IUK–Tage 2003

Institut für Angewandte Mikroelektronik und Datentechnik

IUK-Tage 2003

„Design flow zur Entwicklung Geschwindigkeits- und Leistungsoptimierter Schaltungen“

Frank Sill, Frank Grassert, Dirk Timmermann
Institut für Angewandte Mikroelektronik und Datentechnik
Fachbereich für Elektro- und Informationstechnik
Universität Rostock

20.06.2003

Gliederung

1. Einführung
2. Grundlagen
3. Design flow
4. Optimierung
5. Beispiel
6. Zusammenfassung

Einführung

- Bei Audio-, Video-, Smartcard-Anwendungen liegen Datenströme vor
- Pipelines geeignet für Datenstromverarbeitung
- Mit dynamischer Logik hohe Taktfrequenzen möglich
- Asynchron getaktete Schaltungen bieten Vorteile beim *Clock Skew* und der Latenzzeit

Gliederung

1. Einführung
2. Grundlagen
   - Dynamische Logiken
   - Selbstgetaktete Ketten
   - *Self-timed* Logiken
   - Schaltungsaufbau
3. Designflow
4. Optimierung
5. Beispiel
6. Zusammenfassung

Dynamische nMOS-Logik

Realisierung der Funktion über nMOS-Transistoren

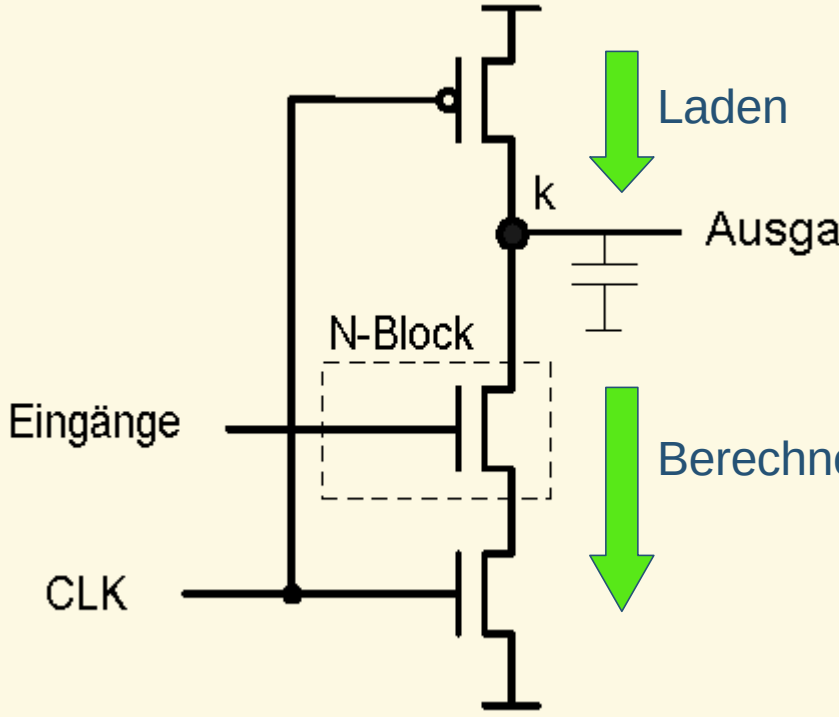
Taktsignal notwendig

<u>Ladephase:</u>

- CLK ist *Low*
- Knoten k wird geladen

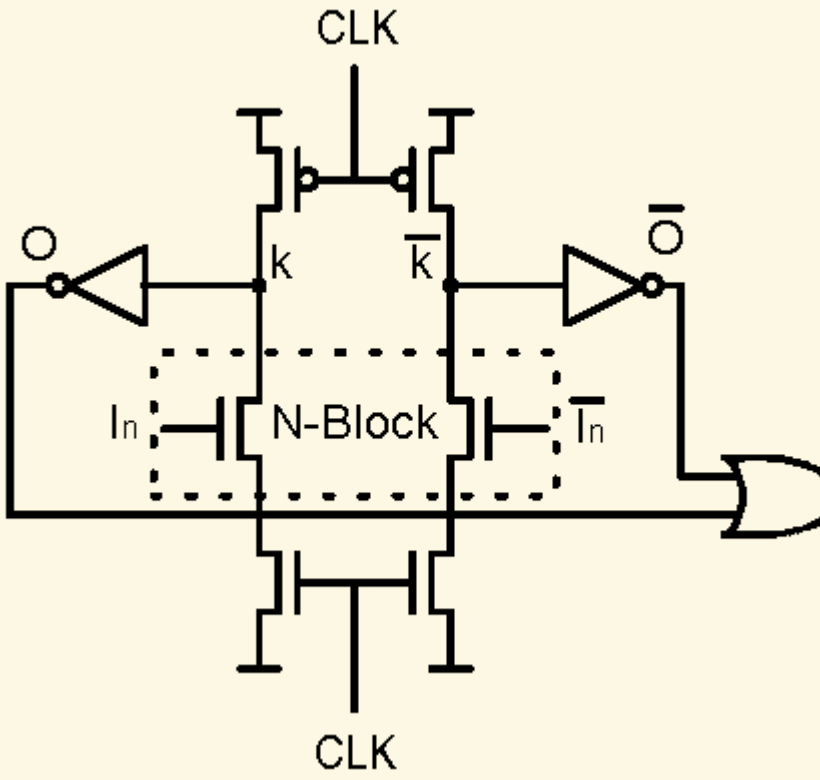
<u>Berechnungsphase:</u>

- CLK ist *High*
- k entlädt sich in Abhängigkeit der Eingänge



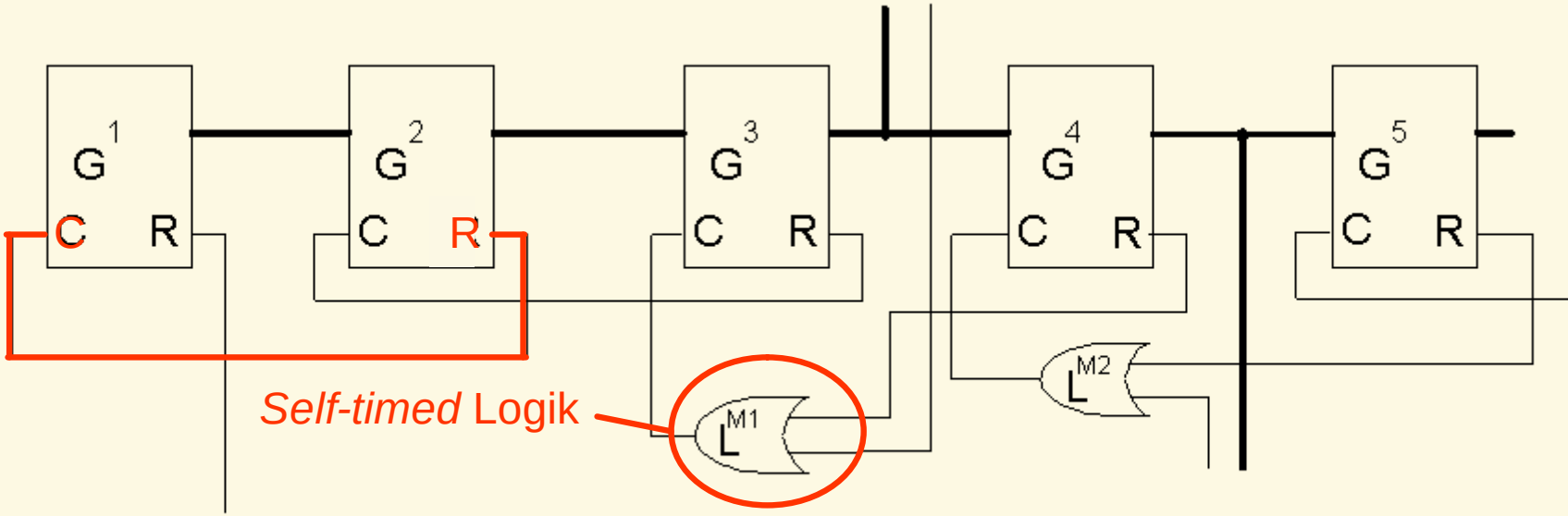
Dual-Rail Gatter

- durch komplementärer Aufbau entlädt sich immer ein Knoten
- Zustandssignal R gibt an, ob die Berechnung bzw. der Aufladevorgang abgeschlossen wurde
- Inverter verhindern vorzeitige Entladung



