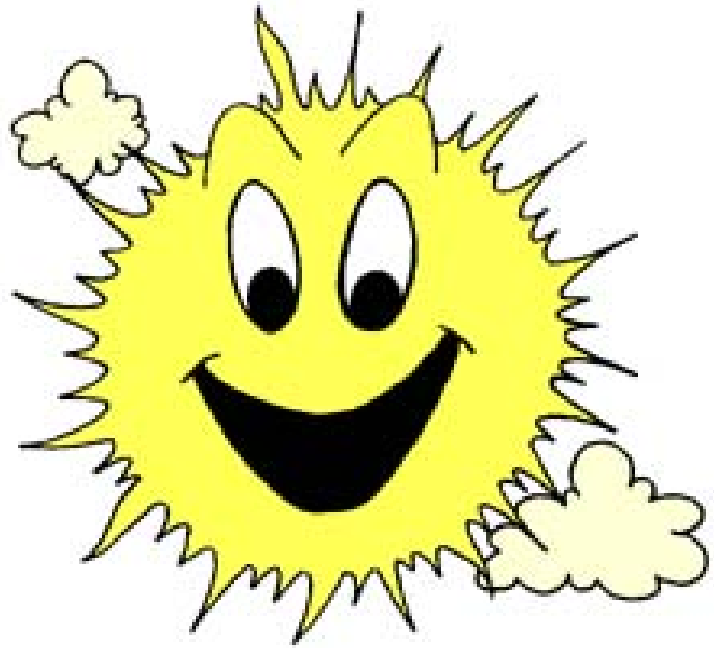
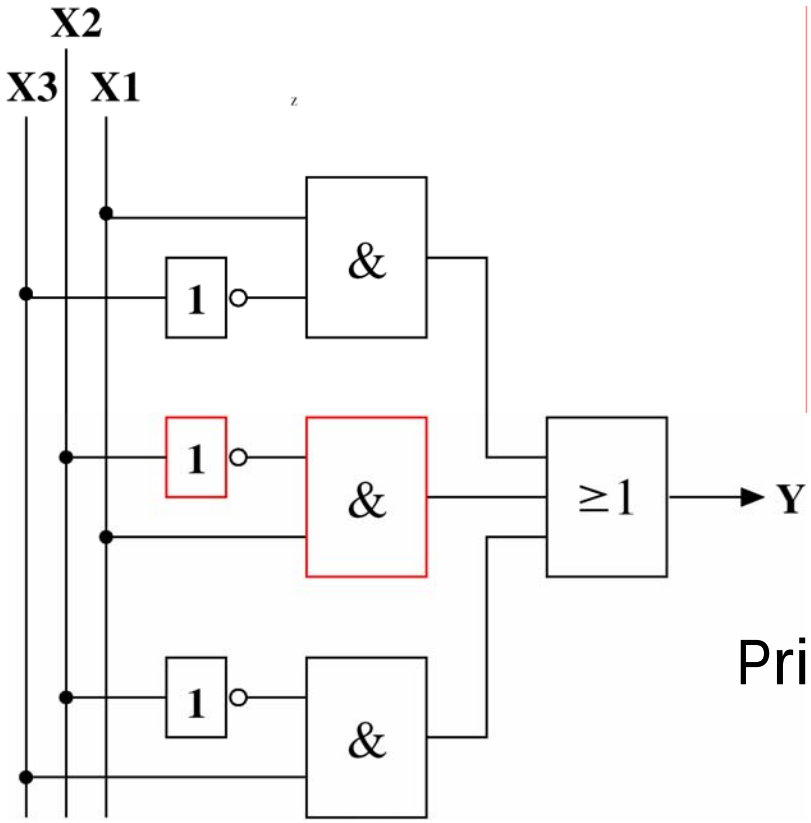
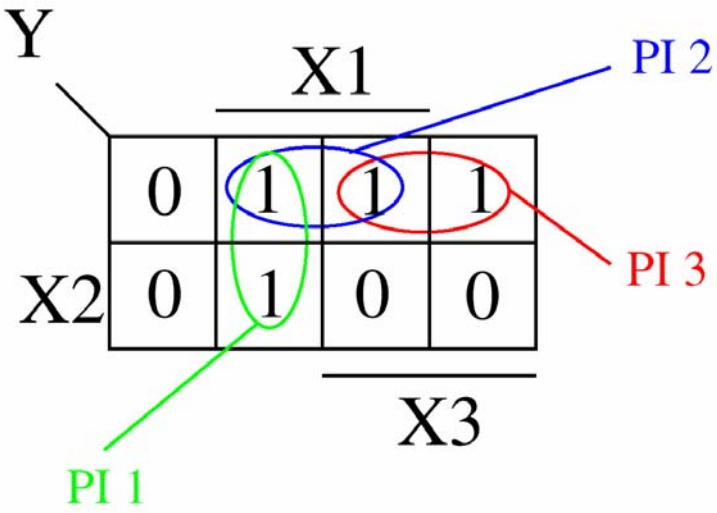
Wiederholung der 7. Vorlesung



12.1 Logik-Hazards – Beispiel Lösung





Übergang von 101 zu 001

Primimplikant 2 wird (trotz Redundanz) schaltungstechnisch realisiert.

Logik-Hazards werden auch als kombinatorische Hazards bezeichnet.

ad RG 2.2 Aiken Code

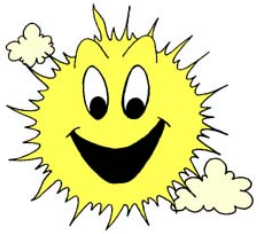
Dezimal	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
	0	1	0	1
	0	1	1	0
	0	1	1	1
	1	0	0	0
	1	0	0	1
	1	0	1	0
5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

Aiken-Code (0 - 4)

Pseudo-tetraden

Aiken-Code (5 - 9)

Ende der Wiederholung



12.1 Vermeidung von Logik-Hazards

- Erzeugung Minimalform.
- Wenn bei dem Übergang einer Komponente der aktive Primimplikant gewechselt wird, kann ein redundanter Primimplikant hinzugefügt werden, so dass der Übergang eliminiert wird.
- Einsatz von getakteten Schaltungen.

