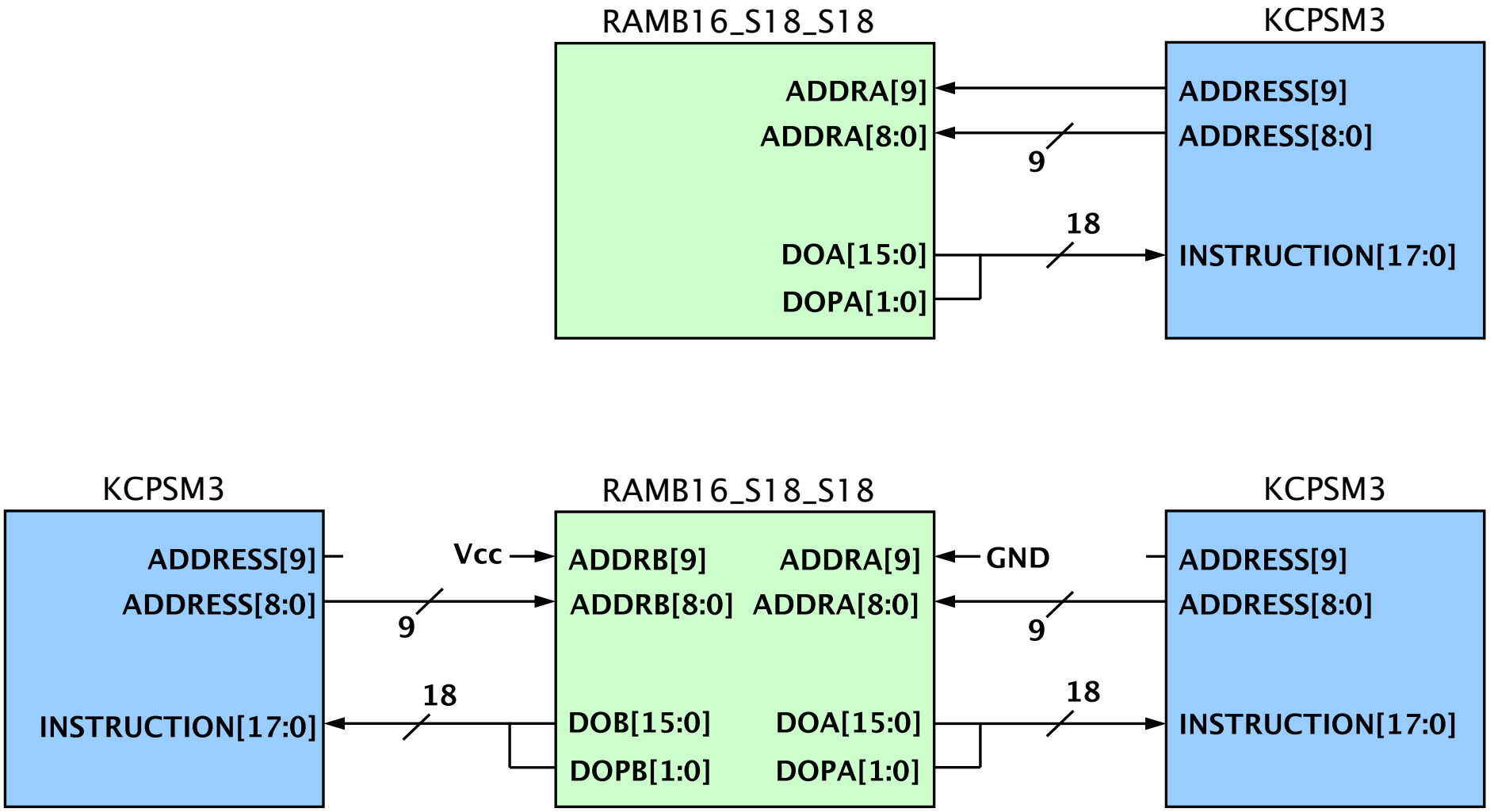
PicoBlaze: Assemblerbefehle



PicoBlaze: Befehlsformate

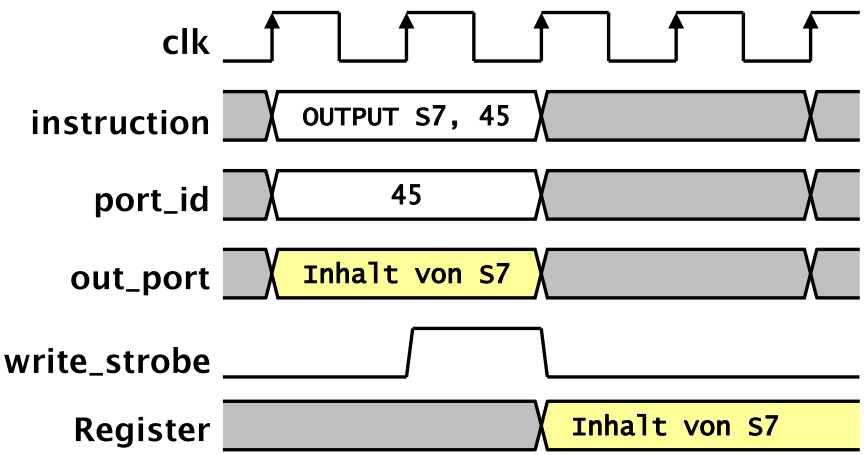


PicoBlaze: Single-/Dual-Core-Modus



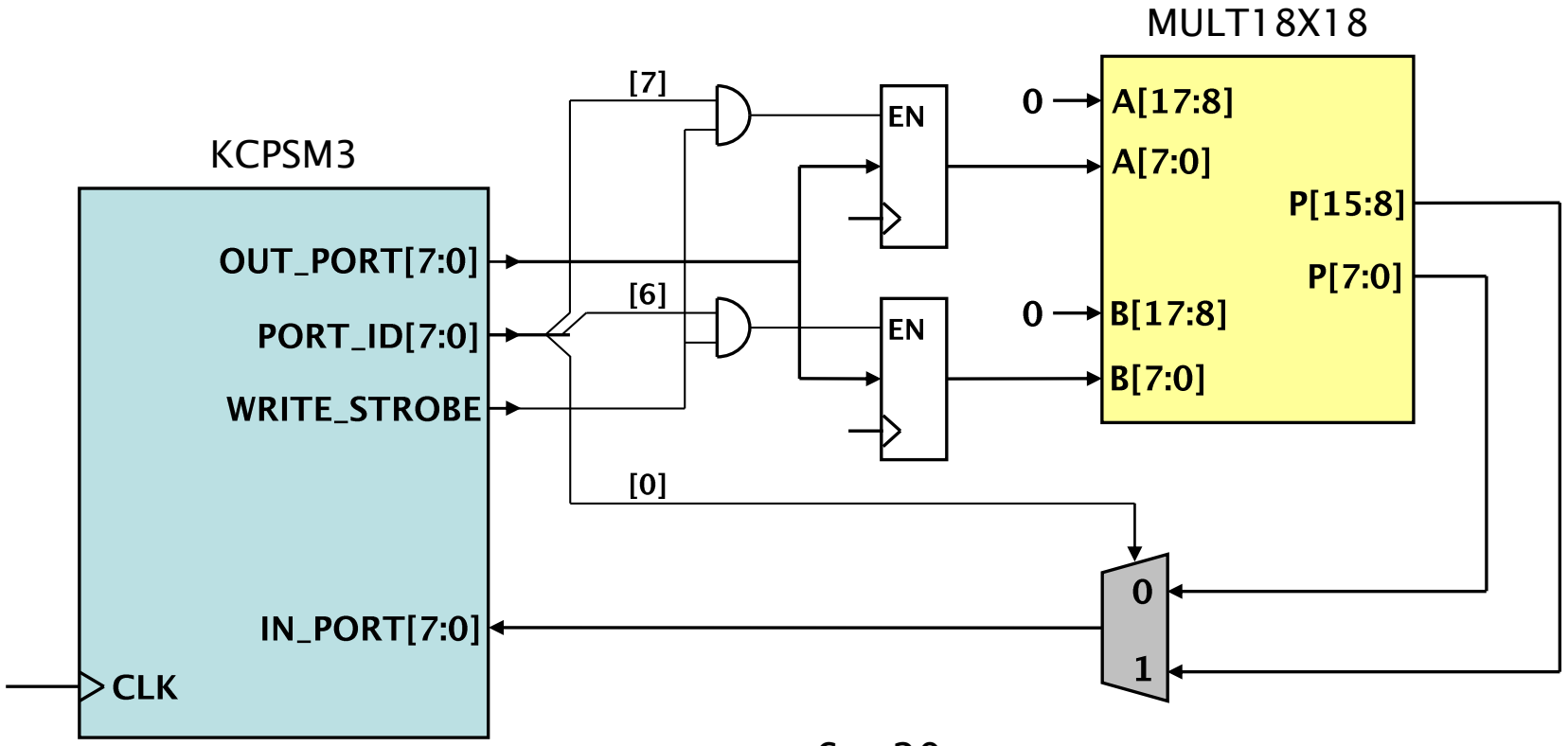
PicoBlaze: Ein-/Ausgabe-Konzept