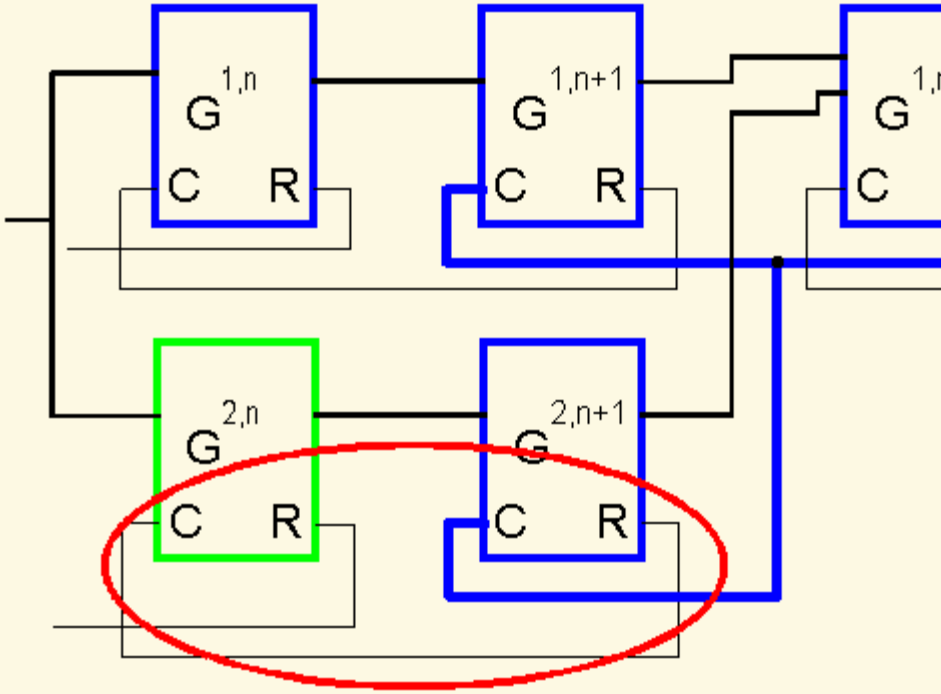
Selbstgetaktete Kette

1. Zustandssignale mit Steuereingängen verbinden
2. *Self-timed* Logiken hinzufügen



Self-timed Logiken

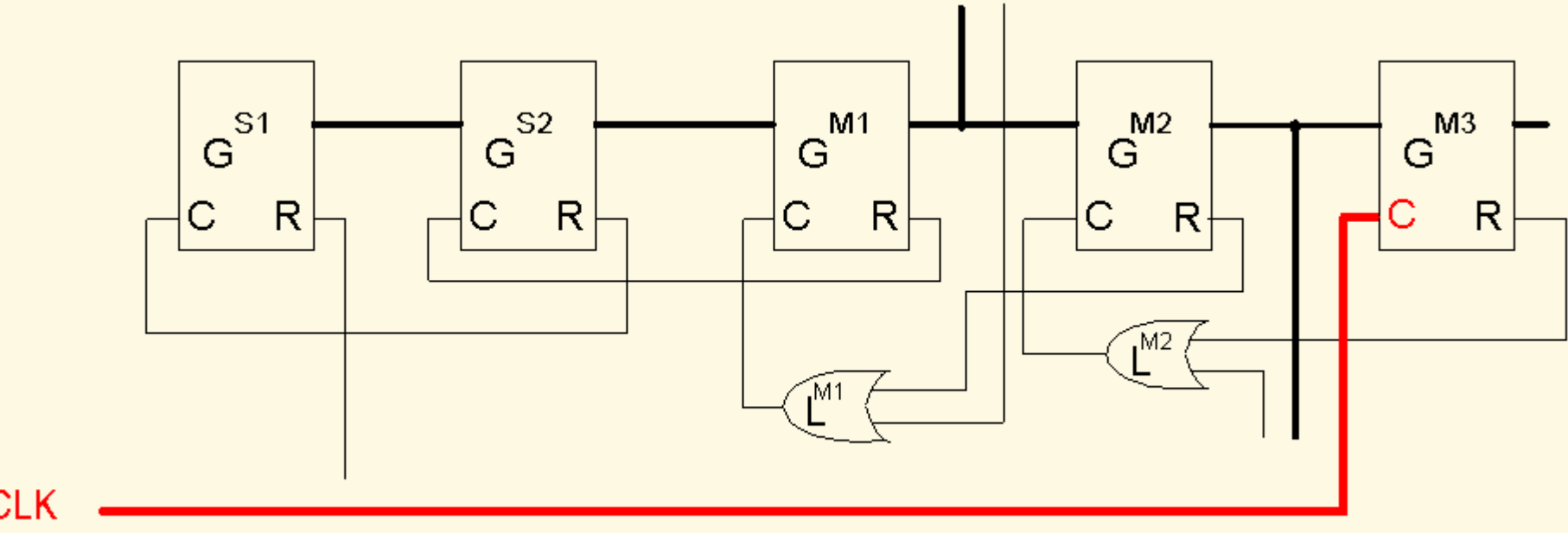
1. $G^{1,n}$ und $G^{2,n}$ berechnen
2. $G^{1,n+1}$ und $G^{2,n+1}$ berechnen
3. - $G^{1,n+1}$ setzt $G^{1,n}$ in Ladephase