12.2 Funktions-Hazards

- werden auch als **Race** (Wettlauf) bezeichnet.
- können auftreten, wenn sich mehr als ein Eingang gleichzeitig ändert (Mehrkomponenten-Übergang)
- Das Ergebnis hängt von dem Ausgang des Wettlaufs ab.
- Dies kann zu Fehlfunktionen im System führen.

12.2 Vermeidung von Funktions-Hazards

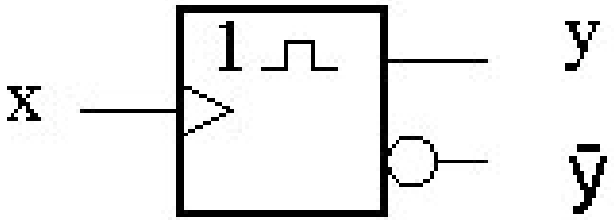
- Überführung von Mehrkomponenten-Übergängen in mehrere Einkomponentenübergänge (Änderung einer einzelnen Eingangsvariablen)
- Einsatz von getakteten Schaltungen

12.2 Zusammenfassung – Fazit: Hazards

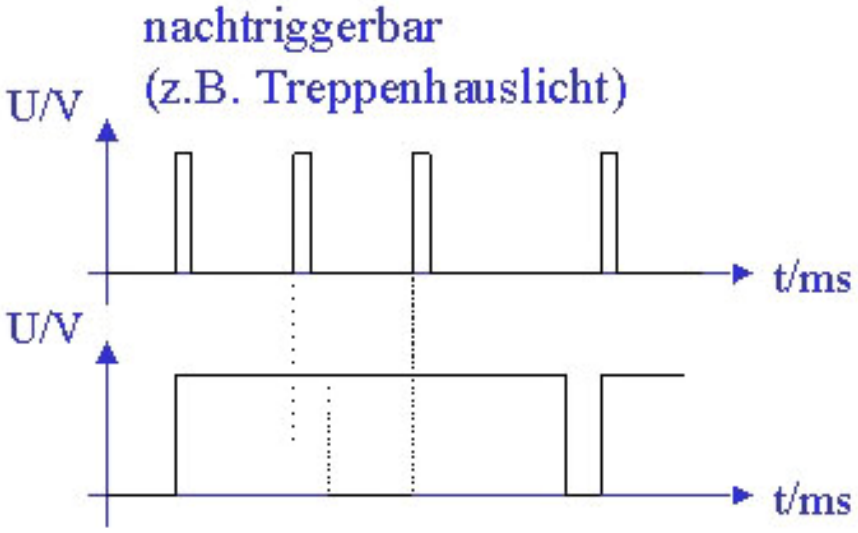
- Bei komplexen asynchronen Schaltungen ist die Wahrscheinlichkeit für Hazards hoch.
- Analyse von asynchronen Schaltungen bezüglich Hazards ist aufwändig.

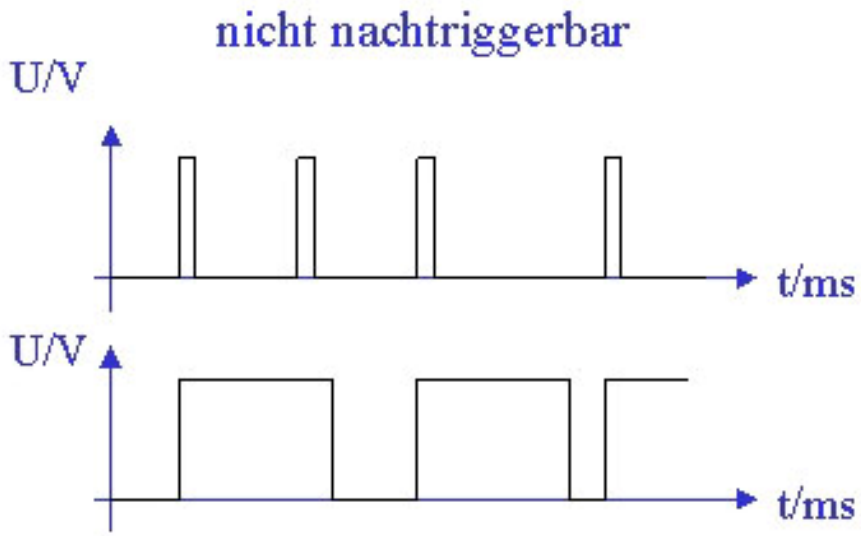
- In der Praxis dominiert vor diesem Hintergrund der Entwurf von synchronen Schaltungen

10.1 weitere Kippschaltungen: Monoflop

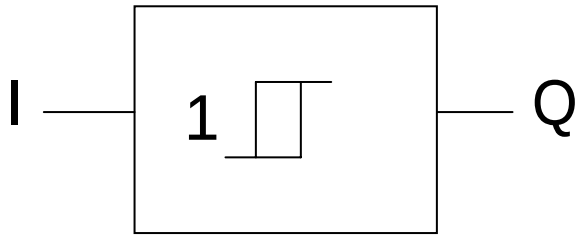


- wechselt auf Taktimpuls seinen Logikzustand
- kippt nach Verweildauer wieder zurück in Ausgangszustand
- Verweildauer wird meist durch externe Beschaltung mit R und C eingestellt.
- 2 Arten des Verhaltens