- drei 8-Bit-Ports
  - Eingangsport IN_PORT
  - Ausgangsport OUT_PORT
  - Adressport PORT_ID



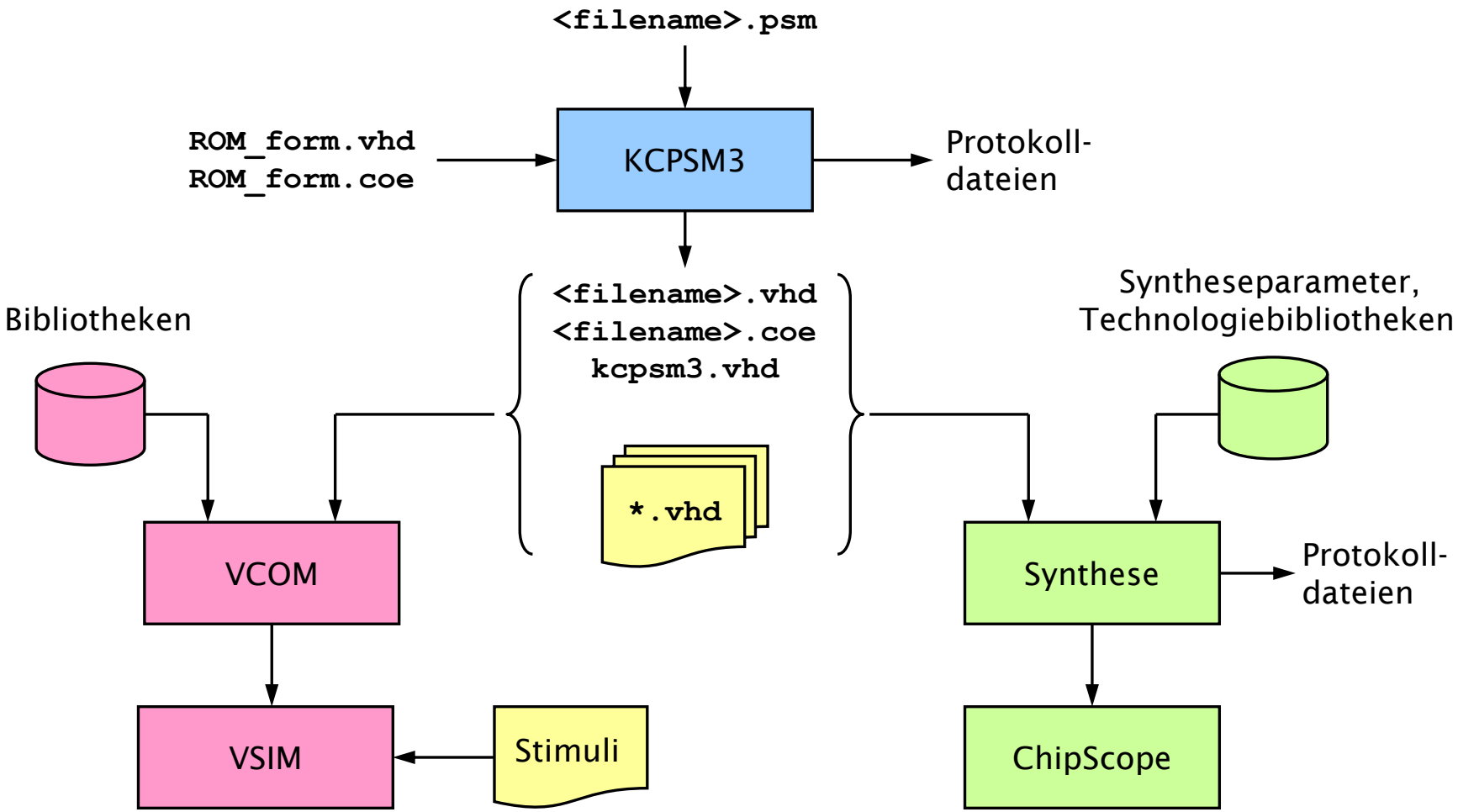
- zwei Steuersignale
  - READ_STROBE und WRITE_STROBE
  - aktiviert im zweiten Takt des Befehlszyklus von INPUT/OUTPUT
- flexible Erweiterbarkeit der I/O-Ports
  - 16-Bit-Ausgangsport durch Zusammenschaltung von Ausgangs- und Adreßport
  - über Adressport
    - acht Ziele bei „one-hot-encoding“
    - bis zu 256 Ziele mit einem Adreßdekoder