   - $G^{2,n+1}$ berechnet weiterhin
   - $G^{1,n+2}$ berechnet
4. - $G^{1,n+1}$ und $G^{2,n+1}$ wecheln in Ladephase
   - ➢ $G^{2,n}$ wird nicht in Ladephase versetzt



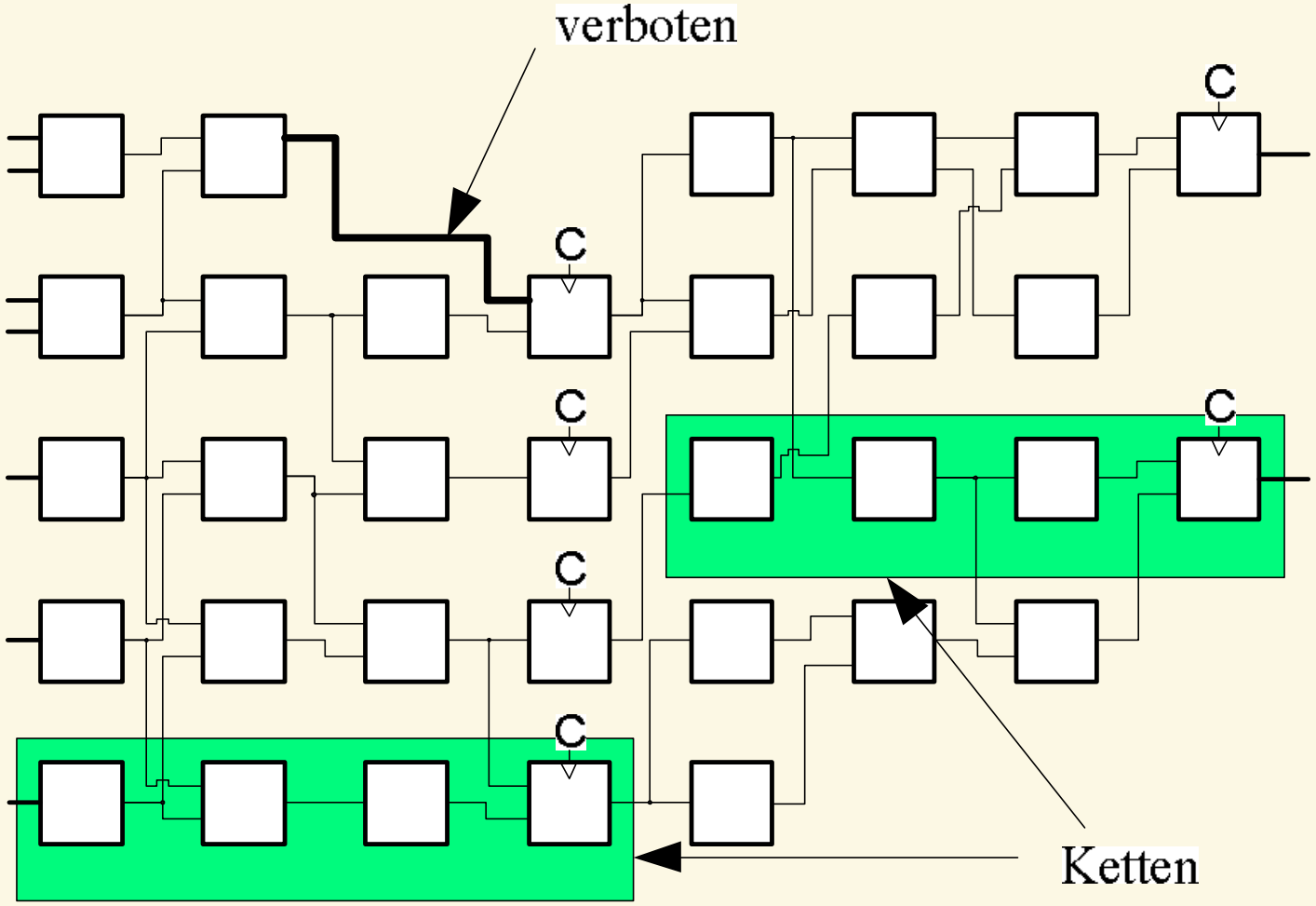
Zusätzliche Logik notwendig

Aufbau von Asynchronous Chain True Single Phase Clock (AC-TSPC)