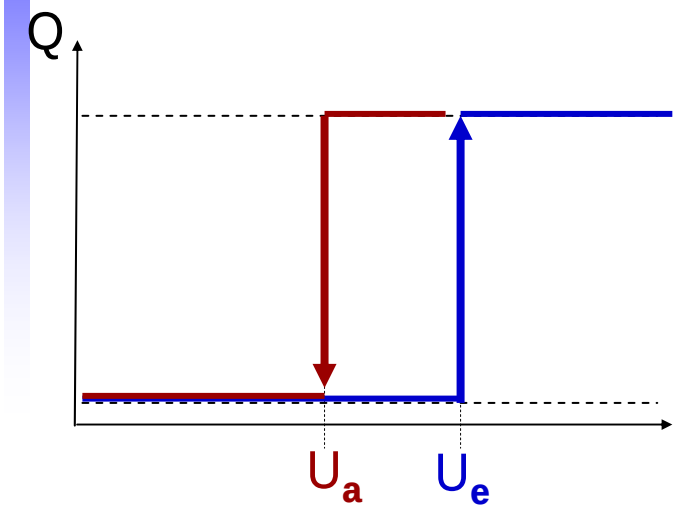


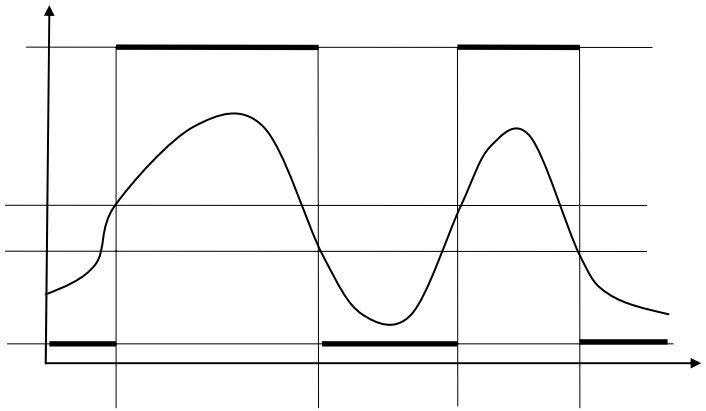


10.1 Schmitt-Trigger



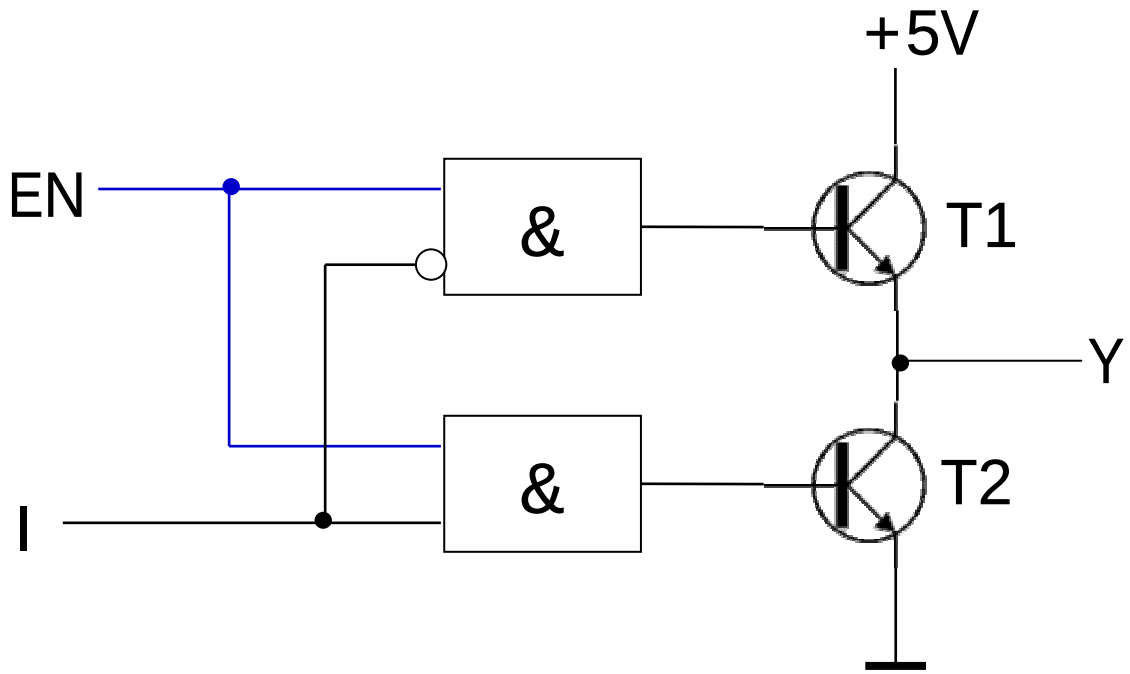
- Eingang I, Ausgang Q
- schaltet bei Überschreiten der Eingangsspannung U_e den Ausgang auf High, d.h. 1
- schaltet bei Unterschreiten der Eingangsspannung U_a den Ausgang auf Low, d.h. 0
- = Hysterese
- analoge Signale → digitale Pegel





7.3 TTL Tri-State Ausgangsschaltung

Ein Tri-State Ausgang kann neben den logischen Pegeln H und L auch noch den Zustand X=„Hochohmig“ annehmen.



- EN = **H**igh
  Y = ¬I
  Schaltung arbeitet als Inverter
  - I=**L**ow, AND1=**H**
    T1 offen → Y = **H**
  - I=**H**igh, AND2=**H**
    T2 offen → Y = **L**
- EN = Low
  beide Tr. sperren
  →Y = hochohmig

Wichtige Anwendungsfälle von Tri-State-Schaltungen sind, wenn mehrere Gatterausgänge zusammengeschaltet sind und wahlweise eines dieser Gatter den logischen Zustand bestimmen sollen. → **Bussysteme**

13. Realisierung digitaler Lösungen

Eine Schaltung soll in gewisser Stückzahl realisiert werden.