PicoBlaze: Ein-/Ausgabe-Konzept




```
LOAD    s6, 20
LOAD    s7, 17
OUTPUT  s6, 80
OUTPUT  s7, 40
INPUT   S8, 00
INPUT   S9, 01
```

PicoBlaze: Entwicklungsumgebung



Literaturquellen

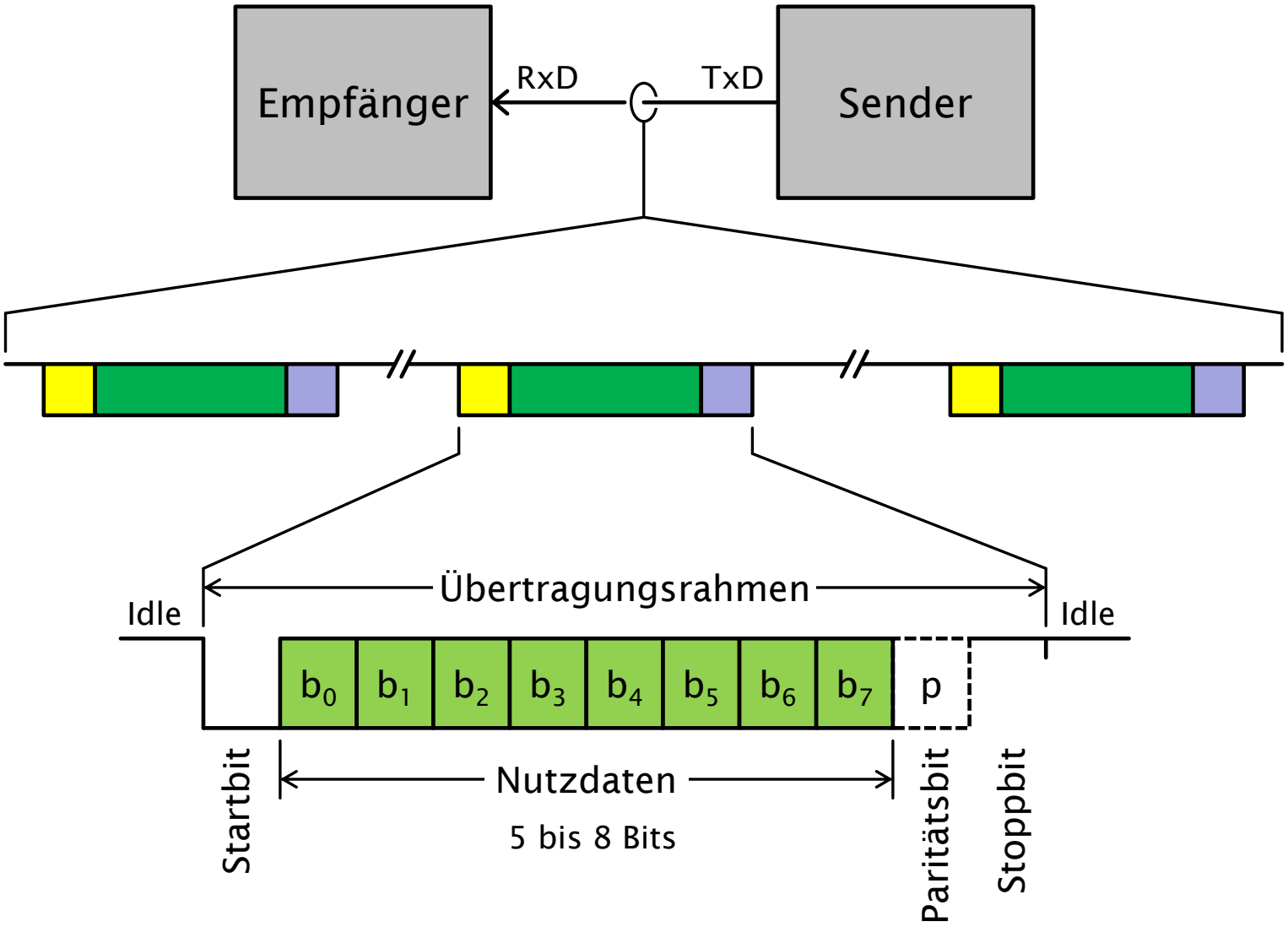
1. K. Chapman: *KCPSM3 Manual*, Xilinx Ltd., 2003
2. Xilinx: *PicoBlaze: 8-bit Embedded Microcontroller User Guide*, UG129, Xilinx Ltd., 2005
3. K. Chapman: *Creating Embedded Microcontrollers*, TechXclusive, Xilinx Ltd., 2002.
4. K. Chapman: *PicoBlaze: 8-Bit Microcontroller for Virtex-E and Spartan-II/IIE Devices*, App. Note XAPP213, Xilinx Ltd., 2003
5. K. Chapman: *PicoBlaze: 8-Bit Microcontroller for Virtex-II Series Devices*, App. Note XAPP627, Xilinx Ltd., 2003
6. Xilinx: *CryptoBlaze: 8-Bit Security Microcontroller*, App. Note XAPP374, Xilinx Ltd., 2003
7. Xilinx: *PicoBlaze: 8-Bit Microcontroller for CPLD Devices*, App. Note XAPP387, Xilinx Ltd., 2003
8. U. Waik: *FPGAs in Theorie und Praxis*, Elektronik 6/2006, 22/2006