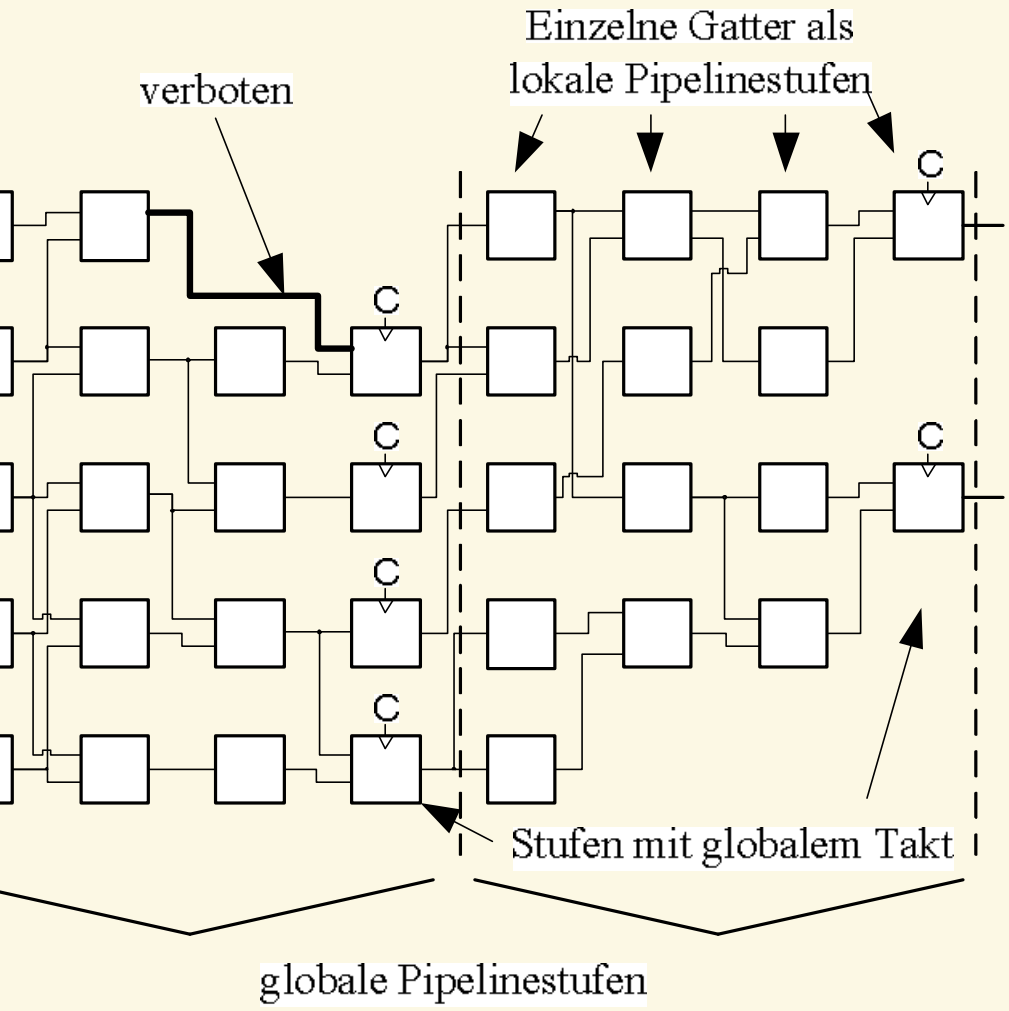
- Letztes Gatter der Ketten mit Taktsignal verbinden
- Ketten können in synchron getaktetes Design integriert werden