Dann gibt es aus technischer und kommerzieller Sicht mehrere Ansätze:

- Full Custom IC
- ASIC (**A**pplication **S**pecific **I**ntegrated **C**ircuit)
- Bausteine mit programmierbarer Logik

13. Full Custom IC

- individuelle Entwicklung eines digitalen Systems
- lange Entwicklungszeiten
- sehr große Stückzahlen
- individuelle Fertigung
- geringer Stückpreis

13. Application Specific IC (ASIC)

- Chip entsteht auf Basis existierender Funktionsblöcke
- Weniger aufwändig als Neudesign des Chip
- Hersteller stellt umfangreiche Bibliotheken für Funktionen zur Verfügung
- verkürzte Entwicklungszeiten
- Schaltung wird mit einer Hardware-Beschreibungssprache beschrieben
- Hersteller realisiert Schaltung auf der Basis eines adäquaten ASIC
- große Stückzahlen
- günstiger Stückpreis

13. Programmierbare Bausteine

- Hersteller bieten programmierbare Logik-Bausteine an
- Lösung wird vom Anwender entwickelt
- hohe Flexibilität
- kleine Stückzahlen
- hoher Stückpreis

13. Programmierbare Bausteine

- PLD (Programmable Logic Device)
- programmierbare Logikelemente (seit Mitte der 70er)
- PLD stellen eine logische Grundstruktur zur Verfügung, die vom Entwickler nach Bedarf konfiguriert (programmiert) werden kann.
- Für hoch integrierte PLD stehen Beschreibungssprachen zur Verfügung.

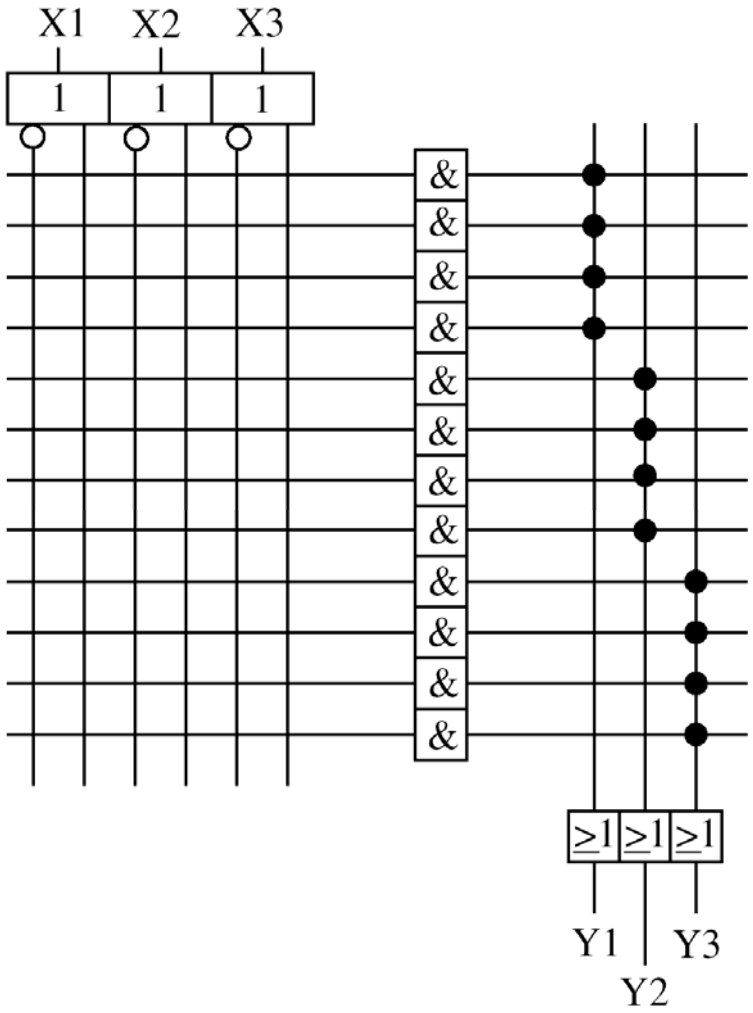
13. Verfahren zur Programmierung

- PROM (Programmable Read Only Memory)-Prinzip: Durchbrennen einer Sicherung (Fuse) oder Entferner einer Isolierung (Antifuse), Programmierung ist irreversibel
- EPROM (Erasable PROM)-Prinzip: Programmierung kann durch Bestrahlung mit UV-Licht wieder gelöscht werden
- EEPROM (Electrical Erasable PROM)-Prinzip: Programmierung kann durch elektrische Impulse wieder gelöscht werden

13.1 PAL (Programmable Array Logic)

- Realisierung logischer Gleichungen in disjunktiver Form.
- Alle Eingangsgrößen werden in negierter und nicht-negierter Form zur Verfügung gestellt.
- Programmierbares UND-Feld das mit den Eingangsgrößen verbunden ist.
- Fest verdrahtetes ODER-Feld.