Asynchron-serielle Datenübertragung

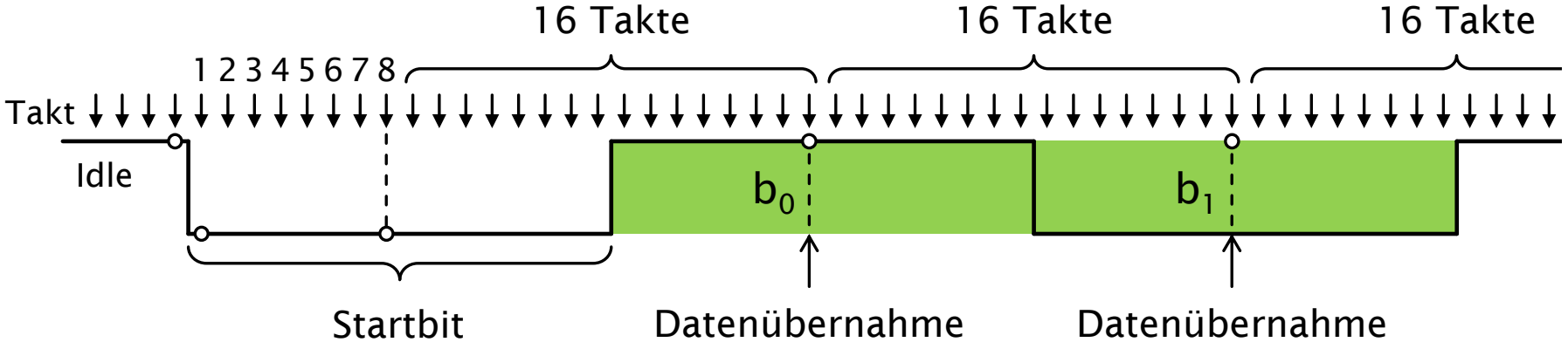
- im industriellen Umfeld weit verbreitet:
  - RS232: 1-zu-1-Verbindung
  - RS422/423: 1-zu-N-Verbindung
  - RS485: N-zu-M-Verbindung (Master/Slave), in sog. Feldbussen

- Eigenschaften (einfache Realisierbarkeit):
  - eine Datenleitung pro Übertragungsrichtung
  - Übertragung ohne eigene Taktleitung => im Empfänger Taktrekonstruktion aus dem Datenstrom notwendig
  - Baudrate (Übertragungsgeschwindigkeit) und Übertragungsrahmen (Datenformat) müssen zwischen Sender und Empfänger vereinbart sein
  - Einfache Erkennungsmechanismen des Übertragungsrahmens: ein Startbit und ein Stoppbit (Start- und Stoppsequenzen)