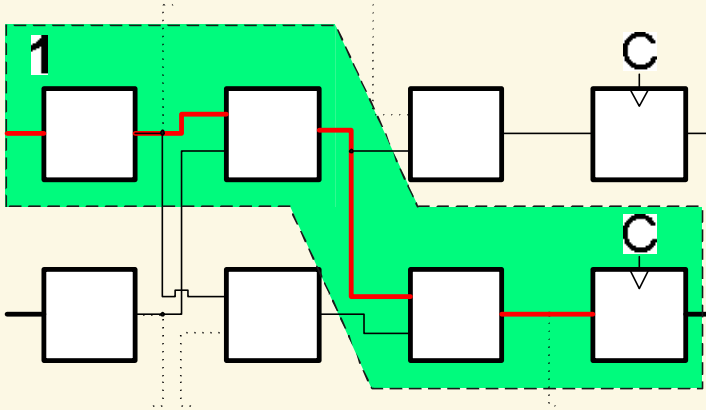


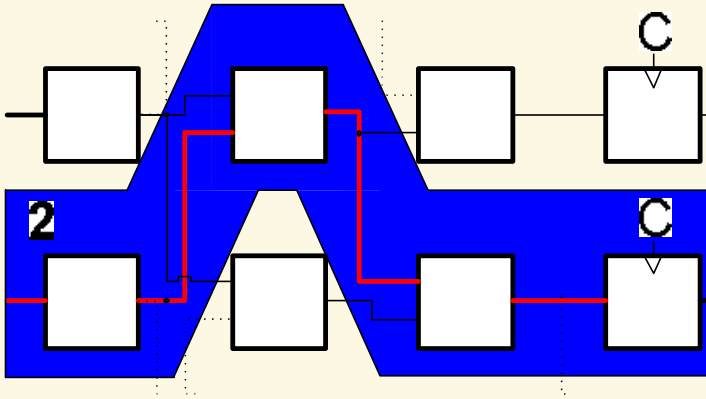
Aufbau der AC-TSPC Schaltungen



Aufbau der AC-TSPC Schaltungen



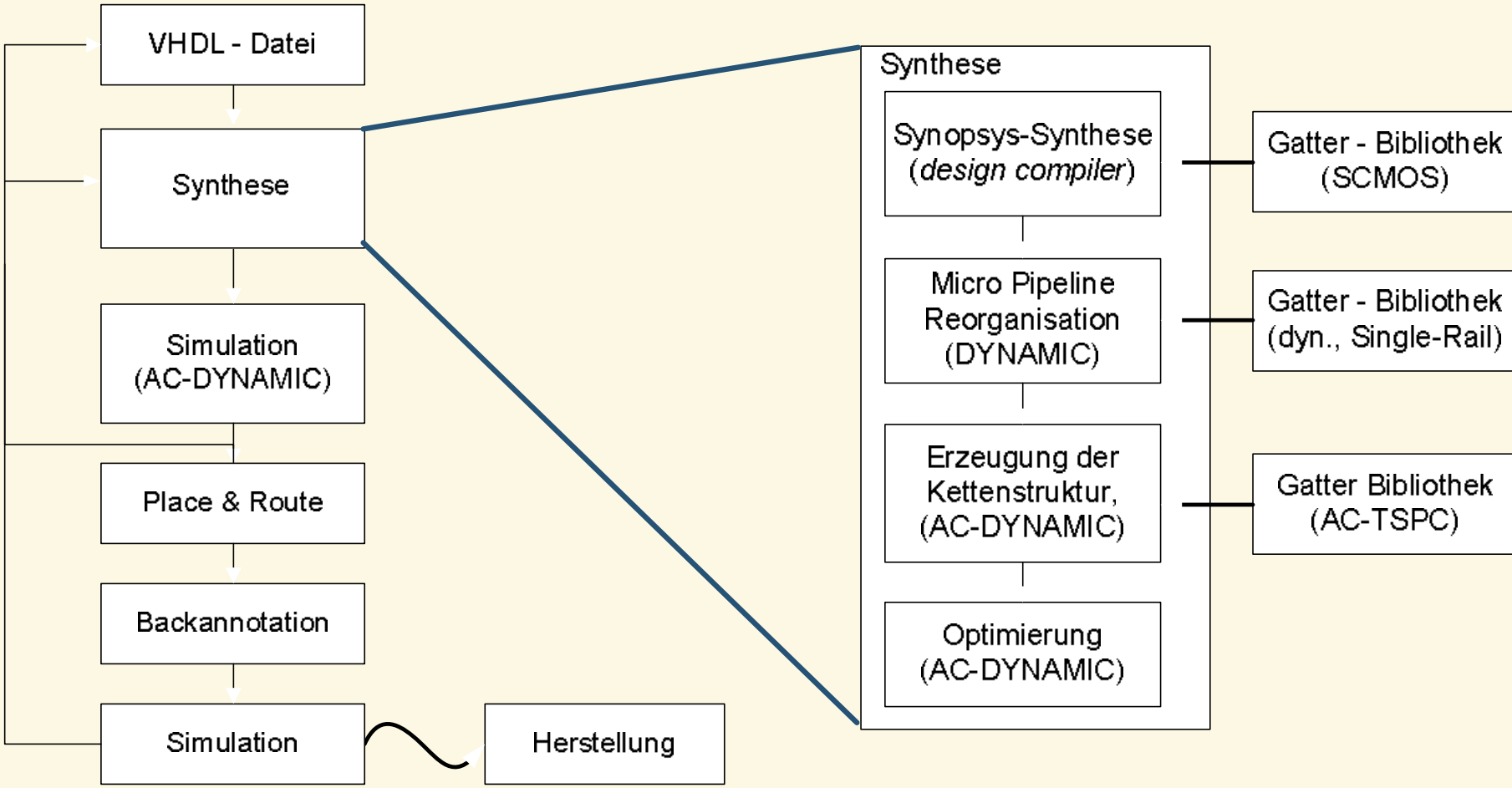




Gliederung

1. Einführung
2. Grundlagen
3. Designflow
   - Aufbau
   - Bibliotheksgenerierung
   - Micro Pipeline Reorganization (MPR)
   - AC-DYNAMIC
4. Optimierung
5. Beispiel
6. Zusammenfassung

Entwickelter Design flow



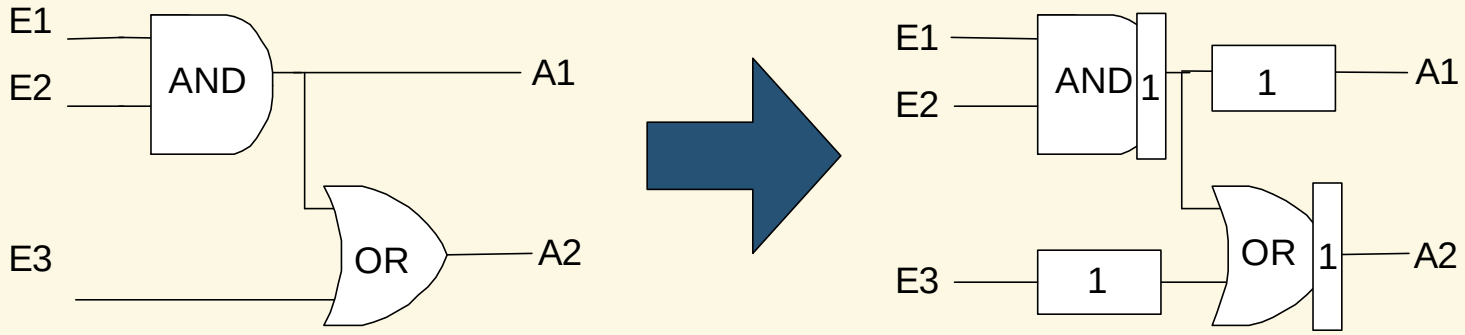
Erstellen neuer Bibliotheken

Bibliotheken:

- statisch CMOS
  - Beschreibung der Kombinatorik
  - Registerfkt. & Zeitverhalten ist uninteressant
  - Kompilierung mit *library compiler* (SYNOPSYS)
- dynamisch Single Rail
  - Schematic, Layout, Verifikation, Simulation
  - Kompilierung mit *library compiler*
- dynamisch Dual-Rail
  - Schematic, Layout, Verifikation, Simulation
  - Bestimmung des Zeitverhaltens

Micro Pipeline Reorganization (MPR)

- Schaltung als Pipeline umorganisiert (DYNAMIC)
- Ersetzen der kombinatorischen Gatter durch äquivalente dynamische Gatter ➡ verfügen über Registerfunktionalität
- Hinzufügen von Registern zur Sicherung des zeitlichen Verhaltens



AC-DYNAMIC

- Einlesen der Netzliste einer als Pipeline aufgebauten Schaltung
- Generierung der Kettenstruktur
- Einfügen der *Self-timed* Logiken
- Optimierungen
- Festlegung des Taktsignals
- Simulation der Schaltung

Gliederung

1. Einführung
2. Grundlagen
3. Designflow
4. Optimierungen
   - Optimierungsziele
   - Optimierungsansätze
5. Beispiel
6. Zusammenfassung

Optimierungsziele

- hohe Taktfrequenz
- geringe Latenzzeit
- geringe zusätzliche Fläche
- geringer Leistungs- und Energieverbrauch

Optimierungen der *Self-timed* Logiken

- Realisierung
  - universale *Self-timed* Logik
    - garantiert Funktionsfähigkeit unter allen Bedingungen
  - angepasste *Self-timed* Logik
    - für jeden Sonderfall spezielle Logik
  - Mehrfachbenutzung von Zustands-/Steuersignalen
- Schrittweise Selektion von Möglichkeiten

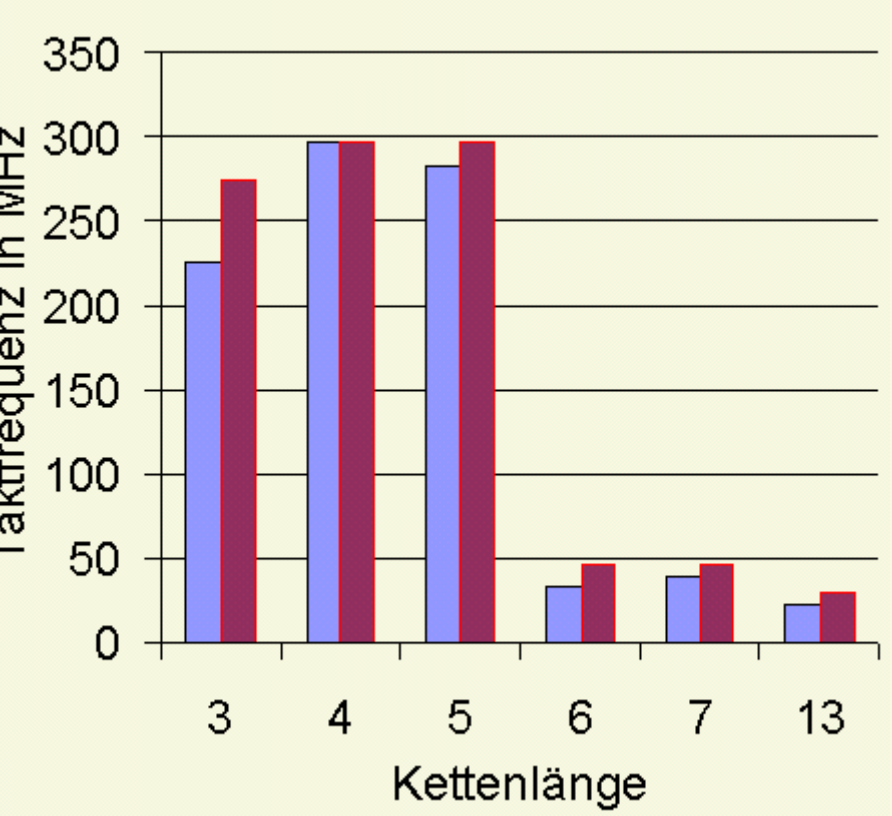
Optimierungen des Taktsignals

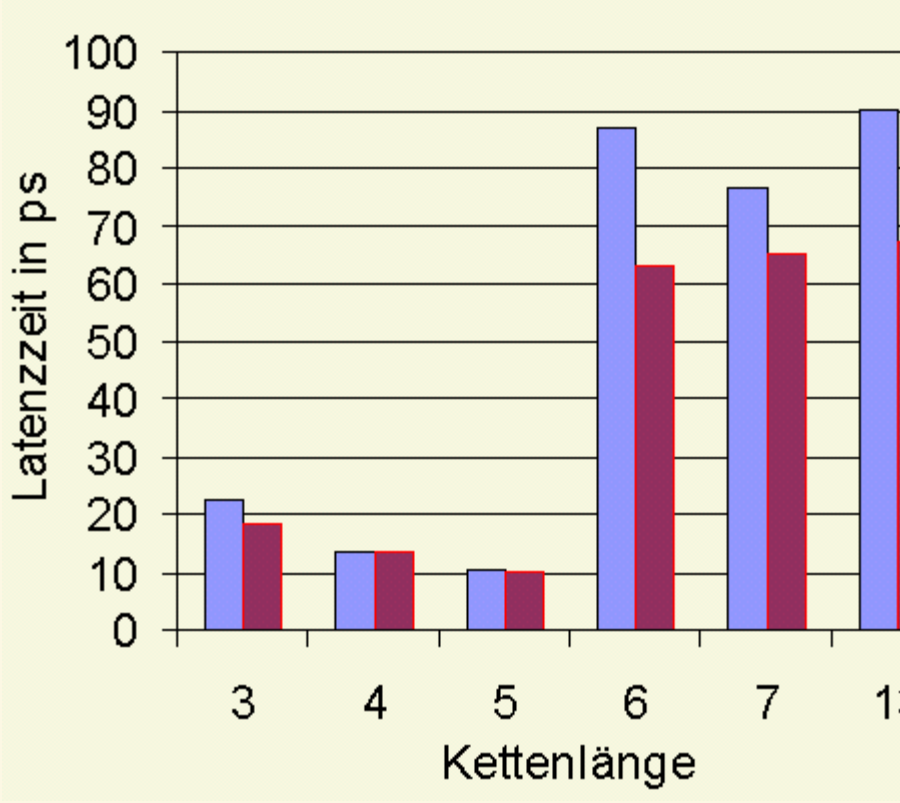
- Frequenz
  - Bedingungen für Dauer der Phasen
  - Phasendauer sind untereinander abhängig
- Symmetrie
  - unsymmetrisches Taktsignal erlaubt bessere Anpassung an notwendige Phasendauer
  - symmetrisches Taktsignal kann besser synthetisiert werden