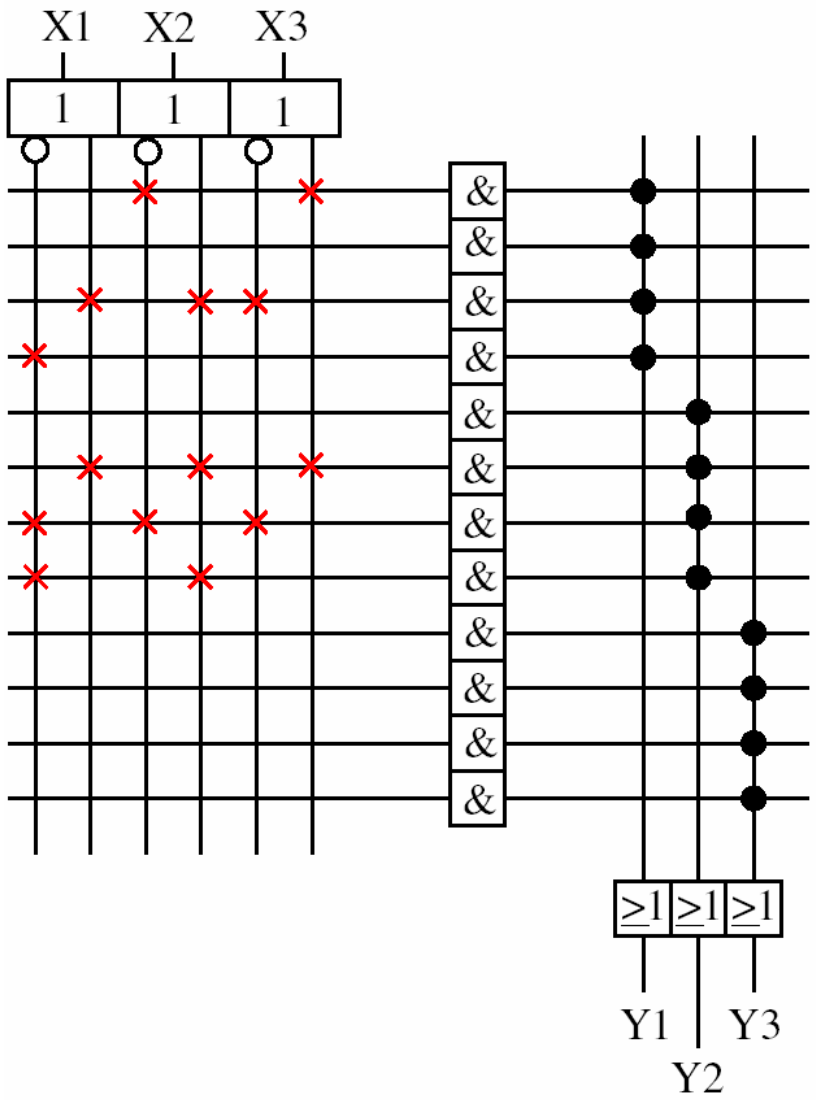
13.1 Prinzip PAL



Frei programmierbare UND-GATTER

Fest verschaltete ODER-Gatter

13.1 Beispiel PAL



$$Y1 = (\bar{X}2 \wedge X3) \vee (X1 \wedge X2 \vee \bar{X}3) \vee \bar{X}1$$

$$Y2 = (X1 \wedge X2 \wedge X3) \vee$$
$$(\bar{X}1 \wedge \bar{X}2 \wedge \bar{X}3) \vee$$
$$(\bar{X}1 \wedge X2)$$

13. Typisierung

- PAL (Programmable Array Logic):

  Programmierbare UND-Matrix, feste Oder-Matrix, von einem Hersteller auch als GAL (Generic Array Logic) bezeichnet
- PLE (Programmable Logic Element):

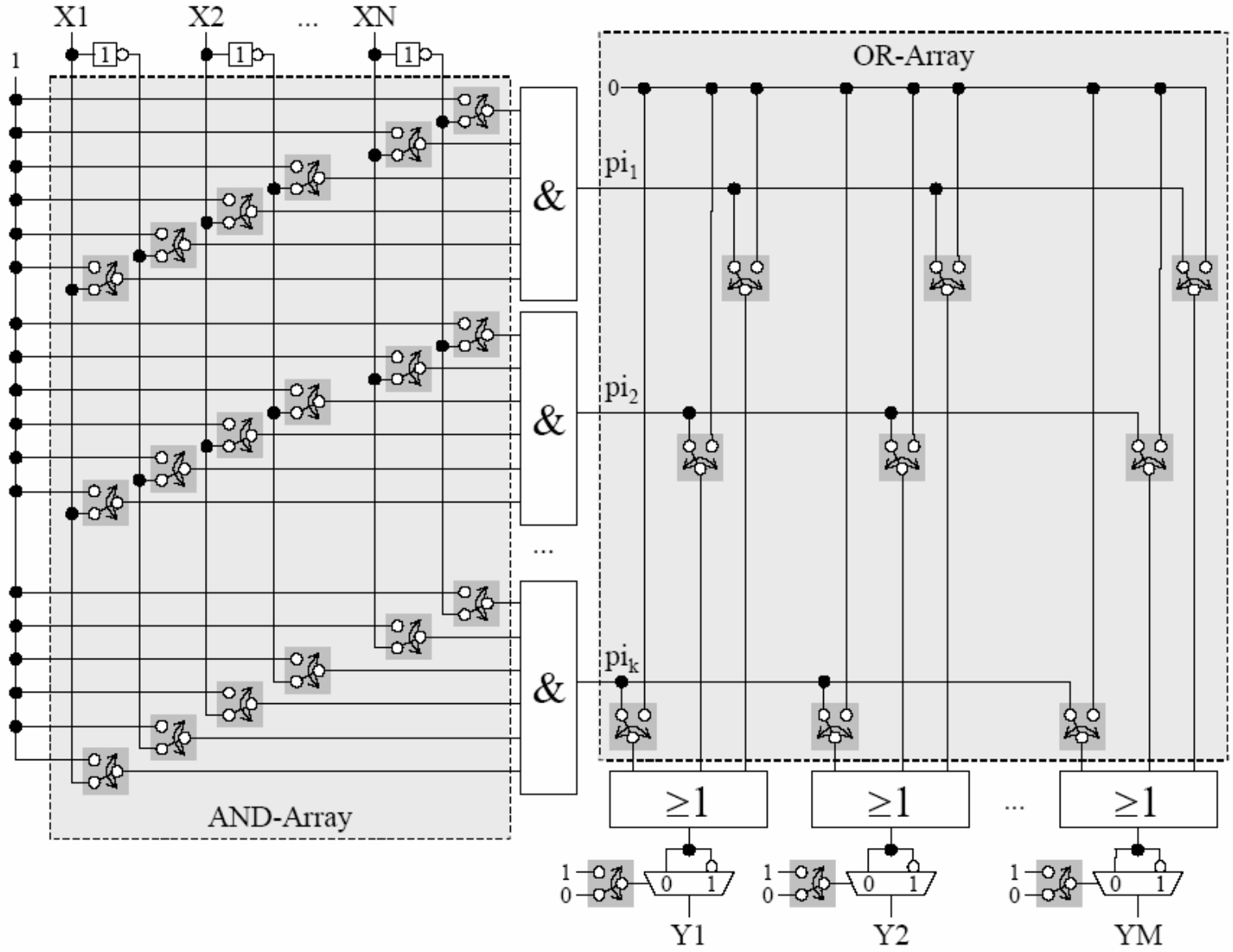
  Programmierbare Oder-Matrix, feste Und-Matrix
- PLA (Programmable Logic Array):