Asynchron-serielle Datenübertragung



- Datenempfang
  - 16fache Überabtastung des Datenstroms
    - Beispiel: Übertragungsgeschwindigkeit 9600 bps
      => Abtastfrequenz Takt = 16*9600 = 153,6 kbps (kHz)
  - Startbiterkennung
    - Übergang von High auf Low
    - Abfrage in der Mitte des Intervalls
      => wenn Signal=Low, dann Startbit erkannt, sonst Fehler



  - Datenübernahme immer in der Mitte eines Bit-Intervalls

Multitasking

- Merkmale „kleiner“ eingebetteter Systeme:
  - stark begrenzte Ressourcen:
    - niedrige Taktfrequenz (<< 10 MHz)
    - knapper Daten-/Programmspeicher (<< 8kB/64kB)
    - fehlende Peripheriekomponenten
  - ohne Betriebssystem
    - Programmierung direkt in C/Assembler
  - oft höhere Sicherheitsanforderungen
    - geschützte Speicherbereiche
    - stark deterministisches Softwareverhalten
  - mehrere Aufgaben (Tasks) mit unterschiedlichen Prioritäten (Wichtigkeitsgraden):
    - ereignisgesteuerte Aufgaben (asynchron und sporadisch: UART-Treiber/Task)
    - zeitgesteuerte Aufgaben (regelmäßig und wiederkehrend: Signalabtastung, Messwerterfassung/-verarbeitung)

Multitasking

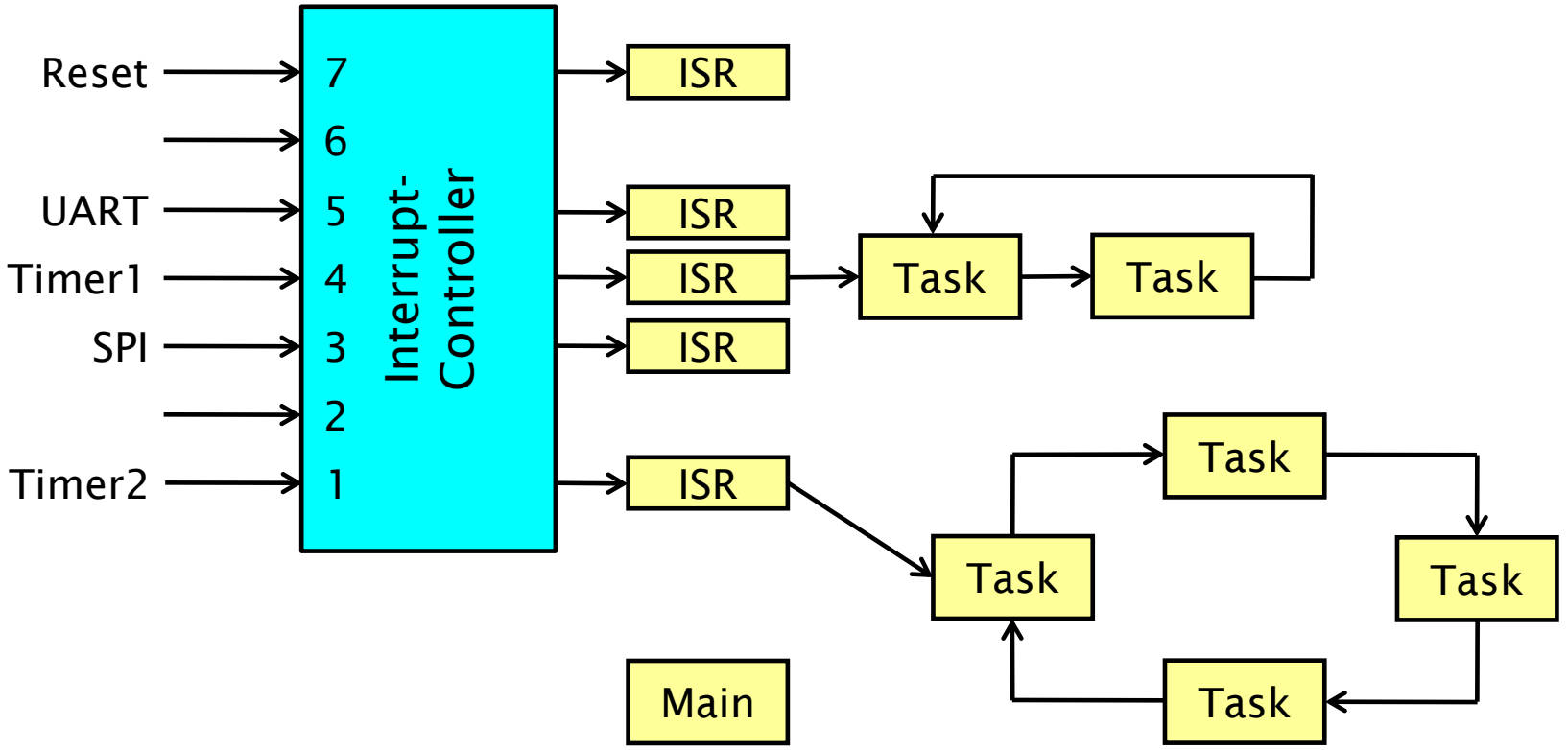
- Eingebettete Applikation:
  - basiert auf dem Multitasking-Prinzip
  - hat eine konstante und statische Anzahl an Tasks
    - Tasks können zur Laufzeit weder erzeugt noch gelöscht werden
  - kommt ohne dynamische Speicherverwaltung aus
  - lässt alle gleich priorisierten Tasks streng zeit-deterministisch (mit fester Reihenfolge) ablaufen (kooperatives Multitasking):
    - keine Task darf eine gleich priorisierte Task überholen
  - hat einen Zeitgeber mit feiner Auflösung
    - jede Task bekommt einen gleich langen Time-Slot zur Verfügung
    - Tasks mit langen Laufzeiten müssen ggf. in Software durch Aufteilung in Zustände angepasst werden.