Optimierungen der Kettenlänge

- Kettenlänge variiert beliebig
  - ➢nicht empfohlen, da viele Fehler
- variable Kettenlänge innerhalb einer Ebene (parallele Ketten haben gleiche Länge)
- sinnvoll bei “Dreiecksform” der Pipelines

Beispiel: 2x4-Bit Multiplizierer





symmetrisches Taktsignal

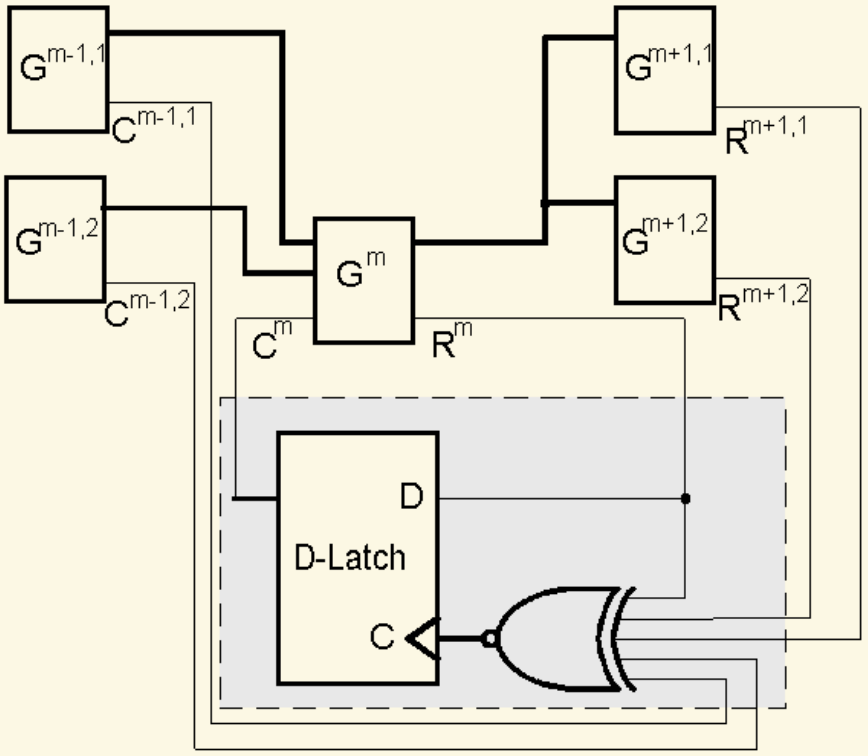
unsymmetrisches Taktsignal

Zusammenfassung

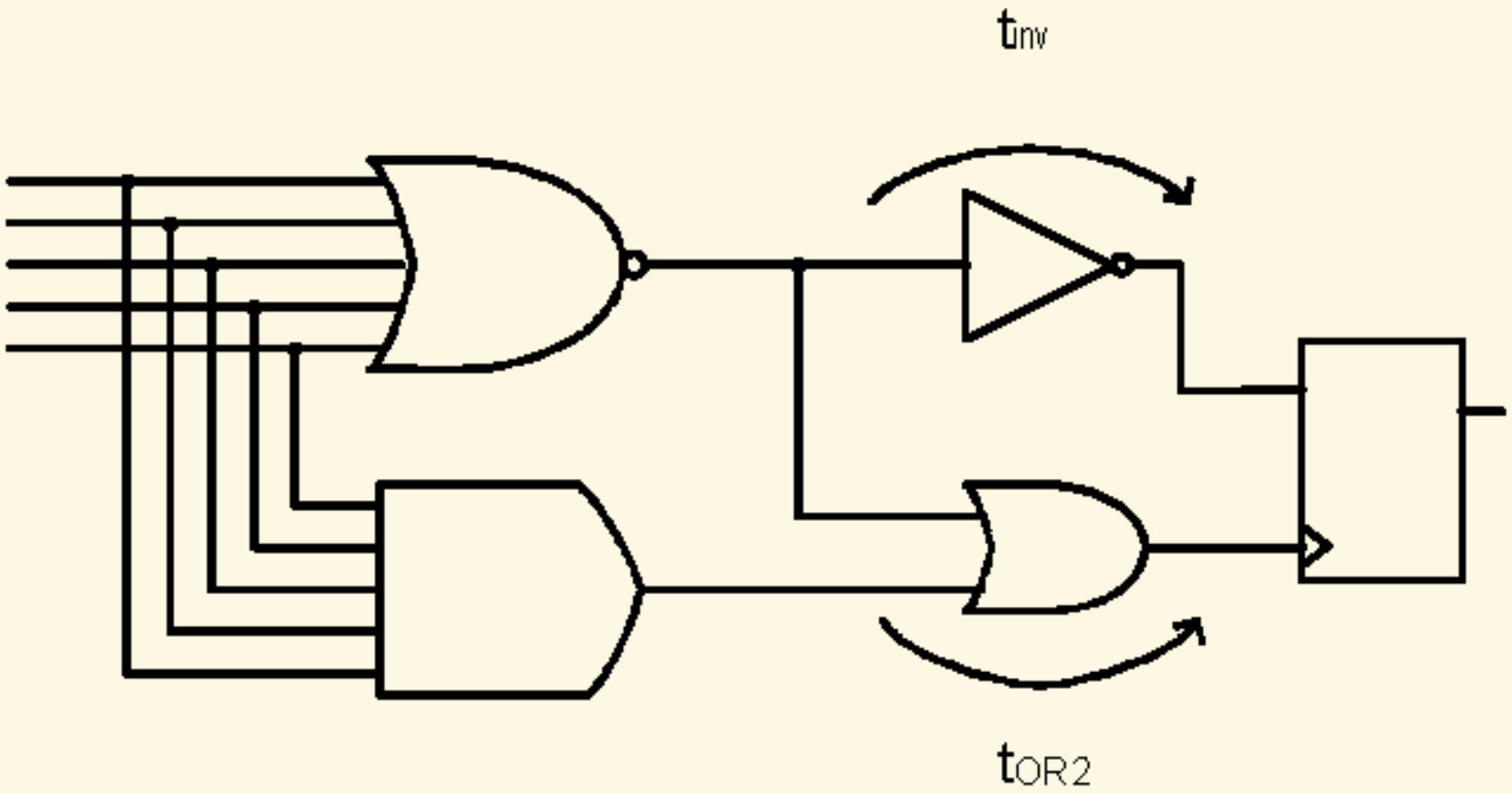
- Reduzierter Leistungsverbrauch und geringe Latenzzeit bei kurzen asynchronen Ketten in einem global synchronen Design
- Design flow zur Generierung von AC-TSPC Schaltungen erstellt
- Notwendige Tools entwickelt
- Optimierungen implementiert