  Programmierbare UND-Matrix und programmierbare ODER-Matrix

13.2 PLA (Programmable Logic Array)

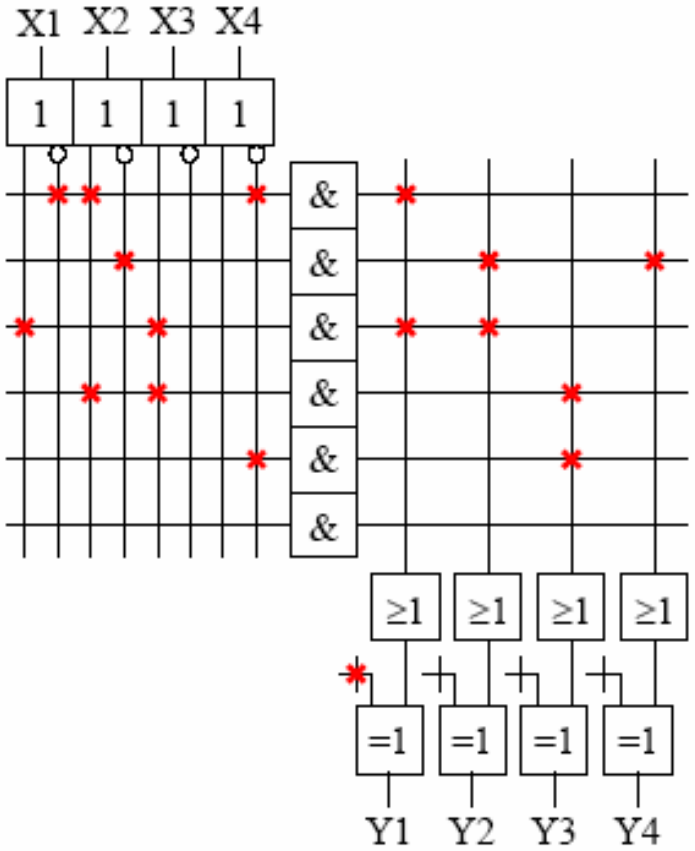
- Mit dem frei programmierbaren AND- und OR-Array können Funktionen in disjuktiver Normalform realisiert werden.
- Durch die programmierbaren Ausgangsinverter ist aufgrund des Shannonschen Gesetzes auch die Realisierung der konjunktiven Form möglich.

13.2 PLA (Programmable Logic Array) (2)



13.2 PLA (Programmable Logic Array) (3)

Üblicherweise wird diese logische Grundstruktur in einer kompakten Form dargestellt, wo bei jedem Gatter eine Linie die Gattereingänge symbolisiert.



$$Y1 = \overline{(\overline{X1} \wedge X2 \wedge \overline{X4}) \vee (X1 \wedge X3)}$$

$$= (X1 \vee \overline{X2} \vee X4) \wedge (\overline{X1} \vee \overline{X3})$$

$$Y2 = \overline{X2} \vee (X1 \wedge X3)$$

$$Y3 = (X2 \wedge X3) \vee \overline{X4}$$

$$Y4 = \overline{X2}$$

Implikanten können in mehreren Funktionen gemeinsam genutzt werden.

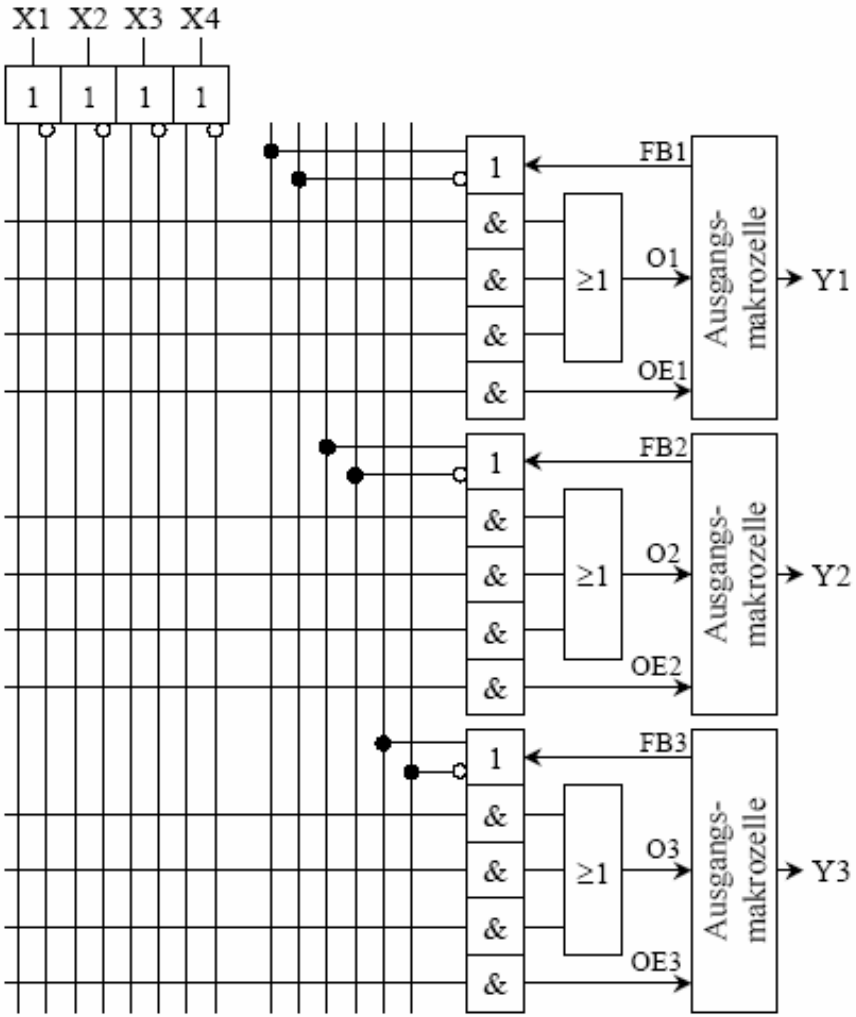
13.2 PLA (Programmable Logic Array) (4)