- Eine Task:
  - ist ein Unterprogramm (in C: eine parameterlose Funktion ohne Rückgabewert), dessen Aufruf durch das Betriebssystem (Dipatcher, Scheduler) verwaltet wird,
  - bekommt beim Compilieren eine feste Priorität zugewiesen, die sich zur Laufzeit nicht ändert,
  - schützt/versteckt lokale Daten-/Programm-/Stack-Bereiche, auf die nur sie lesend/schreibend zugreift
  - lokale Daten-/Programm-/Stack-Bereiche anderer Tasks sind nicht sichtbar/zugreifbar
  - weißt am besten, was sie zu tun hat, und wie sie das macht
    - kommt am besten ohne gemeinsame Betriebssystemfunktionen aus
    - realisiert Abfragen von Ereignissen durch Polling, nicht durch Signale/Alarme/Events vom Betriebssystem

Multitasking

- Eine Task:
  - kann während der Laufzeit einen von drei Zuständen haben:
    - *ready*: eine Task ist inaktiv (aber bereit)
    - *running*: eine Task wird gerade ausgeführt
    - *interrupted*: eine Task wurde in ihrer Ausführung durch eine höher priorisierte Task/ISR unterbrochen
  - kommuniziert i.d.R. mit anderen Tasks nach einem einheitlichen Prinzip:
    Message-Passig (mit nur zwei Funktionen send() und receive())
  - braucht i.d.R. keine Synchronisationsmaßnahmen wie Semaphore
  - wird als Zustandsautomat implementiert

- Scheduler als Interrupt-Controller



- Implementierung eines Zustandsautomaten als
  - verschachtelte switch-/case-Anweisungen
    - variable Zugriffszeit, stark vom Compiler abhängig (Tabelle/Verzweigungsbäume)
    - bei vielen Zuständen oder Abfragen oft unübersichtlich
  - Tabelle mit Funktionszeigern
    - konstante Zugriffszeit
    - relativ hoher Speicheraufwand
    - 10 Bedingungen x 20 Zustände =>
      200 Funktionszeiger x 2 Byte/Zeiger => 400 Bytes
    - oft dünn besetzt => Komprimierung notwendig
  - Zustandsvariable mit Funktionszeigern
    - Jeder Zustand wird als separate Funktion beschrieben
    - nur funktionslokale switch-Anweisungen
    - leicht zu implementieren und übersichtlich
    - konstante Zugriffszeit

- Realisierung mit einem Ein-Ebenen-Interruptssystem