Vielen Dank für Ihre Aufmerksamkeit!

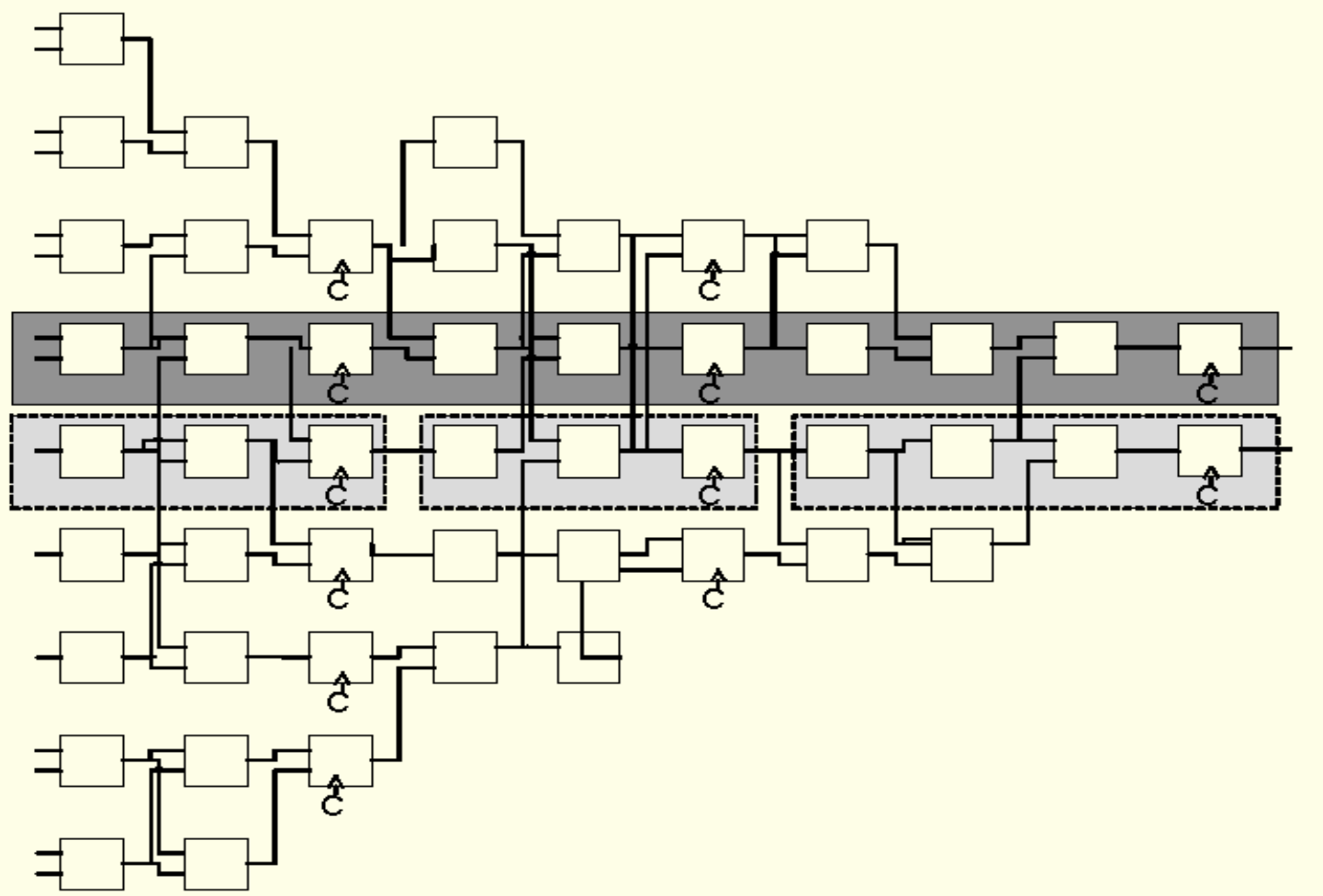
Zusatz: universale Logik I



Zusatz: universale Logik II



Zusatz: variable Kettenlänge



Zusatz: virtuelle Buffer