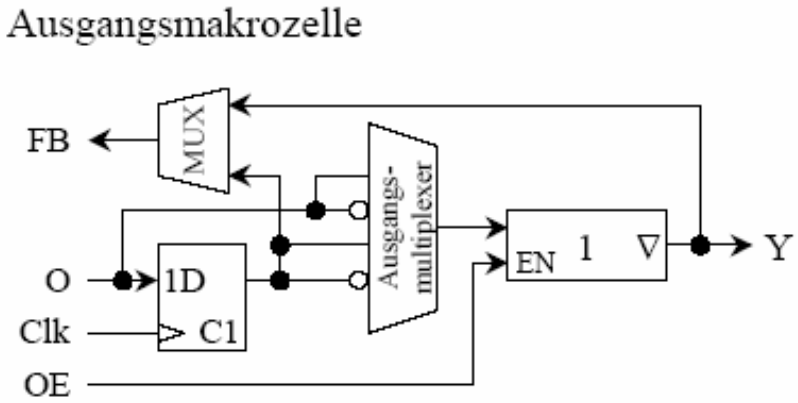
- Die Verwendung von PLA-Bausteinen ist jedoch eher selten.
- Die gebräuchlichen PAL- und Speicher-Bausteine schränken die Möglichkeiten der Programmierung auf ein sinnvolles Maß ein, so dass deren Verwendung gut handhabbar ist.

13.1 Erweiterung von PAL

- Einfache PAL Elemente haben mindestens 8 Ein- und Ausgänge
- Moderne PAL Bausteine verfügen über komplexe, programmierbare Makrozellen
- die Ausgänge verfügen über Register
- die Ausgänge können zurück gekoppelt werden

13.1 PAL, erweiterte Struktur

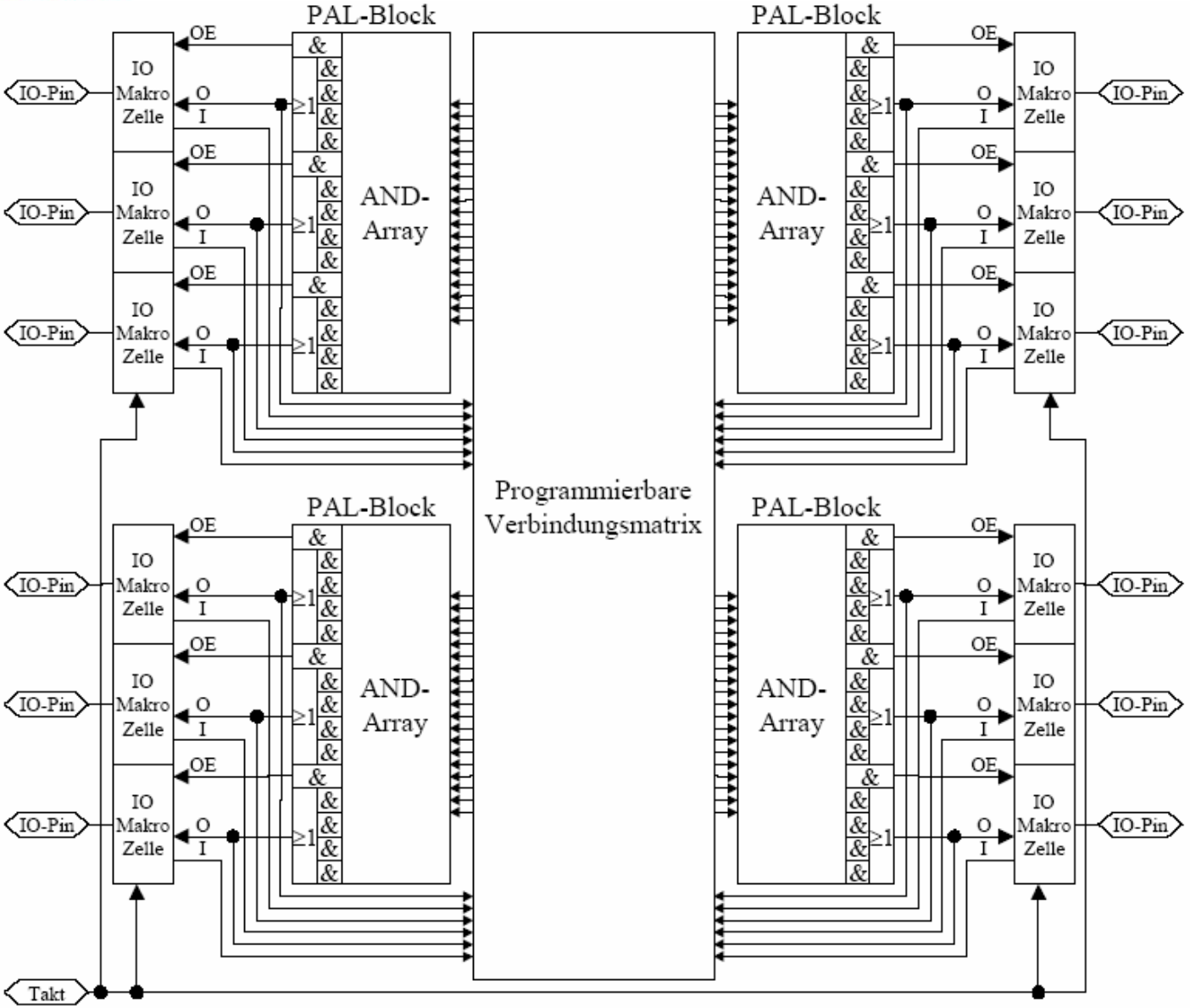




13.3 CPLD (Complex Programmable Logic Device)

- komplexe PLDs mit einer Block-Struktur
- jeder Block entspricht einem einfachen PAL
- die Blöcke werden über eine programmierbare Schaltmatrix miteinander verbunden
- ein einzelner Block enthält typischerweise ca. 50 Eingänge und 10-20 Ausgänge
- jeder Ausgang kann aus 10-15 Produkttermen gebildet werden

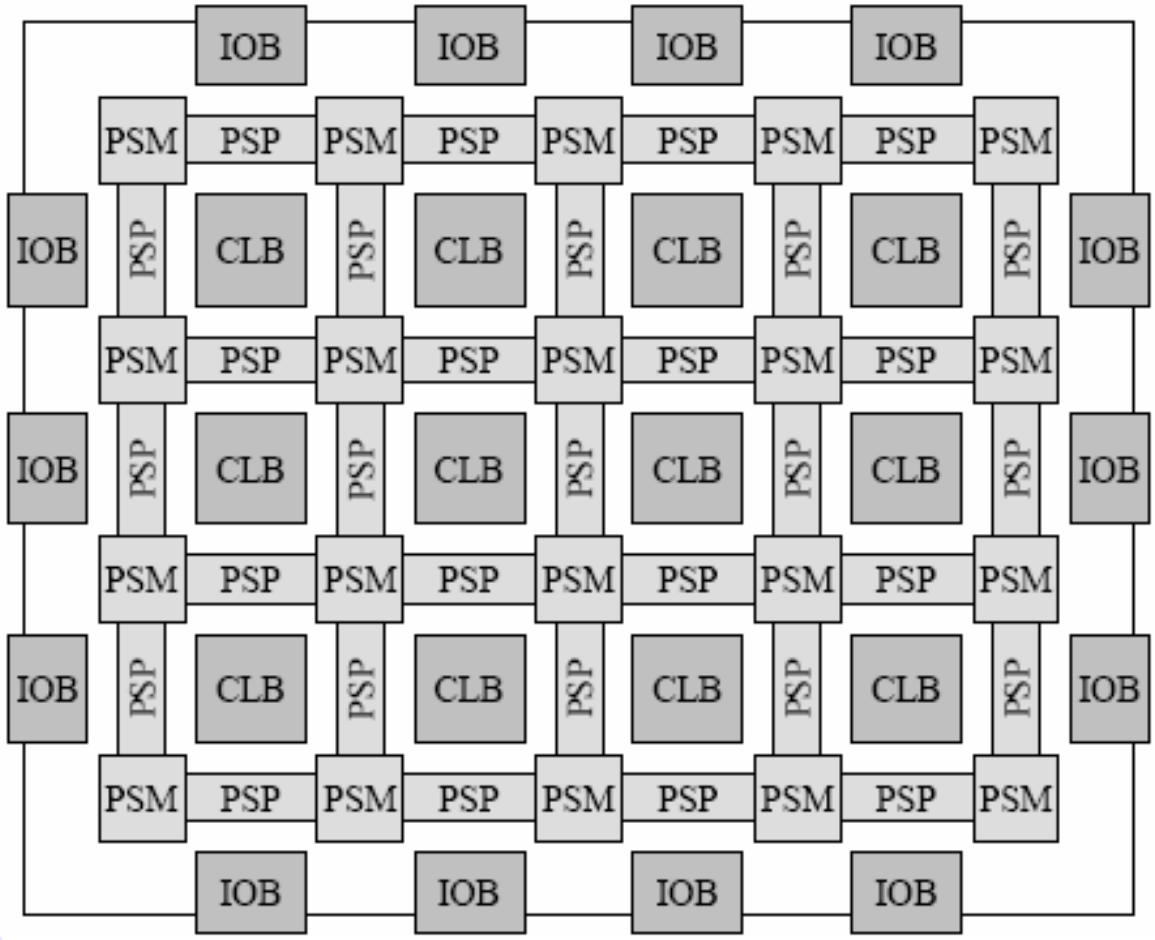
13.3 CPLD (2)


PAL-Block
PAL-Block
PAL-Block
PAL-Block
AND-Array
AND-Array
AND-Array
AND-Array
Programmierbare Verbindungsmatrix
IO Makro Zelle
IO-Pin
OE
O
I
&
≥1
Takt

15 FPGA (Field Programmable Gate Array)

- frei programmierbarer Logikschaltkreis
- aus einzelnen Logikblöcken (CLBs Configurable Logic Blocks) aufgebaut
- in den einzelnen Blöcken werden einfache Operationen und auch Flip-Flop-Logik zur Verfügung gestellt
- teilweise werden FPGAs ausschließlich über Look-Up Tabellen (LUT) realisiert
- hohe Komplexität
- Selbstkonfigurierende Systeme werden möglich

15 FPGA (2)



CLB: Configurable Logic Block

PSP: Programmable Switching Points

PSM: Programmable Switching Matrix

IOB: I/O Block